ICS Part-II · Punjab Board · English Medium

Second Year Computer Science

Complete Notes — All 14 Chapters

freebooks.pk

Table of Contents

Prepared by freebooks.pk — original student notes covering the Punjab Curriculum and Textbook Board (PTB/PCTB) Intermediate Computer Science (ICS) syllabus. Each chapter includes key concepts, definitions, key facts, diagrams, solved examples, short and long questions, MCQs, revision and exam tips.

Freebooks.pk

Free Notes • Past Papers • Guides for Students

2nd Year Computer Science — Punjab Board

CHAPTER 1

Data Basics

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers Data Basics from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains data and information, the traditional file system and its drawbacks, databases, and the database management system (DBMS). These notes are prepared by freebooks.pk.

Storing and managing data is one of the most important uses of computers. This chapter introduces databases, which have replaced older file systems for keeping large amounts of data organised.

Learning Objectives

- Distinguish between data and information.
- Describe the traditional file system and its drawbacks.
- Explain what a database is and its advantages.
- Define a database management system (DBMS).
- State the functions and advantages of a DBMS.
- Give examples of database use.

Key Concepts

Data and Information

Data means raw, unorganised facts and figures, such as names, numbers or marks, which on their own may have little meaning. Information is data that has been processed, organised or presented in a meaningful and useful form, such as a total, an average or a report. The computer processes data to produce information; for example, a list of marks (data) can be processed to give the average (information). Turning data into useful information is a main purpose of computing.

The Traditional File System

Before databases, data was kept in a traditional file system, in which each application program had its own separate data files. This approach had serious drawbacks. The same data was often stored in several files, causing data duplication (redundancy) and wasting space; the copies could disagree, causing data inconsistency; the data was tied to particular programs (data dependence); and it was hard to share data between programs. These problems made the file system difficult to manage as data grew.

What is a Database?

A database is an organised collection of related data stored together in one place so that it can be easily accessed, managed and updated. Instead of separate files for each program, all the data is kept in a single shared database that many programs and users can use. Because the data is stored once and shared, a database greatly reduces duplication and inconsistency and makes the data easier to control and to search.

The Database Management System (DBMS)

A database management system (DBMS) is the software that creates, stores, manages and controls access to a database. It sits between the users (or their programs) and the stored data, so that all access to the database goes through it. Examples of DBMS software include Microsoft Access, Oracle and MySQL. The DBMS allows users to define the data, enter and update it, and retrieve it using queries, while keeping the data secure and consistent.

Functions of a DBMS

A DBMS performs several important functions. It lets users define the structure of the data (the tables and fields). It allows data to be inserted, updated and deleted. It lets users retrieve data by asking questions called queries. It controls who may see or change the data (security), keeps the data accurate (integrity), and allows several users to use the data at the same time without conflict. It can also make backups so that data can be recovered if it is lost.

Advantages of a Database Approach

Using a database with a DBMS has many advantages over the old file system. It reduces data duplication because data is stored once and shared; it improves data consistency and accuracy; it makes data easy to search and to update; it allows data to be shared safely among many users; and it provides security and backup. These advantages are why databases are now used to store the records of banks, schools, hospitals, businesses and governments.

Uses of Databases

Databases are used wherever large amounts of related data must be stored and used. A bank uses a database to keep customers' accounts, a school to keep students' records and results, a hospital to keep patients' records, an airline to keep bookings, and a shop to keep stock and sales. In each case the database, managed by a DBMS, keeps the data organised, accurate and available to those who need it.

Important Definitions

Term	Definition
Data	Raw, unorganised facts and figures.
Information	Processed, organised data that is meaningful and useful.
File system	An older method where each program keeps its own separate data files.
Data redundancy	Unnecessary duplication of the same data in several places.
Data inconsistency	Disagreement between duplicated copies of data.
Database	An organised collection of related data stored together.
DBMS	Software that creates, manages and controls access to a database.

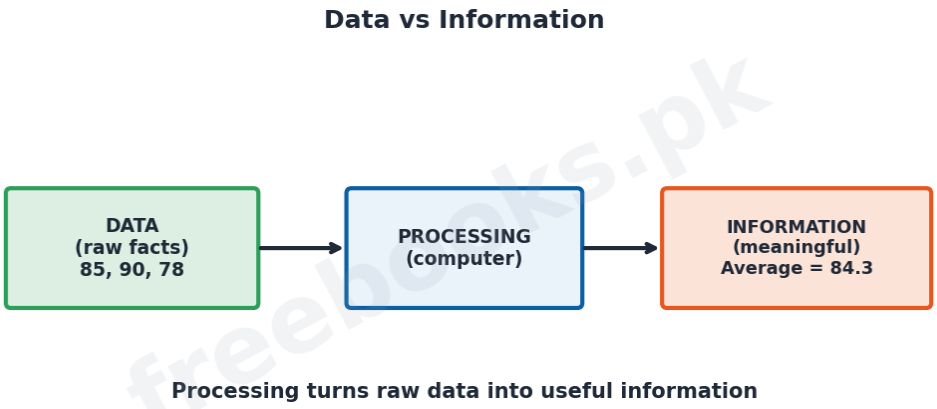
Term	Definition
Query	A question used to retrieve particular data from a database.

Key Facts & Rules

Item	Detail
Data	Raw facts (e.g. 85, 90, 78)
Information	Processed data (e.g. average = 84.3)
File system drawbacks	Redundancy, inconsistency, data dependence
Database	One shared, organised collection of data
DBMS examples	MS Access, Oracle, MySQL
DBMS functions	Define, insert, update, delete, query, secure, backup

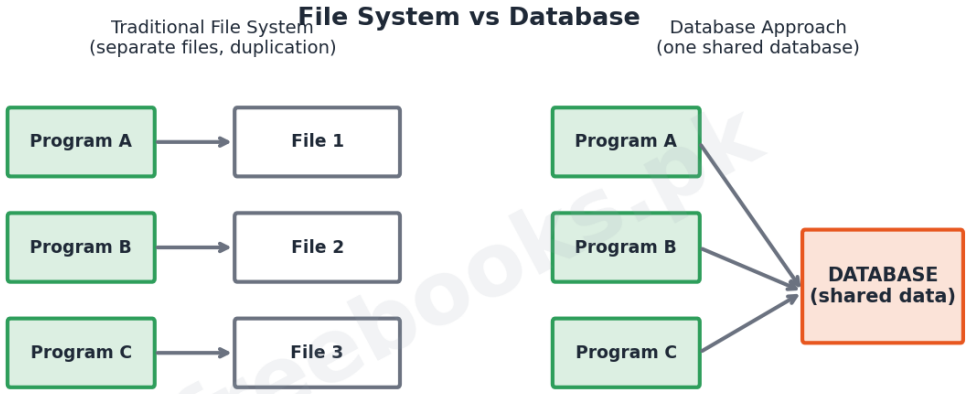
Diagrams & Illustrations

Data vs information: raw data being processed by the computer into meaningful information.



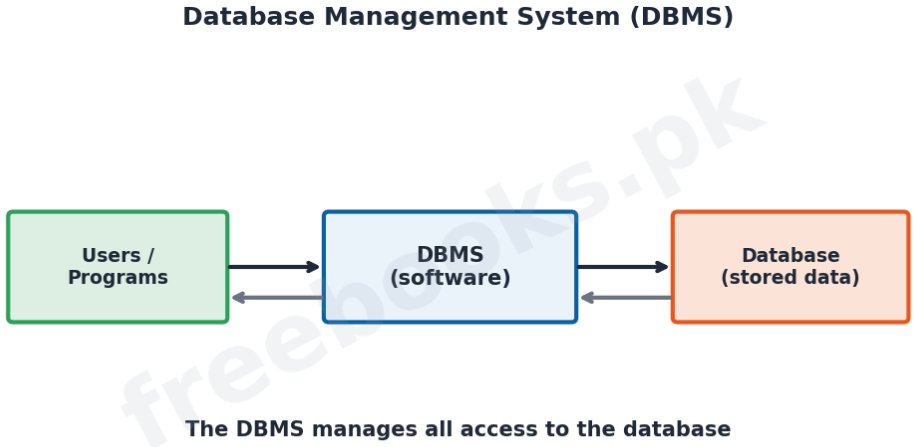
Data vs information

File system vs database: the traditional file system with separate files for each program compared with a single shared database.



File system vs database

Database management system: users and programs accessing a database only through the DBMS software.



Database management system

Solved Examples & Practice

Data or information

A list of exam marks is data; the class average worked out from them is information.

Spot the drawback

If a student's address is stored in three different files and one is updated, the others become wrong; this is data inconsistency caused by redundancy.

Identify the software

Microsoft Access is an example of a DBMS used to create and manage databases.

Choose a query

To list all students who scored above 80, the user runs a query on the database.

Short Questions & Answers

Q1. Differentiate between data and information.
Ans. Data is raw, unorganised facts; information is data that has been processed into a meaningful and useful form.

Q2. State two drawbacks of the traditional file system.
Ans. Data redundancy (the same data stored many times) and data inconsistency (copies that disagree); data is also tied to programs.

Q3. What is a database?
Ans. An organised collection of related data stored together so it can be easily accessed, managed and updated.

Q4. What is a DBMS?
Ans. Database management system, the software that creates, manages and controls access to a database (e.g. MS Access).

Q5. What is a query?
Ans. A question used to retrieve particular data from a database.

Q6. State two advantages of a database over a file system.
Ans. It reduces data duplication and improves consistency; it also makes data easier to share and secure.

Long Questions & Answers

Q1: Differentiate between data and information and explain the drawbacks of the traditional file system.

Although the words data and information are often used to mean the same thing, in computing they are different. Data means raw, unorganised facts and figures, such as a set of names, numbers or examination marks, which on their own may carry little meaning. Information is data that has been processed, that is, organised, summarised or presented in a form that is meaningful and useful to the person receiving it; for instance, a plain list of marks is data, but the class average or a report of pass and fail results worked out from those marks is information. The computer's task is largely to process data into information. Before databases became common, data was kept using a traditional file system, in which every application program had its own separate set of data files. This method suffered from several serious drawbacks. First, because each program kept its own files, the same item of data, such as a customer's address, was often stored in several different files, causing data redundancy, or unnecessary duplication, which wasted storage space. Second, when one copy of such duplicated data was changed but the others were not, the copies disagreed, producing data inconsistency, so it was no longer clear which copy was correct. Third, the

data was closely tied to the particular program that used it, a problem called data dependence, so changing the data format meant changing the programs. Finally, it was difficult to share data between different programs. These weaknesses made the file system hard to manage as the amount of data grew, and led to the development of the database approach.

Q2: Explain what a database and a database management system (DBMS) are, and state the functions of a DBMS.

A database is an organised collection of related data that is stored together in one place so that it can be easily stored, accessed, managed and updated. Instead of keeping a separate set of files for every program, as the old file system did, the database approach keeps all the related data in a single, shared collection that many programs and users can use at the same time. Because each item of data is normally stored only once and then shared, a database greatly reduces the duplication and inconsistency that troubled file systems, and it makes the data much easier to control, search and keep accurate. A database, however, does not work on its own; it is created and controlled by a piece of software called a database management system, or DBMS. The DBMS sits between the users, or their application programs, and the actual stored data, so that every request to read or change the data passes through it; well-known examples of DBMS software are Microsoft Access, Oracle and MySQL. The DBMS carries out a number of important functions. It allows the user to define the structure of the database, that is, the tables and the fields they contain. It allows data to be inserted, updated and deleted. It allows users to retrieve exactly the data they want by asking questions called queries. It enforces security by controlling who is allowed to see or change the data, and it protects the integrity, or accuracy, of the data. It also allows many users to work with the data at the same time without their actions conflicting, and it can make backup copies so that the data can be recovered if it is lost. Through these functions the DBMS makes a shared database safe, reliable and easy to use.

Q3: Describe the advantages of the database approach and give examples of where databases are used.

The database approach, in which data is stored in a single shared database managed by a DBMS, has many advantages over the older traditional file system, and these advantages explain why it is now used almost everywhere. The first advantage is a great reduction in data redundancy: because each piece of data is stored once and shared, the wasteful duplication of the file system is largely removed, saving storage and effort. This in turn brings improved data consistency and accuracy, because there are no longer several copies of the same data that might disagree. A database also makes data much easier to search and to update, since the DBMS provides queries and tools to find and change data quickly. It allows data to be shared safely among many users and programs at the same time, while the DBMS controls exactly who may see or alter each part of the data, giving strong security; the DBMS can also keep backup copies so that data is not lost. Because of these benefits, databases managed by a DBMS are used to hold the important records of almost every large organisation. A bank uses a database to keep the accounts and transactions of its customers; a school keeps the records, marks and results of its students; a hospital keeps

the records of its patients; an airline keeps its flight and seat bookings; a shop or company keeps its stock, sales and payroll; and governments keep records such as identity, tax and land data. In every one of these cases the database, controlled by its DBMS, keeps the large amount of related data organised, accurate, secure and available to the people who need it.

MCQs with Answers

1. Raw, unorganised facts are called:
(a) information (b) data (c) reports (d) queries
Correct Answer: (b) data. Data.

2. Processed, meaningful data is called:
(a) data (b) information (c) a file (d) a byte
Correct Answer: (b) information. Information.

3. Storing the same data many times is:
(a) consistency (b) redundancy (c) integrity (d) security
Correct Answer: (b) redundancy. Data redundancy.

4. When duplicated data disagrees, it is called:
(a) redundancy (b) inconsistency (c) a query (d) a backup
Correct Answer: (b) inconsistency. Data inconsistency.

5. An organised collection of related data is a:
(a) file only (b) database (c) folder (d) query
Correct Answer: (b) database. A database.

6. The software that manages a database is a:
(a) DBMS (b) OS (c) URL (d) NIC
Correct Answer: (a) DBMS. A DBMS.

7. Microsoft Access is an example of a:
(a) browser (b) DBMS (c) spreadsheet only (d) modem
Correct Answer: (b) DBMS. A DBMS.

8. A question used to retrieve data is a:
(a) record (b) query (c) field (d) file
Correct Answer: (b) query. A query.

9. A main advantage of a database over files is:
(a) more duplication (b) less redundancy (c) less security (d) no sharing
Correct Answer: (b) less redundancy. Less redundancy (and better consistency).

10. A bank keeps customers' accounts in a:
(a) spreadsheet only (b) database (c) word processor (d) browser
Correct Answer: (b) database. A database.

Quick Revision Summary

- Data = raw facts; information = processed, meaningful data.

- File system drawbacks: redundancy, inconsistency, data dependence, poor sharing.
- Database = organised collection of related data, shared and stored once.
- DBMS = software managing the database (MS Access, Oracle, MySQL); all access goes through it.
- DBMS functions: define, insert, update, delete, query, security, integrity, backup.
- Advantages: less redundancy, better consistency, easy search, sharing, security. Notes by freebooks.pk.

Exam Tips

- Clearly distinguish data from information with an example.
- List the drawbacks of the file system (redundancy, inconsistency).
- Define a database and a DBMS separately.
- Give examples of DBMS software.
- State the main functions of a DBMS.
- Give real examples of where databases are used.

2nd Year Computer Science — Punjab Board

CHAPTER 2

Basic Concepts and Terminology of Databases

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers the Basic Concepts and Terminology of Databases from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains fields, records, tables, attributes, views, indexes, keys (primary and foreign) and the people who work with databases. These notes are prepared by freebooks.pk.

To work with databases you must know their basic vocabulary. This chapter defines the key terms that describe how data is organised in a database.

Learning Objectives

- Define field, record and table.
- Explain attributes, tuples and entities.
- Describe views and indexes.
- Define the different types of keys.
- Distinguish a primary key from a foreign key.
- Describe the users and administrators of a database.

Key Concepts

Field, Record and Table

In a database, data is stored in tables. A field (or column) is a single item of data of a particular kind, such as a name or a roll number, and it is the smallest named unit of data. A record (or row) is a complete set of related fields about one item, such as all the details of one student. A table is a collection of records of the same kind arranged in rows and columns; for example a STUDENT table holds one record for each student.

Attributes, Tuples and Entities

The same ideas are described using more formal terms in database theory. An entity is a real-world thing about which data is stored, such as a student or a book. An attribute is a property or characteristic of the entity, such as a student's name or age; attributes correspond to the fields (columns) of a table. A tuple is a single row of a table, that is, one record. So a table represents an entity, its columns are the attributes, and its rows are the tuples.

Views and Indexes

A view is a way of looking at the data in a database that shows only selected rows and columns, rather than the whole table; different users can be given different views so that they see only the data they need. An index is a structure that the database keeps to make searching faster; like the index of a book, it lets the DBMS find the required records quickly without scanning the whole table, which speeds up queries.

Keys

A key is a field (or combination of fields) used to identify records. A primary key is a field whose value is unique for every record in a table, so it identifies each record exactly; for example a roll number in a STUDENT table. A candidate key is any field that could serve as the primary key, and a composite key is a primary key made of more than one field. A foreign key is a field in one table that refers to the primary key of another table, and it is used to link the two tables together.

Primary Key and Foreign Key

The primary key and the foreign key are the most important keys. The primary key uniquely identifies each record within its own table and cannot be duplicated or left empty. The foreign key is a field placed in a second table that holds the value of the primary key of the first table, thereby creating a relationship between the two tables; for example, a RESULT table can include the RollNo (a foreign key) to link each result to a student in the STUDENT table. Keys are what make it possible to relate data across tables.

Database Users

Several kinds of people work with a database. End users are the people who use the database to enter, update and retrieve data for their daily work, such as clerks and managers. Application programmers write the programs that use the database. Above these are two special roles: the data administrator, who plans and controls the organisation's overall data policy, and the database administrator (DBA), a technical expert who actually creates, maintains, secures and tunes the database and controls who may use it.

The Database Administrator (DBA)

The database administrator (DBA) is the person responsible for the whole database system. The DBA's duties include designing and creating the database, defining its structure, controlling access by granting rights to users (security), keeping the data accurate (integrity), making regular backups and recovering the database after failure, and improving its performance. Because the DBA has full control over the database, this is a highly responsible and technical role.

Important Definitions

Term	Definition
Field	A single item of data (a column); the smallest named unit.
Record	A complete set of related fields about one item (a row).
Table	A collection of records of the same kind in rows and columns.
Attribute	A property of an entity; corresponds to a field (column).
Tuple	A single row (record) of a table.
Primary key	A field with a unique value that identifies each record.

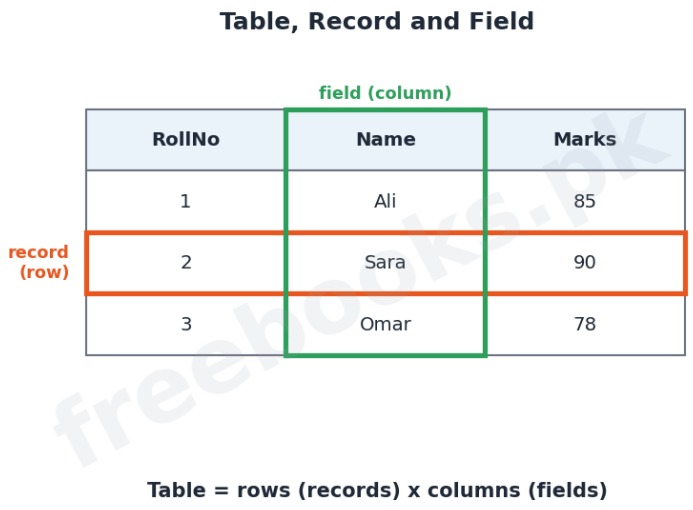
Term	Definition
Foreign key	A field that refers to the primary key of another table.
DBA	Database administrator; the person who controls the database.

Key Facts & Rules

Item	Detail
Field	Column; smallest unit of data
Record	Row; all fields about one item
Table	Rows (records) x columns (fields)
Entity / attribute / tuple	Table / column / row
Primary key	Unique, identifies each record
Foreign key	Links to primary key of another table
DBA	Creates, secures and maintains the database

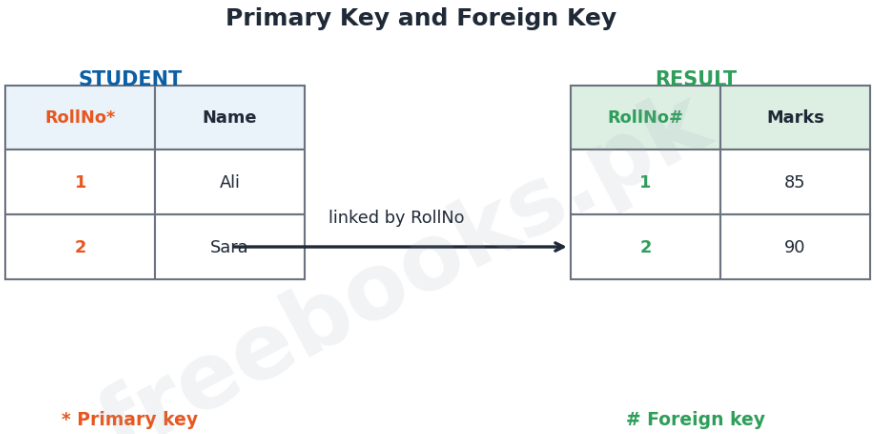
Diagrams & Illustrations

Table, record and field: a database table with a record (row) and a field (column) highlighted.



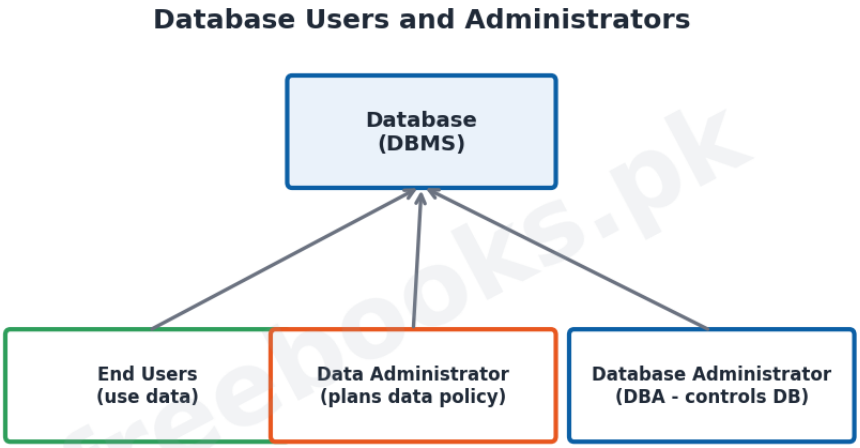
Table, record and field

Primary and foreign key: two tables (STUDENT and RESULT) linked by a primary key and a foreign key.



Primary and foreign key

Database users: the end users, data administrator and database administrator (DBA) who work with a database.



Database users

Solved Examples & Practice

Identify field and record

In a STUDENT table, 'Name' is a field and all the details of one student make up a record.

Choose a primary key

In a STUDENT table, RollNo is a good primary key because it is unique for each student.

Use a foreign key

Putting RollNo into a RESULT table lets each result be linked to a student; there RollNo is a foreign key.

Faster searching

Adding an index on the Name field lets the DBMS find students by name more quickly.

Short Questions & Answers

Q1. Differentiate a field and a record.
Ans. A field is a single item of data (a column); a record is a complete set of related fields about one item (a row).

Q2. What is a table?
Ans. A collection of records of the same kind, arranged in rows and columns.

Q3. Define an attribute and a tuple.
Ans. An attribute is a property of an entity (a column); a tuple is a single row (record) of the table.

Q4. What is a primary key?
Ans. A field whose value is unique for every record, so it identifies each record exactly.

Q5. What is a foreign key?
Ans. A field in one table that refers to the primary key of another table, linking the two tables.

Q6. Who is the DBA?
Ans. The database administrator, the person who creates, controls, secures and maintains the database.

Long Questions & Answers

Q1: Explain the terms field, record, table, attribute, tuple and entity with an example.

Databases have their own vocabulary for describing how data is arranged, and understanding these terms is essential. Data in a database is held in tables. The smallest named unit of data is a field, also called a column, which holds a single item of a particular kind, such as a name, a roll number or a mark. A record, also called a row, is a complete set of related fields describing one particular item; for example, in a table of students, one record contains all the fields, roll number, name, marks and so on, belonging to a single student. A table is then a collection of many records of the same kind, arranged neatly in rows and columns, so that a STUDENT table has one row for each student and one column for each piece of information kept about them. The same ideas are expressed by more formal terms in database theory. An entity is a real-world object or thing about which data is stored, such as a student, a book or an employee; a table usually represents one entity. An

attribute is a property or characteristic of that entity, such as the student's name or age, and the attributes correspond to the columns (fields) of the table. A tuple is a single row of the table, that is, one record. So, taking a STUDENT table as an example, the entity is the student, the attributes are the columns such as RollNo, Name and Marks, and each tuple (row) is the record of one student. These terms allow us to describe the structure of any database precisely.

Q2: Explain the different types of keys used in a database, especially the primary key and the foreign key.

A key is a field, or a combination of fields, that is used to identify records in a table and to relate tables to one another, and several kinds of key are used in databases. The most important is the primary key, which is a field whose value is different, or unique, for every record in the table, so that it identifies each record exactly and no two records can share the same value; a primary key value must always be present and cannot be left empty. For example, in a STUDENT table the roll number can serve as the primary key because each student has a different roll number. A candidate key is any field that has the property of being unique and could therefore be chosen as the primary key; from among the candidate keys, one is selected to be the primary key. A composite key is a primary key that is made up of two or more fields together, used when no single field is unique on its own. The other very important key is the foreign key, which is a field placed in one table that holds the value of the primary key of another table, and its purpose is to create a link, or relationship, between the two tables. For instance, a RESULT table might contain the field RollNo; this RollNo is the primary key in the STUDENT table but appears in the RESULT table as a foreign key, and it connects each result to the correct student. In this way, primary keys uniquely identify records within a table, while foreign keys join related tables together, which is the basis of the relational database.

Q3: Describe the different people who work with a database, including the database administrator (DBA).

A database is used and looked after by several different kinds of people, each with their own role. At the most basic level are the end users, the ordinary people who use the database in their daily work to enter new data, update existing data and retrieve information; examples include the clerks who record transactions and the managers who look at reports. Then there are the application programmers, who write the computer programs through which many end users work with the database. Above these ordinary users are two special and responsible roles concerned with managing the data itself. The first is the data administrator, who is chiefly concerned with policy: this person decides what data the organisation should keep, who should be allowed to use it and how it should be organised, planning and controlling the organisation's overall data resource. The second, and the most technical, is the database administrator, usually shortened to DBA, who is responsible for the database system itself. The DBA designs and creates the database and defines its structure; controls access to it by granting the appropriate rights to different users, so keeping the data secure; maintains the accuracy, or integrity, of the data; makes regular backup copies and recovers the database if it is damaged or lost; and monitors and improves its performance. Because

the DBA has complete control over the database and its security, it is a highly skilled and trusted position. Together, these people ensure that the database is well designed, correctly used, kept safe and made available to everyone who needs it.

MCQs with Answers

1. A single column of a table is a:
(a) record (b) field (c) tuple (d) key
Correct Answer: (b) field. A field.

2. A single row of a table is a:
(a) field (b) record (c) table (d) index
Correct Answer: (b) record. A record (row).

3. A property of an entity is an:
(a) attribute (b) index (c) key (d) view
Correct Answer: (a) attribute. An attribute.

4. A single row is also called a:
(a) tuple (b) field (c) view (d) key
Correct Answer: (a) tuple. A tuple.

5. A field with a unique value for each record is the:
(a) foreign key (b) primary key (c) index (d) view
Correct Answer: (b) primary key. The primary key.

6. A field linking to another table's primary key is a:
(a) primary key (b) foreign key (c) candidate key (d) index
Correct Answer: (b) foreign key. A foreign key.

7. A structure that speeds up searching is an:
(a) index (b) entity (c) tuple (d) view
Correct Answer: (a) index. An index.

8. A selected sub-set of data shown to a user is a:
(a) view (b) key (c) field (d) record
Correct Answer: (a) view. A view.

9. A primary key made of more than one field is a:
(a) foreign key (b) composite key (c) index (d) view
Correct Answer: (b) composite key. A composite key.

10. The person who controls and maintains the database is the:
(a) end user (b) DBA (c) programmer (d) operator
Correct Answer: (b) DBA. The DBA.

Quick Revision Summary

- Field = column; record = row; table = rows x columns.
- Entity = table (a thing); attribute = column; tuple = row.

- View = selected data shown to a user; index = speeds up searching.
- Primary key = unique, identifies each record; candidate/composite keys.
- Foreign key = links to another table's primary key (relates tables).
- Users: end users, programmers, data administrator, DBA (controls the database). Notes by freebooks.pk.

Exam Tips

- Define field, record and table clearly.
- Match entity/attribute/tuple to table/column/row.
- Explain the primary key (unique, identifies records).
- Explain the foreign key (links tables).
- State what views and indexes are for.
- List the DBA's main duties.

2nd Year Computer Science — Punjab Board

CHAPTER 3

Database Design Process

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers the Database Design Process from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains data modeling, entities and relationships, cardinality and modality, the entity-relationship diagram, normalization and physical database design. These notes are prepared by freebooks.pk.

A good database must be carefully designed before it is built. This chapter explains the steps of designing a database so that it stores data correctly and efficiently.

Learning Objectives

- Explain the need for database design.
- Describe data modeling, entities and relationships.
- Explain cardinality and modality.
- Draw and read an entity-relationship (ER) diagram.
- Explain the purpose of normalization.
- Describe physical database design and implementation.

Key Concepts

The Need for Database Design

Database design is the process of planning the structure of a database before it is created, so that it stores all the required data without unnecessary duplication and can be used efficiently. A poorly designed database wastes space, allows errors and is hard to change, while a well-designed one is accurate, efficient and easy to maintain. Design is carried out in a series of steps, beginning with understanding the users' requirements.

Data Modeling, Entities and Relationships

Data modeling means representing the data of the real world in a form that can be stored in a database. The main building blocks are entities and relationships. An entity is a real-world object about which data is kept, such as a student or a course, and it becomes a table. A relationship is an association between entities, such as a student 'takes' a course. Identifying the entities, their attributes and the relationships between them is the heart of database design.

Cardinality and Modality

The relationships between entities are described by their cardinality and modality. Cardinality states how many instances of one entity may be related to instances of another; the three types are one-to-one (1:1), one-to-many (1:M) and many-to-many (M:N). For example, one class has many students, so the relationship between class and student is one-to-many. Modality states whether a relationship is optional or compulsory, that is, whether an instance must take part in the relationship or not.

The Entity-Relationship Diagram

An entity-relationship diagram (ERD) is a picture that shows the design of a database. In an ERD, entities are drawn as rectangles, the attributes may be shown as ovals, and relationships are drawn as diamonds joining the entities, with the cardinality marked on the connecting lines. The ERD gives a clear, visual overview of the whole database and is used as the plan from which the actual tables are created.

From ER Model to Tables and Normalization

Once the ER model is complete, each entity is turned into a table with a primary key, and the relationships are represented by placing foreign keys or by creating extra tables. The tables are then improved by a process called normalization, which reorganises the data to remove redundancy (duplicated data) and to avoid errors when data is inserted, updated or deleted. Normalization splits large, badly organised tables into smaller, well-structured ones linked by keys.

Physical Database Design

After the logical design (the tables and keys) is fixed, the physical database design decides how the data will actually be stored on the computer for good performance. This involves estimating the volume of data and how it will be used, deciding how the data will be distributed and how the files will be organised, choosing which fields to index for fast searching, and defining integrity constraints, which are rules that keep the data correct (for example, that a mark must be between 0 and 100).

Implementation

Implementation is the final stage, in which the completed design is turned into a working database using a DBMS such as Microsoft Access. The tables are created with their fields, data types and keys; the relationships, indexes and integrity constraints are set up; and the database is then filled with data and tested. Once it works correctly the database is put into use, and later it is maintained and improved as needs change.

Important Definitions

Term	Definition
Database design	Planning the structure of a database before creating it.
Data modeling	Representing real-world data in a form suitable for a database.
Entity	A real-world object about which data is stored (becomes a table).
Relationship	An association between entities.
Cardinality	How many instances of one entity relate to another (1:1, 1:M, M:N).
ER diagram	A diagram showing entities and their relationships.
Normalization	Reorganising tables to remove redundancy and avoid errors.

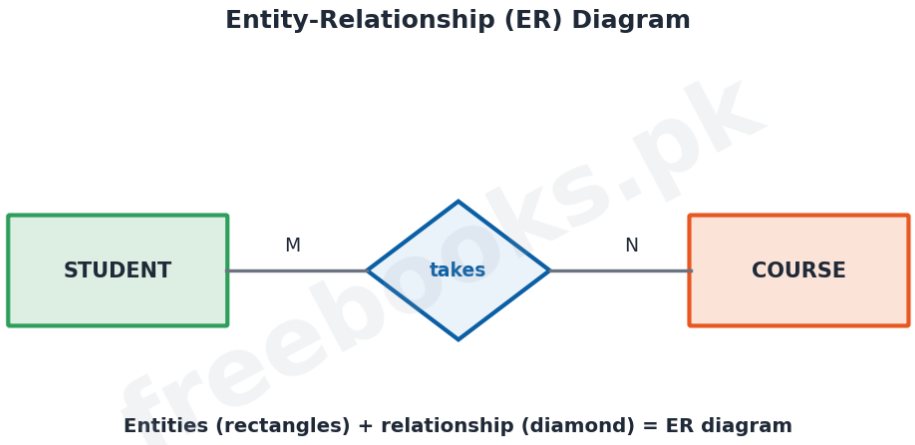
Term	Definition
Integrity constraint	A rule that keeps the data in the database correct.

Key Facts & Rules

Item	Detail
Entity	A thing -> a table
Relationship	Association between entities (a diamond in an ERD)
Cardinality types	1:1, 1:M, M:N
ERD symbols	Rectangle = entity, oval = attribute, diamond = relationship
Normalization	Removes redundancy; splits into linked tables
Physical design	Storage, file organisation, indexes, integrity constraints

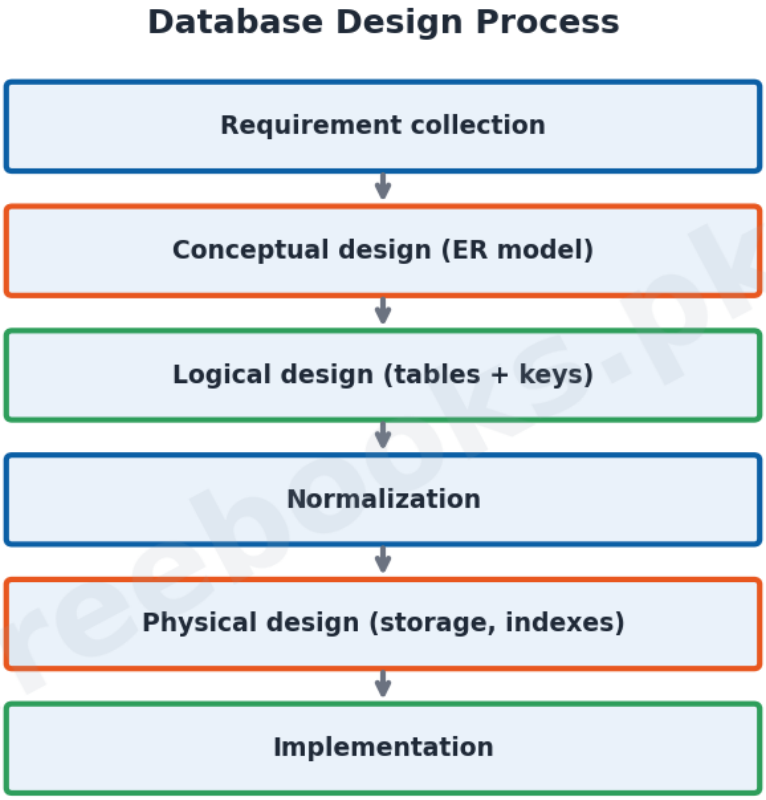
Diagrams & Illustrations

Entity-relationship diagram: an ER diagram with two entities (STUDENT and COURSE) joined by a relationship (takes), with cardinality marked.



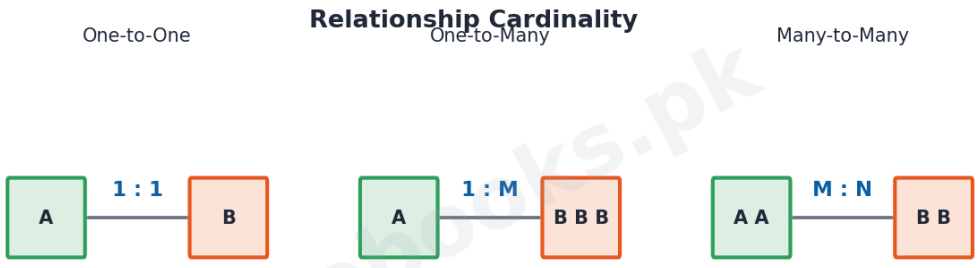
Entity-relationship diagram

Database design process: the steps of database design from requirement collection through conceptual, logical, physical design and implementation.



Database design process

Relationship cardinality: the three kinds of cardinality: one-to-one, one-to-many and many-to-many.



Relationship cardinality

Solved Examples & Practice

Identify entities

In a school database, STUDENT, TEACHER and CLASS are entities; each becomes a table.

State the cardinality

One class contains many students, so the CLASS to STUDENT relationship is one-to-many (1:M).

Read an ERD

In an ERD, a rectangle stands for an entity and a diamond for a relationship between entities.

Why normalize

If a customer's address is repeated in every order, normalization moves it to a separate CUSTOMER table to remove the duplication.

Short Questions & Answers

Q1. What is database design?
Ans. The process of planning the structure of a database before it is created, so it stores data correctly and efficiently.

Q2. What is an entity? Give an example.
Ans. A real-world object about which data is stored, such as a student; it becomes a table.

Q3. What is cardinality? Name its types.
Ans. How many instances of one entity relate to another; the types are one-to-one, one-to-many and many-to-many.

Q4. What is an ER diagram?
Ans. A diagram that shows the entities of a database and the relationships between them.

Q5. What is normalization?
Ans. The process of reorganising tables to remove redundancy and avoid errors during insert, update and delete.

Q6. What is an integrity constraint?
Ans. A rule that keeps the data correct, for example that a mark must be between 0 and 100.

Long Questions & Answers

Q1: Explain data modeling, entities, relationships and cardinality in database design.

Before a database is built it must be carefully designed, and the first and most important part of this design is data modeling, which means representing the data of the real world in a form that can be stored in a database. The two main building blocks of a data model are entities and relationships. An entity is a real-world object or thing about which the organisation needs to keep data, such as a student, a teacher, a course or a book; each

entity usually becomes a table in the database, and the properties we store about it, such as a student's roll number and name, are its attributes. A relationship is an association or connection between entities that reflects how they are related in the real world; for example, a student takes a course, or a teacher teaches a class, so 'takes' and 'teaches' are relationships. When two entities are related, we describe the relationship by its cardinality, which states how many instances of one entity can be linked to instances of the other. There are three kinds of cardinality. In a one-to-one (1:1) relationship, each instance of one entity is related to just one instance of the other. In a one-to-many (1:M) relationship, one instance of the first entity is related to many instances of the second, as when one class has many students. In a many-to-many (M:N) relationship, many instances of one entity are related to many of the other, as when many students take many courses. A related idea, modality, states whether taking part in a relationship is optional or compulsory. Correctly identifying the entities, their attributes and the cardinality of their relationships is the foundation of a good database design.

Q2: Describe the entity-relationship diagram and the process of turning a design into normalized tables.

Once the entities and relationships of a database have been worked out, they are drawn as an entity-relationship diagram, or ERD, which is a clear picture of the whole design. In an ERD, each entity is shown as a rectangle labelled with its name, the attributes may be drawn as ovals attached to the entity, and each relationship is shown as a diamond connecting the entities it joins; the cardinality of the relationship (such as 1, M or N) is written on the lines that link the diamond to the entities. Because it presents the entire structure visually, the ERD serves as the plan from which the actual database is built. The next step is to convert this ER model into tables: each entity becomes a table with a suitable primary key, and the relationships between entities are represented either by adding a foreign key to one of the tables or, for many-to-many relationships, by creating an additional linking table. These tables are then refined by a process called normalization, whose purpose is to organise the data so that each fact is stored in only one place. Normalization removes redundancy, that is, the unnecessary repetition of the same data, and in doing so it prevents the update, insertion and deletion problems that arise when duplicated data gets out of step. It works by splitting large, poorly organised tables into smaller, well-structured tables that are linked together by keys. The result is a set of clean, non-redundant tables that store the data efficiently and reliably.

Q3: Describe physical database design and the implementation of a database.

After the logical design of a database, that is, the set of normalized tables with their keys and relationships, has been decided, two further stages complete the process: physical design and implementation. Physical database design is concerned with how the data will actually be stored inside the computer so that the database performs well. In this stage the designer estimates the volume of data that the database will hold and studies how it will be used, that is, which data will be read and written most often; decides how the data should be distributed and how the files that hold it should be organised on the storage devices;

chooses which fields should be indexed, since an index greatly speeds up searching on those fields; and defines the integrity constraints, which are rules built into the database to keep the data correct, such as insisting that a mark lies between 0 and 100 or that a roll number is never left blank. Good physical design makes the difference between a database that responds quickly and one that is slow. The final stage is implementation, in which the finished design is actually built using a database management system such as Microsoft Access. Here the tables are created with their fields, data types and primary keys; the relationships between tables, the indexes and the integrity constraints are set up; and the database is then loaded with data and thoroughly tested to make sure it works correctly. Once testing is successful, the database is put into everyday use, after which it is maintained and, from time to time, improved as the needs of its users change.

MCQs with Answers

1. Planning a database before building it is:
(a) implementation (b) database design (c) normalization (d) indexing
Correct Answer: (b) database design. Database design.

2. A real-world object stored in a database is an:
(a) attribute (b) entity (c) index (d) key
Correct Answer: (b) entity. An entity.

3. An association between entities is a:
(a) relationship (b) tuple (c) field (d) view
Correct Answer: (a) relationship. A relationship.

4. One class with many students is a ___ relationship:
(a) one-to-one (b) one-to-many (c) many-to-many (d) none
Correct Answer: (b) one-to-many. One-to-many (1:M).

5. In an ERD an entity is drawn as a:
(a) diamond (b) rectangle (c) oval (d) circle
Correct Answer: (b) rectangle. A rectangle.

6. In an ERD a relationship is drawn as a:
(a) rectangle (b) diamond (c) oval (d) line
Correct Answer: (b) diamond. A diamond.

7. Removing redundancy from tables is called:
(a) indexing (b) normalization (c) modeling (d) backup
Correct Answer: (b) normalization. Normalization.

8. A rule that keeps data correct is an:
(a) index (b) integrity constraint (c) entity (d) attribute
Correct Answer: (b) integrity constraint. An integrity constraint.

9. Many students taking many courses is:
(a) 1:1 (b) 1:M (c) M:N (d) none
Correct Answer: (c) M:N. Many-to-many (M:N).

10. Turning the design into a working database is:

(a) design (b) implementation (c) modeling (d) cardinality

Correct Answer: (b) implementation. Implementation.

Quick Revision Summary

- Database design = planning the structure before building; done in steps.
- Data modeling: entities (things -> tables) + relationships (associations).
- Cardinality: 1:1, 1:M, M:N; modality = optional/compulsory.
- ERD: rectangle = entity, oval = attribute, diamond = relationship (with cardinality).
- Normalization removes redundancy; splits into linked tables.
- Physical design: storage, indexes, integrity constraints; then implementation in a DBMS. Notes by freebooks.pk.

Exam Tips

- Define entity and relationship with examples.
- Give the three cardinality types with examples.
- Know the ERD symbols (rectangle/oval/diamond).
- State the purpose of normalization (remove redundancy).
- Explain what physical design decides (indexes, constraints).
- Order the design steps ending in implementation.

2nd Year Computer Science — Punjab Board

CHAPTER 4

Data Integrity and Normalization

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers Data Integrity and Normalization from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains data integrity and its types, data anomalies, functional dependency and the process of normalization up to the third normal form. These notes are prepared by freebooks.pk.

For a database to be trustworthy, its data must be kept correct and free of harmful duplication. This chapter explains integrity rules and normalization, the techniques that ensure this.

Learning Objectives

- Define data integrity and its types.
- Explain entity and referential integrity.
- Describe the anomalies caused by redundancy.
- Explain functional dependency.
- Describe the purpose and process of normalization.
- State the first, second and third normal forms.

Key Concepts

Data Integrity

Data integrity means the correctness, accuracy and consistency of the data stored in a database. A database has integrity when its data obeys the rules that have been defined for it, so that it contains no wrong, missing or contradictory values. Integrity is enforced by integrity constraints, that is, rules built into the database, and it is essential because decisions are made using the data and incorrect data leads to wrong results.

Entity and Referential Integrity

Two important kinds of integrity are entity integrity and referential integrity. Entity integrity is the rule that the primary key of every table must have a value (it cannot be null) and that value must be unique, so that every record can be identified. Referential integrity is the rule that a foreign key value in one table must match an existing primary key value in the related table (or be null); this prevents a record from referring to another record that does not exist, keeping the links between tables valid.

Redundancy and Anomalies

When the same data is stored many times (redundancy), the database can suffer from anomalies, that is, problems that occur when data is changed. An insertion anomaly means some data cannot be added without adding other, unrelated data. An update anomaly means that changing one fact requires changing many copies, and if some are missed the data becomes inconsistent. A deletion anomaly means that deleting one record accidentally loses other useful data. Normalization is used to remove these anomalies.

Functional Dependency

A functional dependency describes how one field depends on another. If, for a given value of field A, there is always exactly one value of field B, then B is said to be functionally dependent on A, written A -> B. For example, a roll number determines a student's name, so Name is functionally dependent on RollNo. Understanding functional dependencies helps decide how to split tables correctly during normalization.

Normalization

Normalization is the process of organising the fields and tables of a database to reduce redundancy and remove anomalies. It is carried out in a series of steps called normal forms, in each of which the tables are refined a little further by splitting them into smaller, well-structured tables linked by keys. The aim is that each piece of data is stored in only one place, so the database becomes accurate, consistent and easy to maintain.

First and Second Normal Forms

A table is in first normal form (1NF) when it contains no repeating groups and every field holds only a single (atomic) value, so that the data is arranged as a simple table of rows and columns. A table is in second normal form (2NF) when it is already in 1NF and, in addition, every non-key field depends on the whole primary key and not on just part of it; this removes partial dependencies, which arise only when the primary key is composite (made of more than one field).

Third Normal Form

A table is in third normal form (3NF) when it is already in 2NF and, in addition, no non-key field depends on another non-key field; that is, there are no transitive dependencies. In practice this means that any field that really belongs to a different entity is moved out into its own table. A database whose tables are all in third normal form has very little redundancy and is generally free of the update, insertion and deletion anomalies, so 3NF is the usual goal of normalization.

Important Definitions

Term	Definition
Data integrity	The correctness, accuracy and consistency of stored data.
Entity integrity	The rule that a primary key must be unique and not null.
Referential integrity	The rule that a foreign key must match an existing primary key or be null.
Anomaly	A problem (insert, update or delete) caused by redundancy.
Functional dependency	When one field's value determines another (A -> B).
Normalization	Organising tables to remove redundancy and anomalies.
1NF	No repeating groups; every field holds a single value.

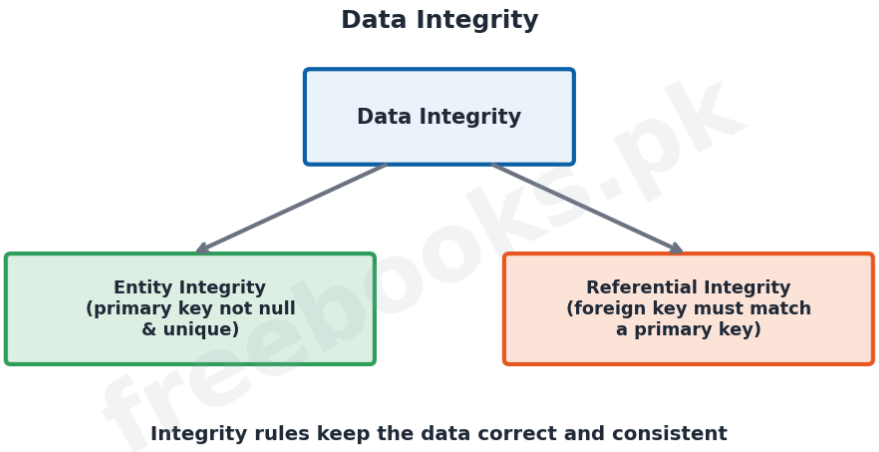
Term	Definition
3NF	In 2NF with no non-key field depending on another non-key field.

Key Facts & Rules

Item	Detail
Data integrity	Data is correct, accurate and consistent
Entity integrity	Primary key: unique and not null
Referential integrity	Foreign key matches a primary key (or null)
Anomalies	Insertion, update, deletion
Functional dependency	A -> B (A determines B)
Normal forms	1NF -> 2NF -> 3NF

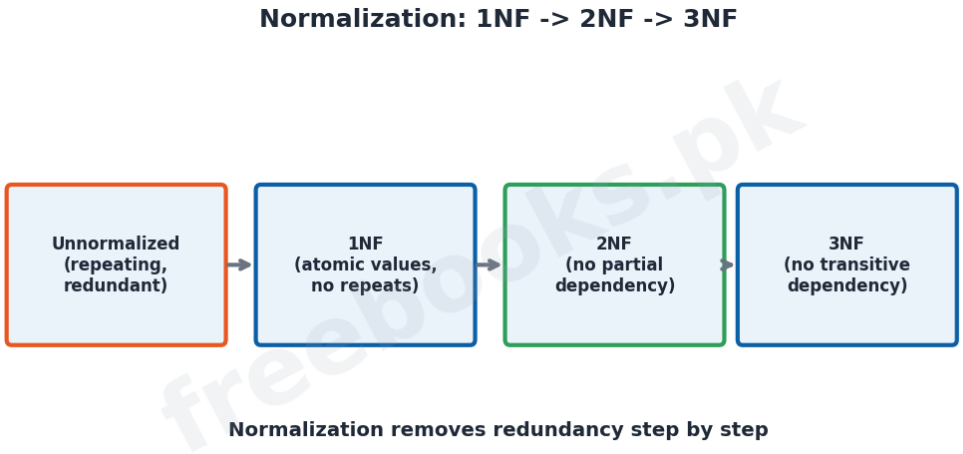
Diagrams & Illustrations

Data integrity: the two main kinds of data integrity: entity integrity (unique, non-null primary key) and referential integrity (valid foreign key).



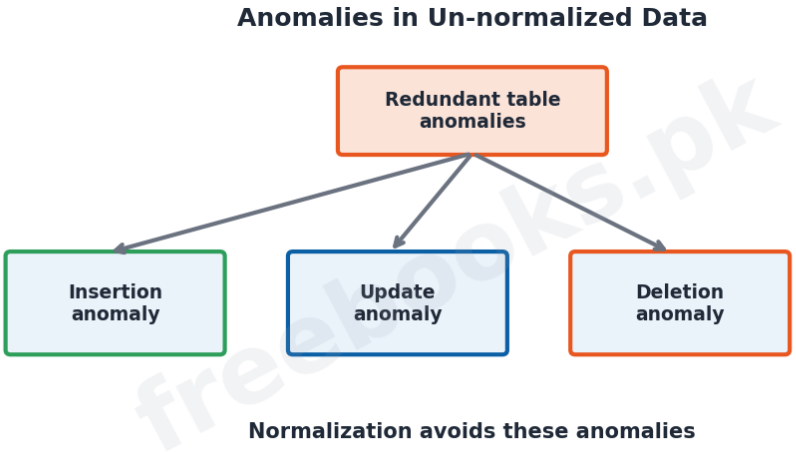
Data integrity

Normalization stages: the stages of normalization from unnormalized data through first, second and third normal forms.



Normalization stages

Data anomalies: the insertion, update and deletion anomalies caused by redundant data, which normalization removes.



Data anomalies

Solved Examples & Practice

Entity integrity

A STUDENT table must not have two students with the same RollNo, and RollNo cannot be blank; this is entity integrity.

Referential integrity

A RESULT record with RollNo = 99 is not allowed if no student with RollNo 99 exists; this protects referential integrity.

Spot the anomaly

If a teacher's phone number is stored in every one of their class records, changing it means editing many rows; this is an update anomaly.

Functional dependency

In a STUDENT table, RollNo -> Name means the roll number determines the name.

Short Questions & Answers

Q1. What is data integrity?
Ans. The correctness, accuracy and consistency of the data stored in a database.

Q2. What is entity integrity?
Ans. The rule that a table's primary key must be unique and must not be null, so each record can be identified.

Q3. What is referential integrity?
Ans. The rule that a foreign key must match an existing primary key value in the related table (or be null).

Q4. Name the three data anomalies.
Ans. Insertion anomaly, update anomaly and deletion anomaly.

Q5. What is a functional dependency?
Ans. A relationship in which the value of one field determines the value of another, written A -> B.

Q6. What is normalization?
Ans. The process of organising tables to remove redundancy and the anomalies it causes.

Long Questions & Answers

Q1: Explain data integrity and describe entity integrity and referential integrity.

Data integrity means the correctness, accuracy and consistency of the data held in a database; a database is said to have integrity when all of its data obeys the rules laid down for it and contains no wrong, missing or contradictory values. Because organisations make important decisions using the data in their databases, and incorrect data would lead to wrong decisions, maintaining integrity is essential, and it is enforced by integrity constraints, which are rules built into the database that the DBMS checks whenever data is entered or changed. Two of the most important kinds of integrity are entity integrity and referential integrity. Entity integrity is concerned with the primary key of a table: the rule states that the primary key of every record must always have a value, that is, it can never be left null, and that this value must be unique, so that no two records share it. This guarantees that every record in the table can be uniquely identified; for example, in a STUDENT table no two students may have the same roll number and no student may be entered without one.

Referential integrity is concerned with the links between tables that are made by foreign keys: the rule states that the value of a foreign key in one table must either match an existing primary key value in the related table or be null. This prevents a record from referring to another record that does not exist; for instance, a RESULT record cannot contain a roll number for which there is no matching student in the STUDENT table. Together, entity integrity keeps each table internally correct, while referential integrity keeps the connections between tables valid, so that the whole database remains trustworthy.

Q2: Explain the anomalies caused by data redundancy and how normalization removes them.

When the same piece of data is stored in more than one place in a database, that is, when there is data redundancy, the database becomes prone to a group of problems called anomalies, which appear when data is inserted, changed or deleted. There are three kinds. An insertion anomaly occurs when it is impossible to add a piece of data without also adding other, unrelated data; for example, if the details of a course could only be stored as part of a student's record, then a new course could not be entered until at least one student had enrolled in it. An update anomaly occurs when a single fact is stored in many rows, so that changing it correctly requires changing every one of those rows; if even one copy is missed, the copies disagree and the data becomes inconsistent, for instance when a teacher's telephone number is repeated in every class record and only some are updated. A deletion anomaly occurs when deleting a record accidentally removes other useful information as well; for example, deleting the last student of a course might also delete the only stored details of that course. These anomalies all arise from redundancy, and the technique used to remove them is normalization. Normalization reorganises the fields and tables so that each fact is stored in only one place, by splitting large, redundant tables into smaller, well-structured tables that are linked together by keys. Once a database has been properly normalized, each item of data appears once, so it can be inserted, updated or deleted without any of these anomalies occurring, and the data stays consistent and correct.

Q3: Describe the process of normalization and explain the first, second and third normal forms.

Normalization is the step-by-step process of organising the tables of a database so as to remove redundancy and the anomalies that go with it, and it is carried out through a series of stages called normal forms, each of which imposes a stricter rule than the one before and refines the tables a little further. The first stage produces the first normal form (1NF). A table is in 1NF when it contains no repeating groups and every field holds only a single, indivisible (atomic) value, so that the data is arranged as a plain table of rows and columns with no lists or repeated columns; this gives the data a regular, tabular shape. The second stage produces the second normal form (2NF). A table is in 2NF when it is already in 1NF and, in addition, every non-key field depends on the whole of the primary key and not on just a part of it. This rule is only relevant when the primary key is a composite key, made up of more than one field, and it removes so-called partial dependencies by moving any field that depends on only part of the key into a separate table. The third stage produces the third normal form (3NF). A table is in 3NF when it is already in 2NF and, in addition, no non-key

field depends on another non-key field; such an indirect dependence is called a transitive dependency, and removing it means taking any field that really describes a different entity and placing it in its own table. When all the tables of a database are in third normal form, each piece of data is stored only once, redundancy is almost eliminated, and the update, insertion and deletion anomalies no longer occur, which is why reaching third normal form is the usual goal of normalization.

MCQs with Answers

1. The correctness and consistency of data is called:
(a) redundancy (b) data integrity (c) a query (d) an index
Correct Answer: (b) data integrity. Data integrity.

2. A primary key must be unique and:
(a) null (b) not null (c) repeated (d) empty
Correct Answer: (b) not null. Not null (entity integrity).

3. A foreign key must match a ___ or be null:
(a) field (b) primary key (c) view (d) index
Correct Answer: (b) primary key. A primary key (referential integrity).

4. A problem caused by redundancy when changing data is an:
(a) entity (b) anomaly (c) index (d) attribute
Correct Answer: (b) anomaly. An anomaly.

5. A -> B means A ___ B:
(a) equals (b) determines (c) deletes (d) copies
Correct Answer: (b) determines. Determines (functional dependency).

6. Removing redundancy by reorganising tables is:
(a) indexing (b) normalization (c) hashing (d) backup
Correct Answer: (b) normalization. Normalization.

7. Every field holding a single value gives:
(a) 1NF (b) 2NF (c) 3NF (d) 0NF
Correct Answer: (a) 1NF. First normal form (1NF).

8. Removing partial dependency gives:
(a) 1NF (b) 2NF (c) 3NF (d) BCNF
Correct Answer: (b) 2NF. Second normal form (2NF).

9. Removing transitive dependency gives:
(a) 1NF (b) 2NF (c) 3NF (d) 0NF
Correct Answer: (c) 3NF. Third normal form (3NF).

10. The usual goal of normalization is:
(a) 0NF (b) 1NF (c) 2NF (d) 3NF
Correct Answer: (d) 3NF. 3NF.

Quick Revision Summary

- Data integrity = correct, accurate, consistent data (enforced by constraints).
- Entity integrity: primary key unique + not null. Referential integrity: foreign key matches a primary key.
- Redundancy causes anomalies: insertion, update, deletion.
- Functional dependency A -> B: A determines B.
- Normalization removes redundancy by splitting tables (linked by keys).
- 1NF (atomic values), 2NF (no partial dependency), 3NF (no transitive dependency) = goal. Notes by freebooks.pk.

Exam Tips

- Define data integrity and its two main types.
- Explain entity vs referential integrity clearly.
- Name and explain the three anomalies.
- State a functional dependency (A -> B) with an example.
- Give the rule for each of 1NF, 2NF and 3NF.
- State that 3NF is the usual goal.

2nd Year Computer Science — Punjab Board

CHAPTER 5

Introduction to Microsoft Access

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers the Introduction to Microsoft Access from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains what MS Access is, how to start and exit it, how to create a database, the Access window and the main database objects. These notes are prepared by freebooks.pk.

Microsoft Access is a popular database management system for building small and medium databases. This chapter introduces Access and its main parts so that a student can begin to create databases.

Learning Objectives

- State what Microsoft Access is.
- Describe how to start and exit Access.
- Explain how to create a database with and without the wizard.
- Identify the parts of the Access application window.
- List and describe the main database objects.
- State the uses of tables, queries, forms and reports.

Key Concepts

What is Microsoft Access?

Microsoft Access is a database management system (DBMS) produced by Microsoft, used to create and manage relational databases on a personal computer. It lets a user store data in tables, retrieve it with queries, enter and view it through forms, and print it as reports, all through an easy graphical interface. Access is well suited to small and medium databases for schools, small businesses and personal use.

Starting and Exiting Access

Microsoft Access is started like any other Windows program, for example from the Start menu or by clicking its icon. When it opens, the user can create a new database or open an existing one. To exit Access, the user closes the database and then closes the program from the File menu or the close button; Access saves the design of any objects and prompts the user to save changes so that no work is lost.

Creating a Database

A new database in Access can be created in two ways. Using the Database Wizard, the user chooses a ready-made template (such as contacts or inventory) and Access automatically builds the tables, forms and reports, which is quick for common tasks. Without the wizard, the user creates a blank database and then designs each table, query, form and report themselves, which gives full control. In both cases the database is given a name and saved as a file before work continues.

The Access Application Window

The Microsoft Access window has several parts. The title bar shows the program and database name. The menu bar or ribbon contains the commands, grouped under tabs such as Home and Create. The Navigation Pane, usually on the left, lists all the objects in the database (tables, queries, forms and reports) and lets the user open them. The main work area in the centre displays whichever object is currently open, and a status bar at the bottom shows information about the current task.

Database Objects

An Access database is made up of several kinds of object, each with its own job. Tables store the actual data in rows and columns. Queries retrieve and display selected data that answers a question. Forms provide a convenient screen layout for entering and viewing data one record at a time. Reports produce neatly formatted output for printing. Access also has macros and modules for automating tasks. Together these objects make up a complete database application.

Tables and Queries

The table is the most important object, because it actually holds the data; every database must have at least one table, and each table stores the records of one entity, such as students, with a field for each attribute and a primary key. A query is an object that asks a question of the data, for example 'list all students with marks above 80'. The query selects only the records and fields that match its conditions and shows the result, without changing the stored data.

Forms and Reports

A form is an object that presents the data in an attractive on-screen layout, usually showing one record at a time, and it is used to make entering, viewing and editing data easy and less error-prone than typing directly into a table. A report is an object designed for printing; it takes data from tables or queries and arranges it neatly, often with headings, totals and page numbers, to produce professional printed output such as mark sheets or bills.

Important Definitions

Term	Definition
Microsoft Access	A DBMS by Microsoft used to create and manage databases on a PC.
Database Wizard	A tool that builds a database automatically from a template.
Navigation Pane	The pane listing all the objects in an Access database.
Table	The object that stores data in rows and columns.
Query	The object that retrieves selected data answering a question.
Form	The object providing a screen layout to enter and view data.
Report	The object that produces formatted output for printing.

Term	Definition
Database object	A component of an Access database (table, query, form, report).

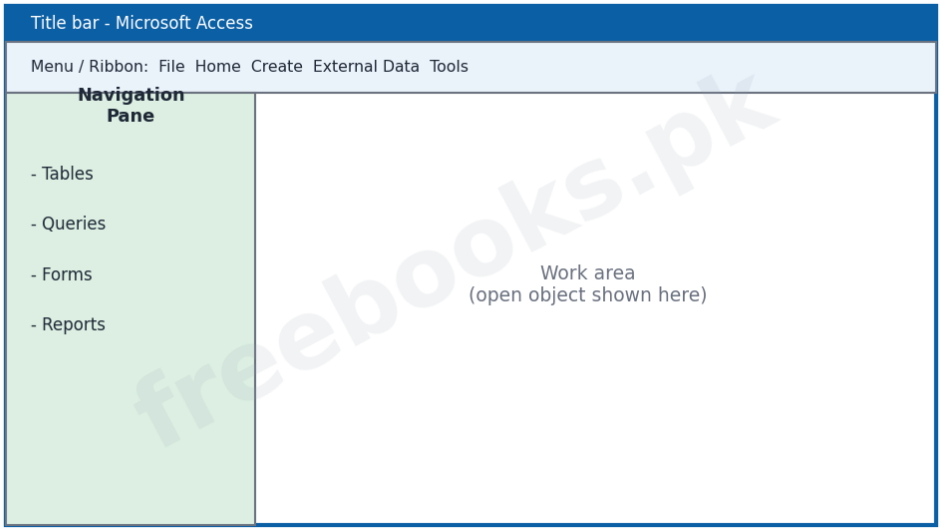
Key Facts & Rules

Item	Detail
MS Access	A DBMS for relational databases on a PC
Create a database	Blank database or Database Wizard (template)
Window parts	Title bar, ribbon, Navigation Pane, work area
Table	Stores data
Query	Retrieves data
Form	Enters/views data
Report	Prints output

Diagrams & Illustrations

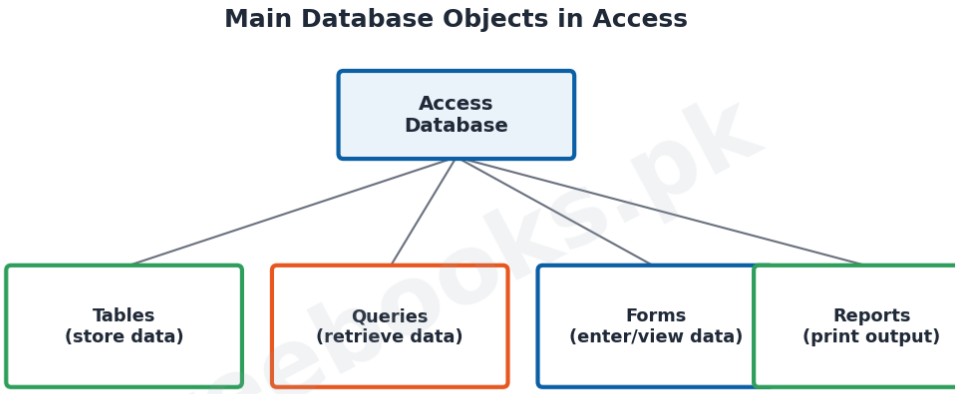
Access application window: the Microsoft Access window with its title bar, ribbon, Navigation Pane listing objects, and the work area.

Microsoft Access Application Window



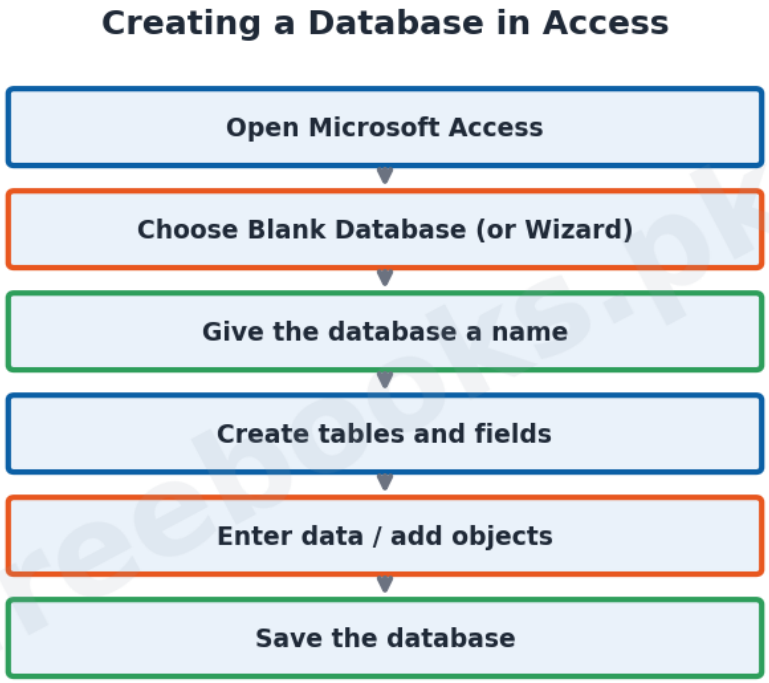
Access application window

Database objects: the main objects of an Access database: tables, queries, forms and reports.



Database objects

Creating a database: the steps to create a database in Access, from opening it to saving the finished database.



Creating a database

Solved Examples & Practice

Choose the object

To type in the details of a new student one record at a time, use a form; to store all students' data, use a table.

Use a query

To list only the students who failed, create a query with the condition Marks < 40.

Create quickly

To build a ready-made contacts database quickly, use the Database Wizard with a contacts template.

Print output

To print a neat mark sheet, design a report based on the students table or a query.

Short Questions & Answers

Q1. What is Microsoft Access?
Ans. A database management system (DBMS) by Microsoft used to create and manage databases on a personal computer.

Q2. Name two ways to create a database in Access.
Ans. Using the Database Wizard (a template) or creating a blank database and designing it yourself.

Q3. What is the Navigation Pane?
Ans. The pane, usually on the left, that lists all the objects (tables, queries, forms, reports) of the database.

Q4. What is the function of a table?
Ans. To store the actual data of the database in rows and columns.

Q5. What is a query used for?
Ans. To retrieve and display selected data that answers a question, without changing the stored data.

Q6. Differentiate a form and a report.
Ans. A form is used to enter and view data on screen; a report is used to produce formatted output for printing.

Long Questions & Answers

Q1: Describe Microsoft Access and explain how a database is created in it.

Microsoft Access is a database management system, or DBMS, produced by Microsoft and widely used to create and manage relational databases on a personal computer. It provides an easy graphical interface through which a user can store data in tables, retrieve it using queries, enter and view it through forms, and print it as reports, without needing to write complicated programs; this makes it very suitable for the small and medium-sized databases used by schools, small businesses and individuals. Access is started like any other Windows application, from the Start menu or by clicking its icon, and when it opens the user may either create a new database or open an existing one. A new database can be created in two

ways. The first is by using the Database Wizard, in which the user selects a ready-made template, such as one for contacts or for inventory, and Access automatically builds a complete set of tables, forms and reports for that purpose; this is quick and convenient for common tasks. The second way is to create a blank database, in which the user starts with an empty database and then designs each table, query, form and report personally; this takes more effort but gives the user full control over the structure of the database. In both cases, the new database must first be given a name and saved as a file on the disk before the user goes on to build or fill it. Once created, the database can be opened, edited and used whenever needed.

Q2: Describe the Microsoft Access application window and its main parts.

When Microsoft Access is opened with a database, its application window is displayed, and this window has several important parts, each with a particular purpose. Across the very top is the title bar, which shows the name of the program and of the database that is currently open. Below it is the menu bar, which in modern versions of Access takes the form of a ribbon: this is a broad strip of commands organised into tabs such as Home, Create and External Data, and clicking a tab reveals the buttons for the related tasks, such as creating a new table or running a query. Down one side, usually the left, is the Navigation Pane, which lists all the objects that make up the database, grouped by type into tables, queries, forms and reports; from this pane the user can open any object simply by double-clicking it, and it gives a clear overview of the whole database. The large central part of the window is the work area, in which whichever object the user has opened, a table, a form, a query result or a report, is actually displayed and worked on. Finally, along the bottom is the status bar, which shows helpful information about the current view or task. Understanding these parts allows the user to move about the Access window confidently and to find and use the commands and objects they need.

Q3: Describe the main objects of an Access database and state the purpose of each.

A Microsoft Access database is not a single thing but is made up of several different kinds of component called objects, each of which does a particular job, and together they form a complete database application. The most important object is the table, because it is the table that actually stores the data; every database must contain at least one table, and each table holds the records of a single entity, such as students or books, with a column (field) for each attribute and a primary key to identify each record uniquely. The next object is the query, whose purpose is to retrieve information from the tables; a query asks a question of the data, such as listing all the students whose marks are above eighty, and it displays just the records and fields that satisfy its conditions, without altering the data that is stored. Then there is the form, which provides a convenient and attractive layout on the screen, usually showing one record at a time, so that entering, viewing and editing data is easy and less likely to cause mistakes than typing directly into a table. There is also the report, which is designed for printing; a report takes data from tables or queries and arranges it neatly, often with headings, groupings, totals and page numbers, to produce professional printed output such as a mark sheet or a bill. In addition, Access provides macros and modules,

which are used to automate repeated tasks and to add extra functions. By combining these objects, tables to store, queries to find, forms to enter and view, and reports to print, a user can build a full, working database system in Access.

MCQs with Answers

1. Microsoft Access is a:
(a) word processor (b) DBMS (c) spreadsheet (d) browser
Correct Answer: (b) DBMS. A DBMS.

2. A ready-made way to build a database quickly is the:
(a) query (b) Database Wizard (c) report (d) macro
Correct Answer: (b) Database Wizard. The Database Wizard.

3. The pane listing all database objects is the:
(a) title bar (b) Navigation Pane (c) status bar (d) ribbon
Correct Answer: (b) Navigation Pane. The Navigation Pane.

4. The object that stores data is the:
(a) query (b) table (c) form (d) report
Correct Answer: (b) table. The table.

5. The object that retrieves selected data is the:
(a) table (b) query (c) form (d) macro
Correct Answer: (b) query. The query.

6. The object used to enter data one record at a time is the:
(a) report (b) form (c) table (d) query
Correct Answer: (b) form. The form.

7. The object used to print formatted output is the:
(a) form (b) report (c) query (d) table
Correct Answer: (b) report. The report.

8. Every database must have at least one:
(a) report (b) form (c) table (d) macro
Correct Answer: (c) table. Table.

9. A query with Marks < 40 will list students who:
(a) passed (b) failed (c) scored 100 (d) are absent
Correct Answer: (b) failed. Failed.

10. Access is best suited for ___ databases:
(a) only huge (b) small and medium (c) only text (d) no
Correct Answer: (b) small and medium. Small and medium databases.

Quick Revision Summary

- MS Access = DBMS for relational databases on a PC (easy graphical interface).

- Create a database: Database Wizard (template) or blank database (full control); save with a name.
- Window: title bar, ribbon (Home/Create...), Navigation Pane (lists objects), work area.
- Objects: table (store), query (retrieve), form (enter/view), report (print); plus macros/modules.
- Table has fields + primary key; query shows matching records without changing data.
- Form = on-screen entry/view; report = printed output. Notes by freebooks.pk.

Exam Tips

- State what Access is (a DBMS).
- Give the two ways to create a database.
- Label the parts of the Access window.
- List the four main objects and their jobs.
- Match table/query/form/report to store/retrieve/enter/print.
- Remember every database needs at least one table.

2nd Year Computer Science — Punjab Board

CHAPTER 6

Table and Query

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers Table and Query from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains how to create tables, field data types and properties, the primary key, and how to create and use queries with criteria and sorting. These notes are prepared by freebooks.pk.

Tables store the data and queries find it. This chapter explains how to build tables correctly and how to write queries that retrieve exactly the information needed.

Learning Objectives

- Create a table and define its fields.
- State the common data types and field properties.
- Set a primary key.
- Enter and edit data in datasheet view.
- Create a query with criteria and sorting.
- Describe the types of queries.

Key Concepts

Creating a Table

A table is created in Microsoft Access to store the data of one entity. A table can be built in Design View, where the user lists the fields and sets their properties, or in Datasheet View, where data is typed directly. In Design View the user gives each field a name, chooses its data type and sets any properties, and this design determines what kind of data the table can hold.

Data Types

Every field in a table must be given a data type, which tells Access what kind of data the field will store. Common data types include Text (for names and addresses), Number (for numeric values used in calculations), Date/Time (for dates), Currency (for money), Yes/No (for true/false values) and AutoNumber (a number that Access generates automatically for each new record, often used as a primary key). Choosing the correct data type keeps the data valid and saves space.

Field Properties and the Primary Key

As well as a data type, each field has properties that control it, such as the field size, a default value, a format, and a validation rule that limits what may be entered (for example, marks between 0 and 100). One field (or a combination) is chosen as the primary key, which uniquely identifies each record; Access marks it and does not allow duplicate or blank values in it. Setting a suitable primary key is an essential part of table design.

Working in Datasheet View

Once a table is designed, data is entered and viewed in Datasheet View, which shows the records in a grid of rows and columns like a spreadsheet. In this view the user can add new records, edit or delete existing records, move between fields, and adjust column widths. The datasheet is the everyday way of looking at and updating the data stored in a table.

Creating a Query

A query is created to retrieve particular data from one or more tables. In the query design grid the user chooses the table, selects the fields to display, and can set a criterion (a condition) for each field to pick out only the records that are wanted; for example the criterion '>80' on the Marks field selects only students who scored above 80. The user can also choose to sort the results and to show or hide particular fields. When the query is run, Access displays the matching records.

Criteria and Sorting

Criteria are conditions written into a query to filter the records. They can use comparison operators such as > (greater than), < (less than) and = (equal to), text values in quotes, and logical operators such as AND and OR to combine conditions. Sorting arranges the results in ascending or descending order of a chosen field, for example listing students from the highest marks to the lowest. Criteria and sorting together let a query answer very specific questions.

Types of Queries

There are two broad kinds of query. A select query retrieves and displays data from tables without changing it; it is the most common type and is used to answer questions and produce lists. An action query changes the data: examples are the update query (which changes values in many records at once), the append query (which adds records) and the delete query (which removes records). Behind all queries lies the language SQL (Structured Query Language), which Access generates automatically from the design grid.

Important Definitions

Term	Definition
Design View	The view used to define a table's fields, data types and properties.
Datasheet View	The grid view used to enter, view and edit the records of a table.
Data type	The kind of data a field can store (Text, Number, Date/Time, etc.).
Field property	A setting that controls a field, such as size or validation rule.
Primary key	The field that uniquely identifies each record in a table.
Query	An object that retrieves selected data answering a question.
Criteria	Conditions in a query that select particular records.

Term	Definition
Select query	A query that retrieves and displays data without changing it.

Key Facts & Rules

Item	Detail
Create a table	Design View (fields + types) or Datasheet View
Data types	Text, Number, Date/Time, Currency, Yes/No, AutoNumber
Primary key	Unique, no duplicates or blanks
Query criteria operators	> , < , = , AND , OR
Sorting	Ascending or descending order
Query types	Select (view) and action (update/append/delete)

Diagrams & Illustrations

Table design: a table design listing fields with their data types and the primary key marked.

Table Design (Fields and Data Types)

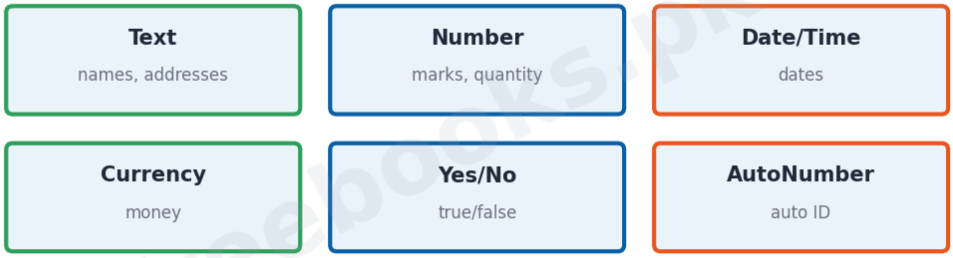
Field Name	Data Type
RollNo (PK)	Number
Name	Text
Marks	Number
DOB	Date/Time

Each field has a name and a data type; PK = primary key

Table design

Field data types: common Access data types such as Text, Number, Date/Time, Currency, Yes/No and AutoNumber.

Common Field Data Types in Access



Field data types

Query design grid: a query design grid showing fields, sorting and a criterion (Marks > 80).

Query Design Grid

Field:	Name	Marks
Table:	STUDENT	STUDENT
Sort:		Descending
Show:	yes	yes
Criteria:		> 80

Criteria '> 80' selects only students scoring above 80

Query design grid

Solved Examples & Practice

Choose a data type

For a field storing a student's date of birth, the correct data type is Date/Time.

Set the primary key

In a STUDENT table, RollNo is made the primary key so no two students share it.

Write a criterion

To find students who scored above 80, put the criterion >80 in the Marks field of a query.

Sort the result

To list students from highest to lowest marks, sort the Marks field in descending order.

Short Questions & Answers

Q1. What is the difference between Design View and Datasheet View?
Ans. Design View is used to define the fields, data types and properties of a table; Datasheet View is used to enter, view and edit the actual records.

Q2. What is a data type? Give two examples.
Ans. The kind of data a field can store, such as Text (for names) and Number (for marks).

Q3. What is a primary key?
Ans. The field that uniquely identifies each record in a table; it cannot contain duplicate or blank values.

Q4. What is a query?
Ans. A database object used to retrieve selected data that answers a question, without changing the stored data.

Q5. What are criteria in a query?
Ans. Conditions written into a query to select only the records that meet them, for example Marks > 80.

Q6. Differentiate a select query and an action query.
Ans. A select query only retrieves and displays data; an action query changes the data (update, append or delete).

Long Questions & Answers

Q1: Describe how a table is created in Access, including data types, field properties and the primary key.

In Microsoft Access, a table is the object that actually stores the data, and it must be carefully designed. A table can be created in one of two views. In Design View, the user builds the structure of the table by listing each field, giving it a name and choosing its properties, and this is the usual way to design a table properly; in Datasheet View, the user can simply start typing data into a grid and Access works out a basic structure automatically. When designing a table, every field must be given a data type, which tells Access what kind of information the field will hold; the common data types are Text, for words such as names and addresses; Number, for numeric values that may be used in calculations, such as marks; Date/Time, for dates; Currency, for amounts of money; Yes/No, for values that can only be true or false; and AutoNumber, a number that Access itself generates for each new record. In addition to its data type, each field has a set of properties that control it more precisely, such as the field size, a default value that is entered automatically, a display format, and a validation rule that restricts what may be typed, for example insisting that a mark lies between 0 and 100. Finally, one field, or occasionally a combination of fields, is chosen as

the primary key, which uniquely identifies every record; Access will not allow the primary key to contain duplicate values or to be left blank. Choosing suitable field names, correct data types, sensible properties and a good primary key produces a well-structured table that stores valid data efficiently.

Q2: Explain how a query is created and used, including criteria and sorting.

A query is the object used in Access to retrieve particular information from the data stored in tables, and it is one of the most powerful features of a database. A query is usually built in the query design grid. The user first chooses the table or tables that hold the required data, and then selects the fields that are to appear in the result. For any of these fields the user may enter a criterion, which is a condition that the records must satisfy in order to be included; for example, placing the criterion greater-than 80 in the Marks field will make the query display only those students who scored above eighty. Criteria may use comparison operators such as greater-than, less-than and equal-to, may contain text values written inside quotation marks, and may be combined using the logical operators AND and OR to express more complex conditions, such as students of a particular class who also scored above a certain mark. The user can also set the sort order of a field, so that the results are arranged in ascending or descending order, for instance listing students from the highest marks down to the lowest, and can choose whether each field is shown or hidden in the result. When the query is run, Access searches the tables and displays just the records and fields that match, without in any way changing the data that is stored. In this way a single well-written query can answer a very specific question about the data quickly and accurately, and the query can be saved and run again whenever the up-to-date answer is needed.

Q3: Describe the different types of queries in Access.

Queries in Access fall into two broad categories according to what they do: select queries and action queries. A select query is the most common and most basic type; its job is simply to retrieve and display data from one or more tables according to the fields, criteria and sort order that the user has specified, and it does not alter the stored data in any way. Select queries are used to answer questions and to produce lists, such as a list of all the students in a particular class or all those who have passed, and their results can also be used as the basis for forms and reports. An action query, by contrast, actually changes the data in the tables, and there are several kinds. An update query changes the values in a field for many records at once, for example adding five bonus marks to every student in a class. An append query adds a group of records from one place to another table. A delete query removes a set of records that meet a given condition, such as deleting all the records of students who have left. A make-table query creates a new table from the results of a query. Because action queries alter data, they must be used carefully. Underlying all of these, whether select or action, is a special language called SQL, which stands for Structured Query Language; SQL is the standard language for working with relational databases, and when a user builds a query in the design grid, Access automatically writes the equivalent SQL statement to carry it out. Thus by choosing the right type of query, a user can not only find data but also update, add to and tidy up the data in the database.

MCQs with Answers

1. The view used to define a table's fields is:
(a) Datasheet View (b) Design View (c) Form View (d) Report View
Correct Answer: (b) Design View. Design View.

2. Data typed and viewed as a grid is in:
(a) Design View (b) Datasheet View (c) Layout View (d) SQL View
Correct Answer: (b) Datasheet View. Datasheet View.

3. The data type for storing money is:
(a) Text (b) Number (c) Currency (d) Yes/No
Correct Answer: (c) Currency. Currency.

4. A number generated automatically for each record is:
(a) Text (b) AutoNumber (c) Currency (d) Memo
Correct Answer: (b) AutoNumber. AutoNumber.

5. A field that uniquely identifies each record is the:
(a) foreign key (b) primary key (c) index (d) criteria
Correct Answer: (b) primary key. The primary key.

6. A condition that selects records in a query is a:
(a) sort (b) criterion (c) format (d) key
Correct Answer: (b) criterion. A criterion.

7. The criterion >80 on Marks selects students who scored:
(a) below 80 (b) above 80 (c) exactly 80 (d) 0
Correct Answer: (b) above 80. Above 80.

8. A query that only displays data is a:
(a) action query (b) select query (c) update query (d) delete query
Correct Answer: (b) select query. A select query.

9. A query that changes many records at once is an:
(a) select query (b) update query (c) form (d) report
Correct Answer: (b) update query. An update query (action query).

10. The language behind Access queries is:
(a) HTML (b) SQL (c) C (d) Python
Correct Answer: (b) SQL. SQL.

Quick Revision Summary

- Create a table in Design View (fields + data types + properties) or Datasheet View.
- Data types: Text, Number, Date/Time, Currency, Yes/No, AutoNumber.
- Field properties (size, validation); primary key = unique, no blanks.
- Datasheet View = enter/edit records in a grid.
- Query: choose fields + criteria (>, <, =, AND, OR) + sorting; select shows matches.

- Query types: select (view) and action (update/append/delete); SQL runs underneath. Notes by freebooks.pk.

Exam Tips

- Distinguish Design View from Datasheet View.
- Learn the common data types and when to use them.
- State the rules for a primary key.
- Write a query criterion with an operator (e.g. >80).
- Explain sorting (ascending/descending).
- Differentiate select and action queries; mention SQL.

2nd Year Computer Science — Punjab Board

CHAPTER 7

Microsoft Access - Forms and Reports

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers Microsoft Access Forms and Reports from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains forms and how they are created, form controls, reports and their sections, and grouping, sorting and totals in reports. These notes are prepared by freebooks.pk.

Forms make entering data easy and reports make printing it neat. This chapter explains these two Access objects that turn a database into a friendly, usable application.

Learning Objectives

- Explain the purpose of a form.
- Describe how a form is created.
- Identify common form controls.
- Explain the purpose of a report.
- Describe the sections of a report.
- Explain grouping, sorting and totals in reports.

Key Concepts

What is a Form?

A form is an Access object that provides an attractive, easy-to-use screen for working with the data in a table, usually showing one record at a time. Forms make entering, viewing and editing data far more convenient and less error-prone than typing directly into a datasheet, because each field is clearly labelled and laid out. Forms are the main way ordinary users interact with a database.

Creating a Form

A form can be created in two main ways. The Form Wizard asks the user a few questions, such as which table and which fields to include, and then builds the form automatically, which is quick and easy. Alternatively the form can be built in Design View, where the user places and arranges the controls by hand for full control over the layout. A form is based on a table or a query, which supplies the data it shows.

Form Controls

The items placed on a form (or report) are called controls. Common controls include the label, which displays fixed text such as a caption; the text box, which shows and lets the user edit the value of a field; the combo box and list box, which let the user choose a value from a list; the check box, for yes/no values; and the command button, which performs an action such as saving a record or moving to the next one when it is clicked. Controls are what make a form interactive.

What is a Report?

A report is an Access object designed to present data in a neatly formatted way for printing on paper. Unlike a form, which is meant for working with data on screen, a report is meant for output; it takes its data from a table or a query and arranges it with headings, groupings, totals and page numbers to produce professional printed documents such as mark sheets, bills and lists. Reports are read-only, so they cannot be used to change the data.

Creating a Report and Its Sections

Like a form, a report can be created quickly with the Report Wizard or built by hand in Design View. A report is divided into sections. The report header appears once at the very start (for the title); the page header appears at the top of every page (for column headings); the detail section prints once for each record and forms the body of the report; the page footer appears at the bottom of every page (for the page number); and the report footer appears once at the very end (for grand totals). These sections give the report its structure.

Grouping and Sorting in Reports

A report can group its records so that related records are printed together under a heading; for example, students can be grouped by class, with each class starting a new group. Within and between groups the records can be sorted into order, such as by name or by marks. Grouping and sorting make a long report easier to read by organising the data logically.

Calculations and Totals

A report can also perform calculations on its data. Using functions such as SUM, AVERAGE, COUNT, MAX and MIN placed in the group footer or report footer, a report can print subtotals for each group and grand totals for the whole report, for example the total marks of each class and the overall average. These automatic calculations make reports very useful for summarising data.

Important Definitions

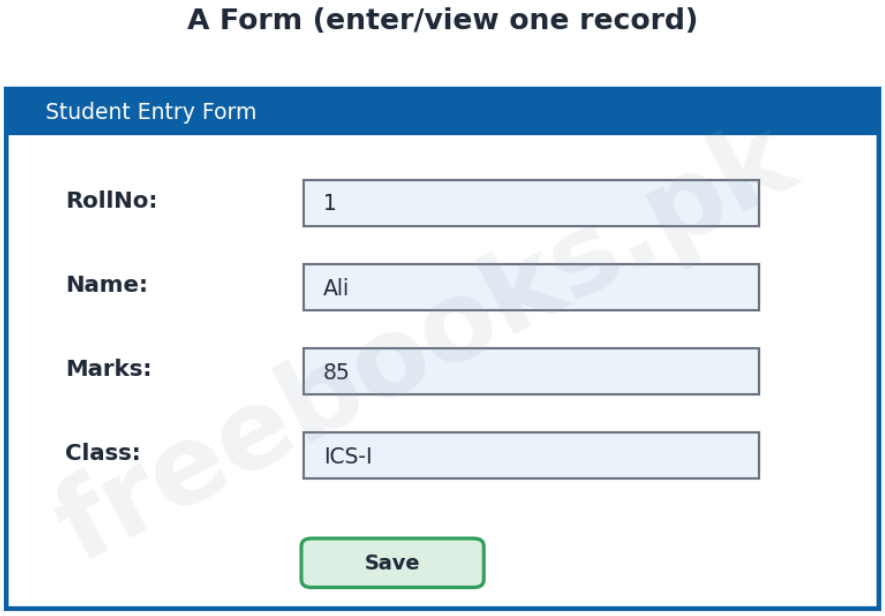
Term	Definition
Form	An object providing a screen layout to enter, view and edit data.
Form Wizard	A tool that builds a form automatically from a table or query.
Control	An item placed on a form or report (label, text box, button, etc.).
Text box	A control that shows and edits the value of a field.
Command button	A control that performs an action when clicked.
Report	An object that presents data in a formatted way for printing.
Detail section	The report section that prints once for each record.
Grouping	Printing related records together under a heading in a report.

Key Facts & Rules

Item	Detail
Form	Enter/view data, one record at a time
Create a form	Form Wizard or Design View (based on table/query)
Controls	Label, text box, combo/list box, check box, command button
Report	Formatted output for printing (read-only)
Report sections	Report header, page header, detail, page footer, report footer
Report totals	SUM, AVERAGE, COUNT in group/report footer

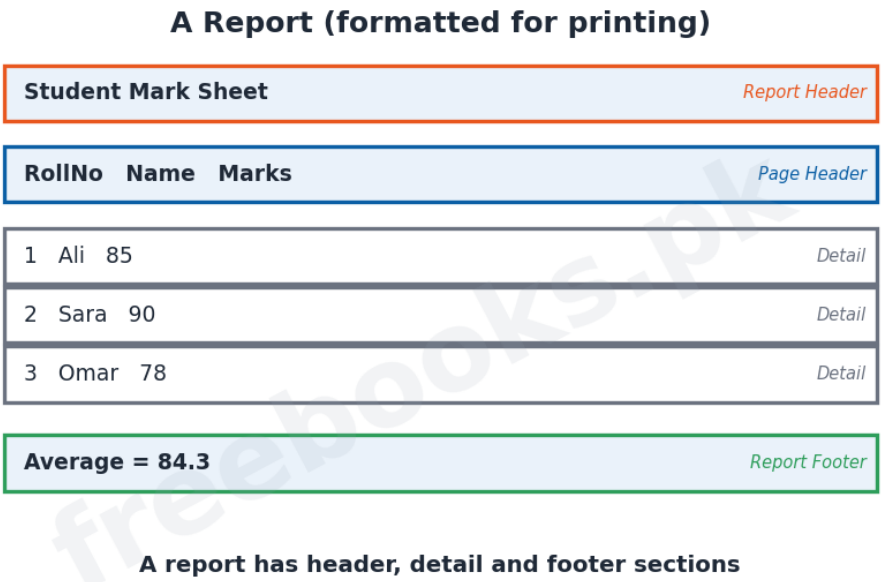
Diagrams & Illustrations

A form: a data entry form showing labelled fields and a Save button for one record.



A form

A report: a report with its header, detail and footer sections producing a printed mark sheet with an average.



A report

Form controls: common controls used on forms and reports: label, text box, combo box, check box, command button and list box.

Form controls

Solved Examples & Practice

Choose the object

To let a clerk type in new students easily, make a form; to print the class result, make a report.

Pick a control

To let the user choose a class from a fixed list on a form, use a combo box.

Report section

Column headings such as 'Name' and 'Marks' are placed in the page header of a report.

Add a total

To print the average marks of a class, put an AVERAGE calculation in the report (group) footer.

Short Questions & Answers

Q1. What is a form?
Ans. An Access object that provides an easy screen layout to enter, view and edit data, usually one record at a time.

Q2. Name two ways to create a form.
Ans. Using the Form Wizard or building it by hand in Design View.

Q3. What is a control? Give two examples.
Ans. An item placed on a form or report, such as a text box (shows/edits a field) and a command button (runs an action).

Q4. What is a report used for?
Ans. To present data in a neatly formatted way for printing, such as a mark sheet; it is read-only.

Q5. Name the section of a report that prints once per record.
Ans. The detail section.

Q6. What is grouping in a report?
Ans. Printing related records together under a heading, for example grouping students by class.

Long Questions & Answers

Q1: Explain what a form is, how it is created, and the common controls used on it.

A form is one of the objects of a Microsoft Access database, and its purpose is to provide a convenient, attractive screen through which users can work with the data held in a table. Instead of typing into a plain datasheet, the user sees a form that usually displays one record at a time, with each field clearly labelled and laid out, which makes entering new data, and viewing or editing existing data, much easier and far less likely to cause mistakes. For this reason the form is the main way in which ordinary users interact with a database. A form can be created in two main ways. The quickest is to use the Form Wizard, which asks

the user a few simple questions, such as which table or query the form should be based on and which fields it should contain, and then builds a complete form automatically. For more control over the appearance, the user can instead build the form by hand in Design View, placing and arranging each item exactly as wanted. Whichever method is used, the form is based on a table or a query, which supplies the data that appears in it. The items that are placed on a form are called controls, and several kinds are commonly used. A label displays fixed text, such as the caption beside a field. A text box shows the value of a field and lets the user edit it, and is the most common control. A combo box or a list box lets the user pick a value from a list rather than typing it. A check box is used for a yes/no value. A command button carries out an action, such as saving the record or moving to the next one, when it is clicked. Together these controls turn a form into an interactive screen for working comfortably with the data.

Q2: Explain what a report is and describe the sections of a report.

A report is the Access object that is designed to present the data of a database in a neat, well-organised way for printing on paper. Whereas a form is intended for working with data interactively on the screen, a report is intended for output; it takes its data from a table or a query and lays it out attractively, with titles, headings, groupings, totals and page numbers, so as to produce professional printed documents such as mark sheets, bills, invoices and lists. Because reports are meant only for output, they are read-only and cannot be used to change the stored data. A report can be created quickly using the Report Wizard, which builds it automatically after asking which data to include, or it can be constructed by hand in Design View for complete control over the layout. Every report is divided into a number of sections, each of which prints at a particular place. The report header appears just once, at the very beginning of the report, and is used for the overall title. The page header appears at the top of every page and is normally used for the column headings, such as 'Name' and 'Marks'. The detail section is the main body of the report; it is printed once for every record in the underlying data, so it is here that the individual records appear. The page footer appears at the bottom of every page and typically holds the page number or the date. Finally, the report footer appears once, at the very end of the report, and is used for grand totals or concluding information. By arranging the data into these sections, a report gives its printed output a clear and consistent structure.

Q3: Describe grouping, sorting and calculations (totals) in an Access report.

Beyond simply listing records, an Access report offers powerful features for organising and summarising data, namely grouping, sorting and calculations. Grouping means arranging the records so that those that belong together are printed together under a common heading; for example, in a report of examination results, the students can be grouped by their class, so that all the students of one class appear together beneath that class name before the next class begins. Grouping makes a long report much easier to read, because the data is broken up into meaningful sections instead of one continuous list. Within the report, and within each group, the records can also be sorted, that is, arranged into a chosen order, such as alphabetical order of name or descending order of marks, so that the reader can find

information easily. In addition, a report can carry out calculations on its data by means of built-in functions such as SUM, which adds values, AVERAGE, which finds the mean, COUNT, which counts records, and MAX and MIN, which find the largest and smallest values. When such a calculation is placed in the footer of a group, it produces a subtotal for that group, for instance the total or average marks of each class; when it is placed in the report footer, it produces a grand total for the whole report, such as the overall average of all students. Through grouping, sorting and these automatic totals, a report can not only display the data but also summarise it clearly, which is what makes reports so useful for presenting the information stored in a database.

MCQs with Answers

1. The object used to enter data one record at a time is the:
(a) report (b) form (c) query (d) table
Correct Answer: (b) form. The form.

2. A quick way to build a form is the:
(a) Report Wizard (b) Form Wizard (c) query (d) macro
Correct Answer: (b) Form Wizard. The Form Wizard.

3. An item placed on a form is a:
(a) control (b) field (c) record (d) key
Correct Answer: (a) control. A control.

4. The control that shows and edits a field is the:
(a) label (b) text box (c) line (d) image
Correct Answer: (b) text box. The text box.

5. A control that runs an action when clicked is a:
(a) label (b) command button (c) text box (d) check box
Correct Answer: (b) command button. A command button.

6. The object designed for printing is the:
(a) form (b) report (c) query (d) table
Correct Answer: (b) report. The report.

7. The report section printed once per record is the:
(a) report header (b) detail (c) page footer (d) report footer
Correct Answer: (b) detail. The detail section.

8. Column headings in a report go in the:
(a) detail (b) page header (c) report footer (d) form
Correct Answer: (b) page header. The page header.

9. Printing students grouped by class is called:
(a) sorting (b) grouping (c) filtering (d) indexing
Correct Answer: (b) grouping. Grouping.

10. A grand total is placed in the:

(a) page header (b) report footer (c) detail (d) form header
Correct Answer: (b) report footer. The report footer.

Quick Revision Summary

- Form = easy screen to enter/view/edit data (one record at a time); made by Form Wizard or Design View.
- Controls: label, text box, combo/list box, check box, command button.
- Report = formatted output for printing (read-only); made by Report Wizard or Design View.
- Report sections: report header, page header, detail (per record), page footer, report footer.
- Grouping puts related records together; sorting orders them.
- Totals: SUM/AVERAGE/COUNT in group footer (subtotal) or report footer (grand total). Notes by freebooks.pk.

Exam Tips

- State the purpose of a form vs a report.
- Give the two ways to create a form/report.
- List the common controls and their jobs.
- Name the report sections in order.
- Explain grouping and sorting.
- Show where subtotals and grand totals go.

Freebooks.pk

Free Notes • Past Papers • Guides for Students

2nd Year Computer Science — Punjab Board

CHAPTER 8

Getting Started with C

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers Getting Started with C from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It introduces the C language, the structure of a C program, comments and header files, the compilation process and a first simple program. These notes are prepared by freebooks.pk.

C is a powerful and widely used programming language. This chapter introduces C and shows how a C program is written, structured and turned into a running program.

Learning Objectives

- State what C is and its features.
- Describe the structure of a C program.
- Explain header files and comments.
- Describe the compilation process.
- Write and understand a simple C program.
- State the C character set and keywords.

Key Concepts

The C Language

C is a general-purpose, high-level programming language developed by Dennis Ritchie in the early 1970s. It is powerful, efficient and portable, meaning a C program can run on many different computers with little change, and it is used to write operating systems, application software and much other software. Because it teaches the fundamentals of programming clearly, C is widely used as a first serious programming language.

Structure of a C Program

A C program has a definite structure. It usually begins with preprocessor directives such as #include, which bring in header files. It then contains one or more functions, and every C program must have a function called main(), where execution begins. Inside a function, statements are written between braces { and }, each statement normally ending with a semicolon. A simple program consists of the header, the main() function, some declarations and statements, and a return statement.

Header Files and Comments

A header file is a file, included with a #include directive, that contains ready-made declarations the program needs; for example #include <stdio.h> brings in the standard input/output functions such as printf and scanf. Comments are explanatory notes added to the source code to make it easier to understand; they are ignored by the compiler and are written between /* and */ (or after // for a single line). Good comments make a program easier to read and maintain.

The Compilation Process

A C program is written as source code in a file (with the extension .c), but the computer cannot run this directly. A special program called a compiler translates the source code into machine code, first producing object code and then, after linking, an executable file (with the extension .exe on Windows) that the computer can run. If the source contains errors, the compiler reports them so that the programmer can correct them before the program will compile.

A First C Program

A simple first C program prints a message on the screen. It includes the header stdio.h, defines the main() function, and uses the printf() function to display text; for example printf("Hello, World!"); prints the words Hello, World! on the screen. The program ends with return 0; inside main(). Running this program shows how source code produces visible output.

The C Character Set

The C character set is the set of characters that may be used to write a C program. It includes the letters A to Z and a to z, the digits 0 to 9, special symbols such as + - * / = ( ) { } ; and white-space characters such as spaces and new lines. These characters are combined to form the words and symbols of the language.

Keywords and Identifiers

Keywords are reserved words that have a fixed, special meaning in C, such as int, float, if, else, while and return; they cannot be used for any other purpose. Identifiers are the names that the programmer gives to things such as variables and functions; an identifier must begin with a letter or underscore, may contain letters, digits and underscores, and must not be a keyword. Choosing clear identifiers makes a program easier to understand.

Important Definitions

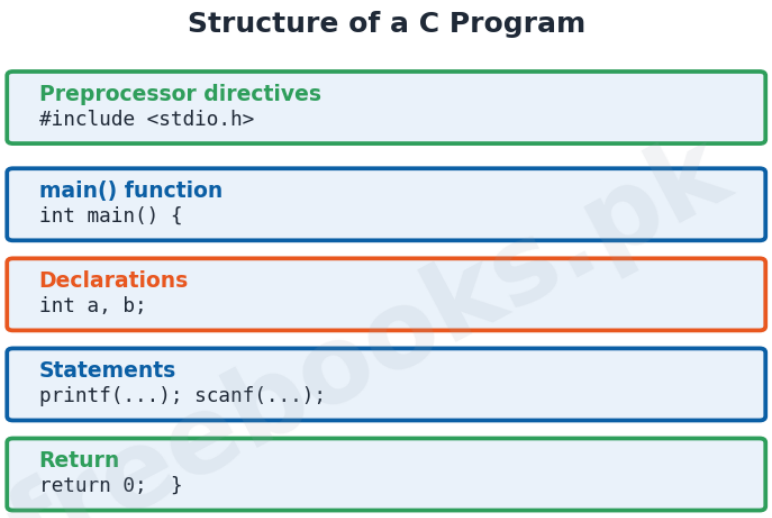
Term	Definition
C	A general-purpose high-level programming language by Dennis Ritchie.
main()	The function where a C program begins execution.
Preprocessor directive	A line beginning with # (e.g. #include) processed before compilation.
Header file	A file included to provide ready-made declarations (e.g. stdio.h).
Comment	An explanatory note in the code, ignored by the compiler.
Compiler	A program that translates source code into machine code.
Keyword	A reserved word with a fixed meaning (e.g. int, if, while).
Identifier	A name given by the programmer to a variable or function.

Key Facts & Rules

Item	Detail
C	High-level language by Dennis Ritchie (~1972)
Program start	main() function
Include a header	#include <stdio.h>
Comment	/* ... */ or // ... (ignored by compiler)
Compile	source (.c) -> object -> executable (.exe)
Print text	printf("...");

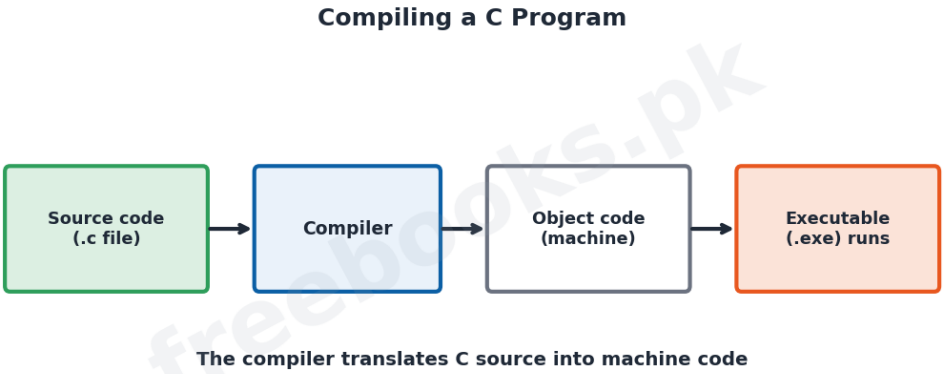
Diagrams & Illustrations

Structure of a C program: the structure of a C program: preprocessor directives, the main() function, declarations, statements and a return.



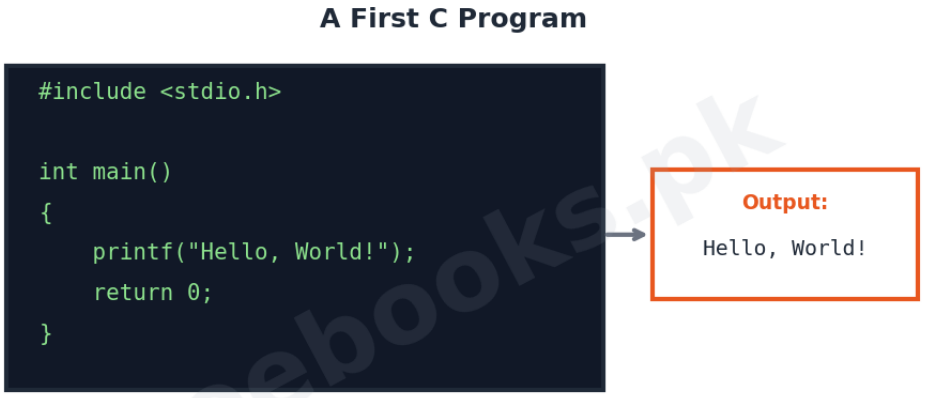
Structure of a C program

Compiling a C program: the compilation process turning C source code into object code and then an executable file.



Compiling a C program

A first C program: a simple C program that prints Hello, World! and its output.



A first C program

Solved Examples & Practice

Include a header

To use printf and scanf, the program must include the header with the line #include <stdio.h>.

Print a message

printf("Pakistan"); displays the word Pakistan on the screen.

Valid identifier

'marks1' is a valid identifier, but '1marks' is not because it starts with a digit, and 'int' is not because it is a keyword.

Compile then run

A .c source file is first compiled into an .exe file, which is then run to produce the output.

Short Questions & Answers

Q1. Who developed C and when?
Ans. C was developed by Dennis Ritchie in the early 1970s (around 1972).

Q2. What is the main() function?
Ans. The function where the execution of every C program begins.

Q3. What is a header file? Give an example.
Ans. A file included to provide ready-made declarations, such as stdio.h for input/output functions.

Q4. What is a comment in C?
Ans. An explanatory note added to the code that is ignored by the compiler, written between /* and */ or after //.

Q5. What is the role of the compiler?
Ans. To translate the C source code into machine code (an executable) that the computer can run.

Q6. Differentiate a keyword and an identifier.
Ans. A keyword is a reserved word with a fixed meaning (e.g. int); an identifier is a name the programmer gives to a variable or function.

Long Questions & Answers

Q1: Describe the C language and the structure of a C program.

C is a general-purpose, high-level programming language that was developed by Dennis Ritchie at Bell Laboratories in the early 1970s. It became extremely popular because it is powerful and efficient, giving the programmer a great deal of control, and because it is portable, which means that a program written in C can be run on many different kinds of computer with little or no change. C has been used to write operating systems, compilers, application programs and much other software, and because it clearly teaches the fundamental ideas of programming it is very widely used as a first serious programming language for students. Every C program follows a definite structure. It usually begins with one or more preprocessor directives, lines that start with a hash sign, the most common being #include, which brings a header file into the program. After the directives come the functions of the program, and every C program must contain one special function called main(), because this is the point at which the program starts running when it is executed. The body of a function is written between an opening brace and a closing brace, and inside it come the declarations, which introduce the variables the program will use, followed by the statements, which are the instructions that actually do the work, such as printing output or reading input; each statement normally ends with a semicolon. A typical simple program

therefore consists of a header line, the main() function, some declarations, some statements and a return statement at the end. Understanding this structure is the first step in learning to write C programs.

Q2: Explain header files, comments and the compilation process in C.

Three ideas are important when beginning to write C programs: header files, comments and compilation. A header file is a file that contains ready-made declarations which a program needs in order to use certain built-in functions, and it is brought into the program by a preprocessor directive of the form #include. For example, the line #include <stdio.h> includes the standard input/output header, which makes available the functions printf, used to display output, and scanf, used to read input; without including this header, a program could not use these functions. Comments are explanatory notes that the programmer writes inside the source code to describe what the program, or a part of it, does; they are meant only for human readers and are completely ignored by the compiler, and they are written either between the symbols /* and */, which can span several lines, or after //, which comments out the rest of a single line. Good comments make a program much easier to read, understand and correct later. Finally, although the program is written in C as source code and saved in a file with the extension .c, the computer cannot run this source code directly, because it understands only machine code. A special program called a compiler is therefore used to translate the source code into machine code: it first produces object code and then, after a step called linking, an executable file, which on Windows has the extension .exe and which the computer can actually run. If the source code contains any errors, the compiler detects them and reports them to the programmer, who must correct them before the program will successfully compile and run. Thus header files supply needed functions, comments explain the code, and the compiler turns the finished source into a working program.

Q3: Explain, with an example, a simple C program and describe the C character set, keywords and identifiers.

A good way to understand C is to look at a simple complete program and the basic elements from which it is built. A first program usually just prints a message on the screen. Such a program begins with the directive #include <stdio.h> to make the output function available; it then defines the function main(); inside the braces of main() it uses the statement printf("Hello, World!"); which displays the words Hello, World! on the screen; and it ends with the statement return 0; before the closing brace. When this program is compiled and run, the words Hello, World! appear as output, showing clearly how the instructions in the source code produce a visible result. The elements used to write such programs come from the C character set, which is the collection of characters allowed in a C program; it includes the capital letters A to Z and small letters a to z, the digits 0 to 9, special symbols such as the plus, minus, multiply, divide, equals, brackets, braces and semicolon, and white-space characters such as the space, the tab and the new line. From these characters two important kinds of word are formed. Keywords are reserved words that have a fixed, special meaning in the language, such as int, float, char, if, else, while, for and return, and because C already uses them for a definite purpose they may not be used for anything else. Identifiers, on the

other hand, are the names that the programmer invents for things such as variables and functions; an identifier must begin with a letter or an underscore, may then contain letters, digits and underscores, and must not be the same as any keyword. Choosing meaningful identifiers, such as marks or total, makes a program much easier to read and understand.

MCQs with Answers

1. C was developed by:
(a) Bill Gates (b) Dennis Ritchie (c) Charles Babbage (d) James Gosling
Correct Answer: (b) Dennis Ritchie. Dennis Ritchie.

2. Execution of a C program begins at:
(a) the first line (b) main() (c) the last function (d) printf
Correct Answer: (b) main(). The main() function.

3. A line starting with # is a:
(a) comment (b) preprocessor directive (c) statement (d) keyword
Correct Answer: (b) preprocessor directive. A preprocessor directive.

4. The header for input/output is:
(a) conio.h (b) stdio.h (c) math.h (d) string.h
Correct Answer: (b) stdio.h. stdio.h.

5. The function used to print output is:
(a) scanf (b) printf (c) gets (d) main
Correct Answer: (b) printf. printf.

6. A comment in C is ignored by the:
(a) user (b) compiler (c) printer (d) keyboard
Correct Answer: (b) compiler. The compiler.

7. A program that translates C into machine code is a:
(a) editor (b) compiler (c) browser (d) modem
Correct Answer: (b) compiler. A compiler.

8. Which is a keyword in C?
(a) marks (b) int (c) total (d) sum
Correct Answer: (b) int. int.

9. Most C statements end with a:
(a) comma (b) semicolon (c) full stop (d) colon
Correct Answer: (b) semicolon. Semicolon.

10. Which is a valid identifier?
(a) 1num (b) int (c) num1 (d) float
Correct Answer: (c) num1. num1.

Quick Revision Summary

- C = high-level, portable language by Dennis Ritchie (~1972).

- Structure: #include directives, main() function, declarations, statements, return.
- Header files (e.g. stdio.h) supply functions (printf, scanf); comments /* */ // ignored by compiler.
- Compilation: source (.c) -> object -> executable (.exe); compiler reports errors.
- printf("..."); prints output; every statement ends with a semicolon.
- Keywords (int, if, while...) are reserved; identifiers = programmer's names (start with letter/underscore). Notes by freebooks.pk.

Exam Tips

- State who developed C and its key features (portable, efficient).
- Draw the structure of a C program.
- Explain #include and give a header example.
- Describe the compilation process (source -> executable).
- Write a simple printf program.
- Distinguish keywords from identifiers (with valid/invalid examples).

Freebooks.pk

Free Notes • Past Papers • Guides for Students

2nd Year Computer Science — Punjab Board

CHAPTER 9

Elements of C

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers the Elements of C from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains variables and constants, data types, operators, expressions and type conversion, the basic building blocks of every C program. These notes are prepared by freebooks.pk.

Before writing real programs, we must know the elements from which they are built. This chapter explains variables, constants, data types and operators in C.

Learning Objectives

- Define a variable and a constant.
- State the basic data types of C.
- Declare and initialise variables.
- Describe the arithmetic, relational, logical and assignment operators.
- Form and evaluate expressions.
- Explain type conversion.

Key Concepts

Variables

A variable is a named location in the computer's memory used to store a value that can change while the program runs. Every variable has a name (an identifier), a data type that decides what kind of value it can hold, and a value. Before it is used, a variable must be declared, for example int age; which reserves memory for an integer called age. A variable can also be initialised, that is, given a starting value, as in int age = 20;.

Constants

A constant is a value that does not change while the program runs. Constants may be numbers (such as 100 or 3.14), single characters written in single quotes (such as 'A'), or strings of characters written in double quotes (such as "Pakistan"). A named constant can be created using #define or the const keyword, for example #define PI 3.14, so that the fixed value can be used by name throughout the program.

Data Types

The data type of a variable tells C what kind of value it will store and how much memory to use. The basic data types in C are int, for whole numbers such as 5 or -20; float, for numbers with a decimal part such as 3.14; double, for decimal numbers needing greater precision; and char, for a single character such as 'A'. Choosing the correct data type ensures the data is stored correctly and efficiently.

Arithmetic Operators

Operators are symbols that perform operations on values. The arithmetic operators carry out calculations: + (addition), - (subtraction), * (multiplication), / (division) and % (modulus,

which gives the remainder of a division). For example, 7 % 3 gives 1. These operators, combined with variables and constants, are used to form arithmetic expressions such as a + b * c.

Relational and Logical Operators

Relational operators compare two values and give a result that is true or false; they are < (less than), > (greater than), <= (less than or equal), >= (greater than or equal), == (equal to) and != (not equal to). Logical operators combine such conditions: && (AND, true only if both are true), || (OR, true if either is true) and ! (NOT, which reverses a condition). These operators are used to make decisions in a program.

Assignment Operators and Expressions

The assignment operator = stores the value on its right into the variable on its left, as in sum = a + b;. C also provides shorthand assignment operators such as += , -= , *= and /=; for example, x += 5; means x = x + 5;. An expression is a combination of variables, constants and operators that produces a value, such as (a + b) / 2; the computer evaluates the expression according to the rules of operator precedence.

Type Conversion

Type conversion is the changing of a value from one data type to another. It may happen automatically (implicit conversion), for example when an int is used where a float is expected, or it may be done deliberately by the programmer (explicit conversion or type casting), by writing the required type in brackets, as in (float)a. Type conversion is important to make sure that calculations, especially division, give the correct kind of result.

Important Definitions

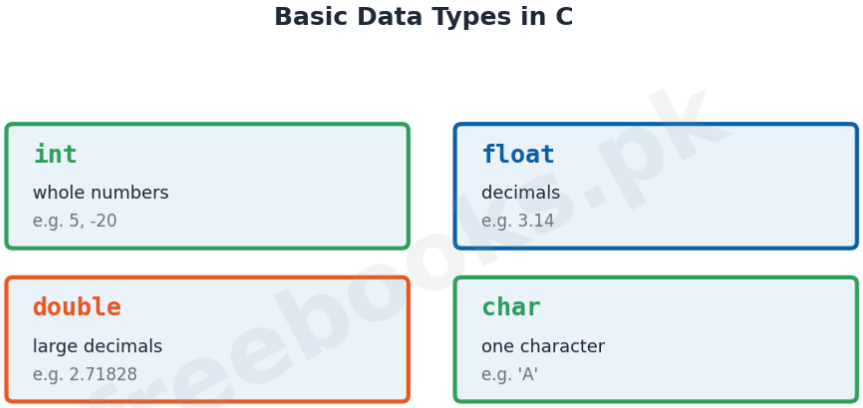
Term	Definition
Variable	A named memory location whose value can change.
Constant	A value that does not change during the program.
Data type	The kind of value a variable can store (int, float, char, double).
Declaration	Stating a variable's name and type before use (e.g. int age;).
Operator	A symbol that performs an operation on values.
Expression	A combination of variables, constants and operators giving a value.
Modulus (%)	An operator giving the remainder of a division.
Type conversion	Changing a value from one data type to another.

Key Facts & Rules

Item	Detail
Declare a variable	int age; (type name;)
Initialise	int age = 20;
Data types	int, float, double, char
Arithmetic	+ - * / % (remainder)
Relational	< > <= >= == !=
Logical	&& || !
Shorthand assignment	x += 5; means x = x + 5;

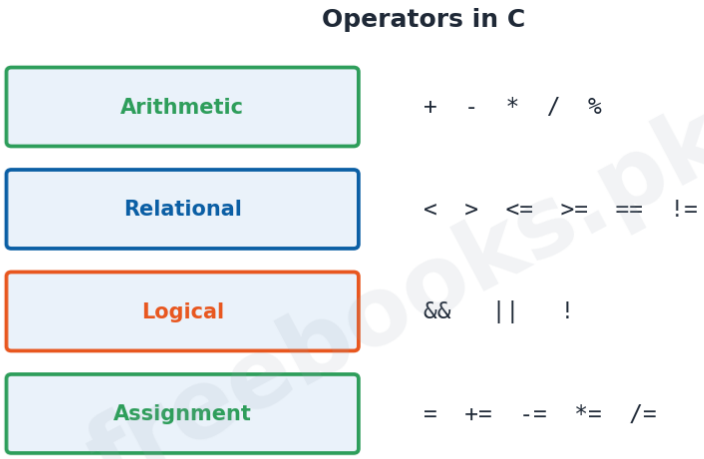
Diagrams & Illustrations

Basic data types in C: the basic C data types: int (whole numbers), float and double (decimals) and char (a single character).



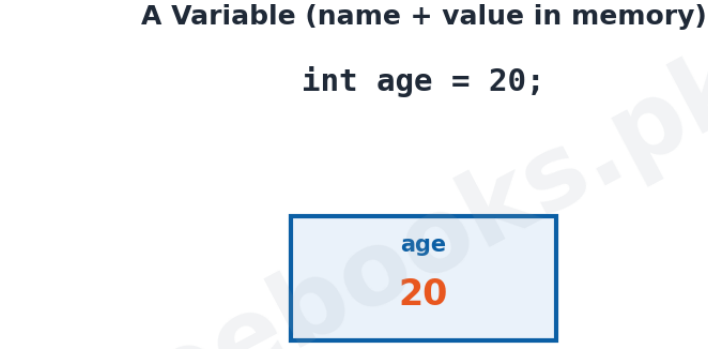
Basic data types in C

Operators in C: the groups of operators in C: arithmetic, relational, logical and assignment.



Operators in C

A variable: a variable shown as a named memory location holding a value (int age = 20).



A variable

Solved Examples & Practice

Declare and initialise

int marks = 85; declares an integer variable called marks and gives it the value 85.

Modulus

The expression 10 % 3 gives 1, the remainder when 10 is divided by 3.

Relational result

The expression 5 > 3 is true, and 5 == 3 is false.

Shorthand

If x is 10, then x += 5; makes x become 15.

Short Questions & Answers

Q1. What is a variable?
Ans. A named location in memory used to store a value that can change while the program runs.

Q2. Differentiate a variable and a constant.
Ans. A variable's value can change during the program; a constant's value stays the same.

Q3. Name the basic data types of C.
Ans. int (whole numbers), float and double (decimals), and char (a single character).

Q4. What does the modulus operator (%) do?
Ans. It gives the remainder of a division, for example 7 % 3 gives 1.

Q5. Name the logical operators.
Ans. && (AND), || (OR) and ! (NOT).

Q6. What is type conversion?
Ans. Changing a value from one data type to another, either automatically or by type casting.

Long Questions & Answers

Q1: Explain variables, constants and data types in C with examples.

The most basic elements used to store information in a C program are variables and constants. A variable is a named location in the computer's memory that is used to hold a value which may change while the program is running. Every variable has three things: a name, which is an identifier chosen by the programmer, such as age or marks; a data type, which decides what kind of value the variable can hold; and a value, the actual data stored in it. Before a variable can be used it must be declared, which reserves memory for it and states its type, as in the declaration int age; and it may also be initialised, that is, given a starting value at the same time, as in int age = 20;. A constant, in contrast, is a value that does not change while the program runs. Constants may be integer or real numbers such as 100 or 3.14, single characters written between single quotes such as 'A', or strings of characters written between double quotes such as "Pakistan"; a named constant can be created with #define or const, for example #define PI 3.14, so that a fixed value can be referred to by a meaningful name. Closely tied to variables is the idea of the data type, which tells C what kind of value a variable will store and how much memory to give it. The basic data types are int, used for whole numbers such as 5 or -20; float, used for numbers that have a decimal part, such as 3.14; double, used for decimal numbers that need greater precision; and char, used for a single character such as 'A'. Choosing the correct data type for each variable ensures that data is stored accurately and that memory is used efficiently.

Q2: Describe the arithmetic, relational, logical and assignment operators in C.

Operators are the special symbols that tell C to perform operations on data, and C provides several groups of them. The arithmetic operators are used for calculations: the plus sign adds, the minus sign subtracts, the asterisk multiplies, the slash divides, and the percent sign, called the modulus operator, gives the remainder left after a division, so that 10 % 3 gives 1. Using these, the programmer can build arithmetic expressions such as a + b * c. The relational operators are used to compare two values, and each gives a result that is either true or false: less-than, greater-than, less-than-or-equal, greater-than-or-equal, the double equals sign for is-equal-to, and the exclamation-equals for is-not-equal-to. For example, the comparison 5 > 3 is true while 5 == 3 is false. The logical operators are used to combine such conditions: the double ampersand means AND and is true only when both conditions are true; the double vertical bar means OR and is true when at least one condition is true; and the exclamation mark means NOT and reverses the truth of a condition. Relational and logical operators together allow a program to test conditions and make decisions. Finally, the assignment operators are used to store values in variables. The single equals sign is the basic assignment operator, which copies the value on its right into the variable on its left, as in sum = a + b;. C also offers shorthand assignment operators such as +=, -=, *= and /=; for instance x += 5; is a short way of writing x = x + 5;. Together these four groups of operators provide everything needed to calculate, compare, combine and store data in a program.

Q3: Explain expressions and type conversion in C.

An expression in C is any valid combination of variables, constants and operators that the computer can evaluate to produce a single value; for example a + b, (a + b) / 2 and marks > 50 are all expressions. When the computer evaluates an expression that contains several operators, it does not simply work from left to right but follows the rules of operator precedence, which decide the order in which the operators are applied; for instance, multiplication and division are done before addition and subtraction, so that in the expression 2 + 3 * 4 the multiplication is performed first, giving 14, and brackets can be used to change this order when needed. Expressions are the means by which all calculations and tests in a program are carried out. Closely related to expressions is the idea of type conversion, which is the changing of a value from one data type to another. Type conversion can happen in two ways. Implicit conversion is done automatically by C when values of different types are mixed in an expression; for example, if an integer is used in a calculation that also involves a float, C automatically converts the integer to a float so that the arithmetic can be carried out correctly. Explicit conversion, also called type casting, is done deliberately by the programmer by writing the desired type in brackets in front of the value, as in (float)a, which treats the value of a as a float. Type conversion is important because it affects the result of calculations; a common example is integer division, where dividing one integer by another discards the fractional part, so a programmer who wants a decimal answer must convert one of the values to a float first. Understanding expressions and type conversion therefore helps a programmer to write calculations that give correct results.

MCQs with Answers

1. A named memory location whose value can change is a:
(a) constant (b) variable (c) keyword (d) operator
Correct Answer: (b) variable. A variable.

2. A value that does not change is a:
(a) variable (b) constant (c) function (d) loop
Correct Answer: (b) constant. A constant.

3. The data type for whole numbers is:
(a) float (b) int (c) char (d) double
Correct Answer: (b) int. int.

4. The data type for a single character is:
(a) int (b) char (c) float (d) string
Correct Answer: (b) char. char.

5. The operator that gives a remainder is:
(a) / (b) % (c) * (d) +
Correct Answer: (b) %. % (modulus).

6. Which is a relational operator?
(a) + (b) && (c) == (d) =
Correct Answer: (c) ==. == (equal to).

7. The logical AND operator is:
(a) && (b) || (c) ! (d) &
Correct Answer: (a) &&. &&.

8. x += 5; is the same as:
(a) x = 5 (b) x = x + 5 (c) x = x - 5 (d) x == 5
Correct Answer: (b) x = x + 5. x = x + 5;.

9. A combination of variables and operators giving a value is an:
(a) expression (b) identifier (c) array (d) header
Correct Answer: (a) expression. An expression.

10. Changing a value from int to float is:
(a) type conversion (b) assignment (c) declaration (d) comment
Correct Answer: (a) type conversion. Type conversion.

Quick Revision Summary

- Variable = named memory holding a changeable value; declare (int age;) then use.
- Constant = fixed value (numbers, 'A', "text"); #define PI 3.14.
- Data types: int, float, double, char.
- Operators: arithmetic (+ - * / %), relational (< > == !=), logical (&& || !), assignment (= += ...).
- Expression = variables + constants + operators -> a value (operator precedence).

- Type conversion: implicit (automatic) or explicit ((float)a) casting. Notes by freebooks.pk.

Exam Tips

- Define variable and constant with examples.
- List the four basic data types and their uses.
- Show a declaration and an initialisation.
- Give each operator group with its symbols.
- Explain the modulus operator with an example.
- Describe implicit and explicit type conversion.

Freebooks.pk

Free Notes • Past Papers • Guides for Students

2nd Year Computer Science — Punjab Board

CHAPTER 10

Input / Output

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers Input / Output from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains how a C program takes input from the keyboard and shows output on the screen using the printf and scanf functions, format specifiers, field width, precision and escape sequences. These notes are prepared by freebooks.pk.

By standard input and output we mean the keyboard and the monitor. In C these operations are carried out by two standard functions, printf and scanf, available through the stdio.h library.

Learning Objectives

- Explain the purpose of the standard input/output library (stdio.h).
- Use the printf function to display formatted output.
- Use format specifiers such as %d, %f, %c and %s.
- Control field width and precision of output.
- Use the scanf function and the address operator (&) to take input.
- Describe common escape sequences.

Key Concepts

Standard Input and Output

In C, the standard input/output library provides the functions needed to perform input and output. By standard input and output we mean the keyboard and the monitor respectively. Input and output are carried out by two standard functions, printf() and scanf(), which become available when the header file stdio.h is included in the program with the directive #include <stdio.h>. Without this library the compiler would not know the meaning of printf and scanf.

The printf Function

The standard library function printf (pronounced print-eff) is used for formatted output, that is, to display results on the screen. It takes a format string and an optional list of variables, and has the form printf(format string, var1, var2, ...);. The first argument is always a string enclosed in double quotes, called the format string; it may contain ordinary text and one or more format specifiers. The values of the variables listed after the format string are displayed in place of the specifiers.

Format Specifiers

A format specifier tells printf what kind of value to display and where to place it in the output. The common specifiers are %d for an integer (int), %f for a float or double, %c for a single character, %s for a string, and %lf for a double. For example, in printf("Area = %d", area); the %d is replaced by the value of the integer variable area. The type of each specifier must match the type of the variable it displays.

Field Width and Precision

The output of printf can be formatted more neatly by controlling field width and precision. Placing a number between the % and the specifier sets the field width, that is, the minimum number of columns used to display the value; for example printf("Area = %4d", area); reserves four columns for the value. For real numbers a precision can also be given after a dot, as in %6.2f, which displays the number in six columns with two digits after the decimal point.

Escape Sequences

Escape sequences are special characters written in the format string using a backslash (\). They cause an escape from the normal meaning of the string so that the next character is treated specially. Common escape sequences are \n (new line), \t (horizontal tab), \b (backspace) and \f (form feed). Because the format string itself is enclosed in double quotes, a single or double quote inside it must be written with a backslash as \' and \", and a backslash itself is written as \\.

The scanf Function

Most useful programs are interactive, meaning they take input from the user. The scanf function (pronounced scan-eff) reads input typed at the keyboard and stores it in variables. It has the form scanf(format string, &var1, &var2, ...);. The format string of scanf consists only of format specifiers, no other text. When execution reaches a scanf statement the program pauses until the user enters a value and presses the Return key.

The Address Operator (&)

In a call to scanf, each variable name is preceded by an ampersand (&), which in C is the address of operator. The & tells scanf the address in memory where the input value should be stored, so scanf can place the value directly into that variable. If the & is omitted, scanf cannot locate the variable in memory, the value is not stored, and the variable is left holding a meaningless garbage value.

Character Input Functions

Besides scanf, C provides functions specialised for reading single characters. scanf can read a character but needs the Return key to be pressed after the input. In situations such as games where each key press must act at once, functions like getch and getche are used; getch reads a character without showing it on the screen (without echo) while getche reads and displays (echoes) it. These functions belong to the conio (console input output) library.

Important Definitions

Term	Definition
Standard input/output	Input from the keyboard and output to the monitor.
stdio.h	The standard input/output header file that defines printf and scanf.

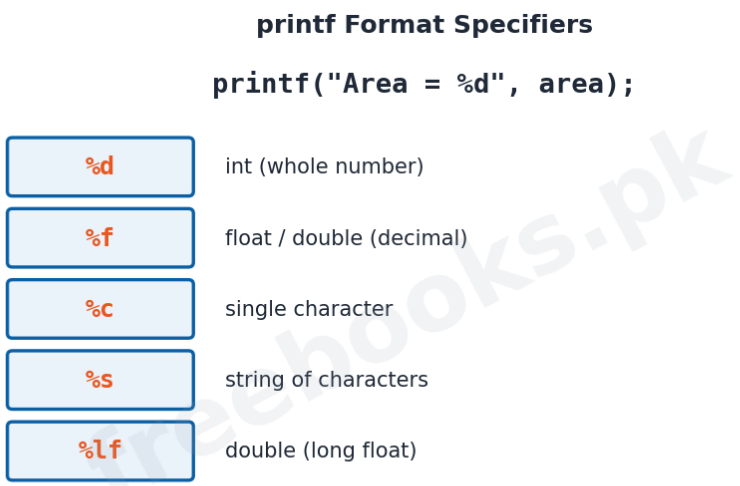
Term	Definition
printf	A standard function used to display formatted output on the screen.
scanf	A standard function used to read input from the keyboard into variables.
Format string	The string of text and format specifiers given to printf or scanf.
Format specifier	A symbol (e.g. %d, %f) that states the type of value to display or read.
Escape sequence	A backslash character combination (e.g. \n, \t) with a special meaning.
Address operator (&)	The operator that gives the memory address of a variable.

Key Facts & Rules

Item	Detail
Include the library	#include <stdio.h>
Output	printf("format string", var1, var2, ...);
Input	scanf("format string", &var1, &var2, ...);
Common specifiers	%d int, %f float, %c char, %s string, %lf double
Field width	printf("%4d", area); (four columns)
Precision	printf("%6.2f", height); (2 decimal places)
New line	\n starts a new line in the output

Diagrams & Illustrations

printf format specifiers: the common printf format specifiers (%d, %f, %c, %s, %lf) and the type each one displays.



printf format specifiers

scanf and the & operator: how scanf reads a value from the keyboard and stores it at a variable's address using the & operator.

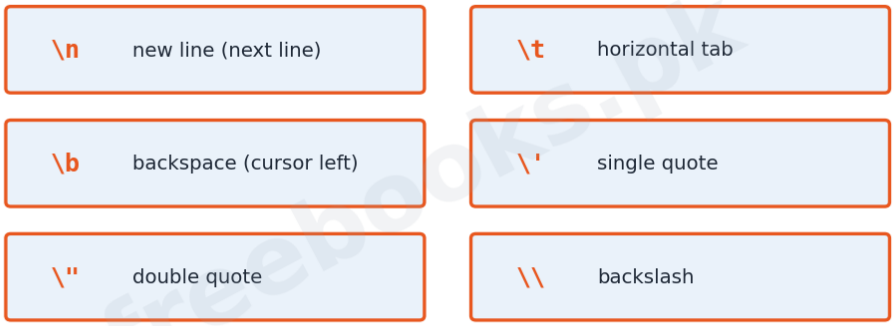
scanf Reads Input into a Variable's Address (&)

```
scanf("%lf", &kilometer);
```

scanf and the & operator

Escape sequences: the common escape sequences in C such as new line, tab, backspace and quotes.

Escape Sequences in C



Escape sequences

Solved Examples & Practice

Display an integer

printf("Area = %d", area); shows the text Area = followed by the value stored in the integer variable area.

Read a number

scanf("%lf", &kilometer); waits for the user to type a number and stores it in the double variable kilometer.

Field width

printf("Area = %4d", area); prints the value of area in a field four columns wide.

Escape sequence

printf("Name\tRoll_No"); prints Name, then a tab space, then Roll_No on the same line.

Short Questions & Answers

Q1. What are standard input and output in C?
Ans. Input from the keyboard and output to the monitor, handled through the stdio.h library.

Q2. What is the use of the printf function?
Ans. It displays formatted output on the screen according to its format string.

Q3. What is a format specifier?
Ans. A symbol such as %d or %f that tells printf/scanf the type of value to display or read.

Q4. Why is the & used in scanf?
Ans. & is the address operator; it gives scanf the memory address where the input value should be stored.

Q5. What happens if & is omitted in scanf?
Ans. scanf cannot find the variable in memory, so the value is not stored and the variable holds garbage.

Q6. What is an escape sequence? Give one example.
Ans. A backslash combination with a special meaning, for example \n which moves to a new line.

Long Questions & Answers

Q1: Explain the printf function with its format string, format specifiers, field width and precision.

The printf function, whose name is pronounced print-eff, is the standard library function used in C for formatted output, that is, for displaying results on the screen. It becomes available when the standard input/output library stdio.h is included in the program. A call to printf has the general form printf(format string, var1, var2, ...); where the first argument is always a string enclosed in double quotes. This string is called the format string and it may contain two kinds of things: ordinary text, which is printed exactly as written, and format specifiers, which are replaced by the values of the variables listed after the string. A format specifier begins with a percent sign and states the type of value to be displayed: %d is used for an integer, %f for a float or double, %c for a single character, %s for a string of characters and %lf for a double. The specifiers must appear in the same order as the variables they display, and their types must match. For example, the statement printf("Area of Square = %d", area); prints the text Area of Square = and then the value stored in the integer variable area in place of %d. The appearance of the output can be improved by controlling the field width

and the precision. Writing a number between the percent sign and the specifier sets the field width, the minimum number of columns used for the value; thus printf("Area = %4d", area); displays the value in a field four columns wide. For real numbers a precision may also be given after a dot, as in %6.2f, which prints the number in six columns with two digits after the decimal point. In this way printf gives the programmer full control over how results are shown.

Q2: Explain the scanf function and the role of the address operator (&) in taking input.

Most useful programs are interactive, which means that they take input from the user while they run. The scanf function, pronounced scan-eff, is the standard C function used to read input typed at the keyboard and to store it in variables; it is versatile because it can read numbers, characters and strings. A call to scanf has the form scanf(format string, &var1, &var2, ...);. Unlike the format string of printf, the format string of scanf consists only of format specifiers and contains no other text, because its only job is to tell scanf what kind of data to read into each variable. When the flow of execution reaches a scanf statement, the program stops and waits until the user enters a value and presses the Return key, after which the value is stored and execution continues. A very important feature of scanf is that each variable name in the list is preceded by an ampersand, the symbol &, which in C is the address of operator. The ampersand gives scanf the address in memory of the variable, so that scanf knows exactly where to place the value it reads. If the programmer forgets the &, scanf is unable to locate the variable in memory; the value is therefore not stored and the variable is left holding a meaningless garbage value, which is a common source of program errors. For reading single characters without waiting for the Return key, C also provides special functions such as getch and getche from the conio library; getch reads a character without echoing it on the screen, while getche reads and displays it.

Q3: What are escape sequences? Describe the common escape sequences used in C with examples.

Escape sequences are special characters that are written inside the format string of a printf statement using a backslash. The backslash causes an escape from the normal interpretation of the string, so that the character following it is recognised as having a special meaning rather than being printed literally. They are needed because certain effects, such as moving to a new line, and certain characters, such as the double quote, cannot be placed directly in a format string. The most commonly used escape sequence is \n, the newline character, which moves the cursor to the beginning of the next line; for example printf("\nAmir"); prints Amir on a new line. The escape sequence \t is the horizontal tab, which moves the cursor to the next tab position and is useful for arranging output in neat columns, as in printf("Name\t\tRoll_No\t\tMarks");. The sequence \b is the backspace, which moves the cursor one space to the left, and \f is the form feed, which moves to the next page on a printer. Some characters need an escape sequence because of the way strings are written: since the format string is enclosed in double quotes, a double quote inside it would be mistaken for the closing quote, so it must be written as \"; similarly a single quote is written as \' and a backslash itself as \\. For instance printf("Escape Sequence is a \"Cool\"

feature of C."); correctly prints the word Cool inside double quotes. Escape sequences therefore give the programmer control over the layout of output and allow special characters to be displayed.

MCQs with Answers

1. Standard input and output in C mean the:
(a) mouse and printer (b) keyboard and monitor (c) disk and CD (d) scanner and speaker
Correct Answer: (b) keyboard and monitor. Keyboard and monitor.

2. The library needed for printf and scanf is:
(a) math.h (b) conio.h (c) stdio.h (d) string.h
Correct Answer: (c) stdio.h. stdio.h.

3. The function used for formatted output is:
(a) scanf (b) printf (c) getch (d) main
Correct Answer: (b) printf. printf.

4. The function used to read input from the keyboard is:
(a) printf (b) scanf (c) puts (d) clrscr
Correct Answer: (b) scanf. scanf.

5. The format specifier for an integer is:
(a) %f (b) %c (c) %d (d) %s
Correct Answer: (c) %d. %d.

6. The format specifier for a single character is:
(a) %d (b) %c (c) %f (d) %s
Correct Answer: (b) %c. %c.

7. The & in scanf is the:
(a) and operator (b) address operator (c) logical operator (d) assignment operator
Correct Answer: (b) address operator. Address operator.

8. The escape sequence for a new line is:
(a) \t (b) \b (c) \n (d) \f
Correct Answer: (c) \n. \n.

9. The escape sequence \t stands for:
(a) new line (b) tab (c) backspace (d) form feed
Correct Answer: (b) tab. Tab.

10. getch and getche belong to the library:
(a) stdio.h (b) conio.h (c) math.h (d) stdlib.h
Correct Answer: (b) conio.h. conio.h.

Quick Revision Summary

- Standard I/O = keyboard (input) and monitor (output); use #include <stdio.h>.

- printf(format string, vars...) shows formatted output; scanf(format string, &vars...) reads input.
- Specifiers: %d int, %f float/double, %c char, %s string, %lf double.
- Field width %4d and precision %6.2f control the layout of output.
- scanf needs & (address operator) before each variable; missing & gives garbage values.
- Escape sequences: \n new line, \t tab, \b backspace, \' \" \\ for quotes/backslash. Notes by freebooks.pk.

Exam Tips

- State the two standard I/O functions and the library they need.
- Explain printf with an example and list the format specifiers.
- Explain field width and precision with examples.
- Explain scanf and why the & operator is required.
- Define escape sequence and give at least four examples.
- Remember getch/getche read characters without pressing Return.

Freebooks.pk

Free Notes • Past Papers • Guides for Students

2nd Year Computer Science — Punjab Board

CHAPTER 11

Decision Constructs

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers Decision Constructs from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains control structures and the selection (decision) statements of C, the simple if, the if-else, the nested if, the if-else if ladder and the switch statement. These notes are prepared by freebooks.pk.

A program often needs to choose a path of execution based on a condition. Decision constructs, also called selection structures, let a program decide which statements to run.

Learning Objectives

- Explain the three control structures: sequence, selection and repetition.
- Define a condition and a compound statement.
- Use the simple if statement.
- Use the if-else statement.
- Use nested if and the if-else if ladder.
- Use the switch statement with case, break and default.

Key Concepts

Control Structures

Control structures are statements that control the flow of execution in a program or function. They let us group individual instructions into a single logical unit that has one entry point and one exit point. Program instructions can be organised into three kinds of control structures: sequence, in which instructions are executed in the order they are written; selection, in which the program chooses between alternatives; and repetition (also called iteration or loop), in which a statement or group of statements is repeated. Every program, simple or complex, is built from these three structures.

Conditions and Compound Statements

A condition is an expression that evaluates to either true, usually represented by 1, or false, represented by 0. Decision constructs use such conditions to choose a path. A compound statement is a group of statements enclosed in opening and closing braces { }, which are treated as a single unit. When more than one statement must be executed as part of a decision, they must be written as a compound statement inside braces; for a single statement the braces are optional.

The Simple if Statement

The if statement is the simplest decision construct. It allows a statement or a set of statements to be executed only when a condition is true. Its general form is if (condition) { statements }. If the condition is true the statements in the if block are executed; if it is false they are skipped and the program continues with the next statement. When the block contains more than one statement the braces are required; for a single statement they are optional.

The if-else Statement

The keyword else is used with if to provide two different choices. Its general form is if (condition) { true block } else { false block }. If the condition is true the if block is executed; if it is false the else block is executed. Exactly one of the two blocks runs. It is important to enclose a compound statement in braces; omitting the braces around a multi-statement if block causes the error Illegal else without matching if.

The Nested if Statement

A nested if statement is an if statement written inside another if statement, and nesting can be done to any level. It is useful when a decision depends on more than one condition, for example to find the largest of three numbers. An advantage of nested if over a sequence of separate ifs is that, once a decision is reached, the remaining conditions are skipped, whereas a sequence of ifs tests every condition. However, deep nesting increases the complexity of the code and can make it hard to read.

The if-else if Statement

When there are more than three alternatives, nested ifs become complex, so the if-else if statement (an if with multiple alternatives) is a better choice. Its conditions are tested one after another until a true condition is found; the statements following that condition are executed and the remaining alternatives are skipped. If a condition is false the next condition is tested, and if all conditions are false the statements after the final else are executed. This ladder structure is clear and easy to read for many alternatives.

The switch Statement

The switch statement selects one of many alternatives by comparing the value of an expression against a list of case labels. The expression and the case labels must be integer or character values, not float or double. The value of the expression is compared to each case; the statements after the first matching case are executed until a break statement is reached, which causes the rest of the switch to be skipped. A default label provides code to run when no case matches. If the break statements are omitted, execution falls through from the matching case to the end of the switch.

Important Definitions

Term	Definition
Control structure	A statement that controls the flow of execution in a program.
Sequence	Executing instructions in the order they are written.
Selection	Choosing which statements to execute based on a condition.
Condition	An expression that evaluates to true (1) or false (0).
Compound statement	A group of statements enclosed in braces { }.
Nested if	An if statement placed inside another if statement.

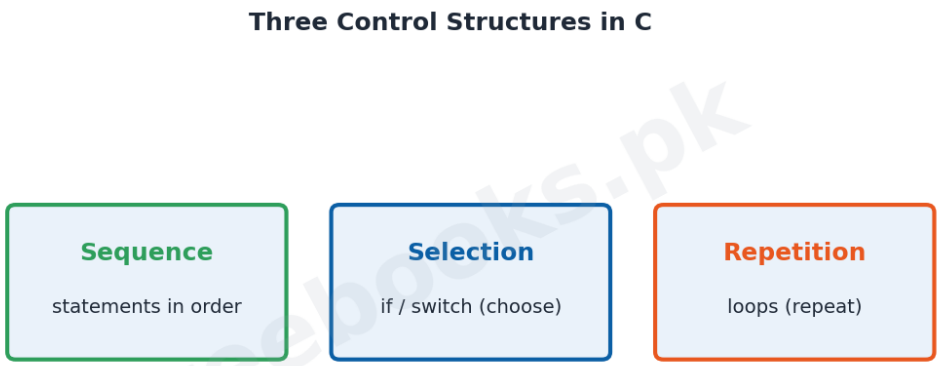
Term	Definition
switch	A selection statement that matches an expression to case labels.
default	The switch label whose code runs when no case matches.

Key Facts & Rules

Item	Detail
Simple if	if (condition) { statements }
if-else	if (condition) { true } else { false }
if-else if	if (c1){} else if (c2){} else {}
switch	switch(expr){ case v1: ... break; default: ... }
Condition value	true = 1, false = 0
case labels	must be integer or character constants
break	ends a case and exits the switch

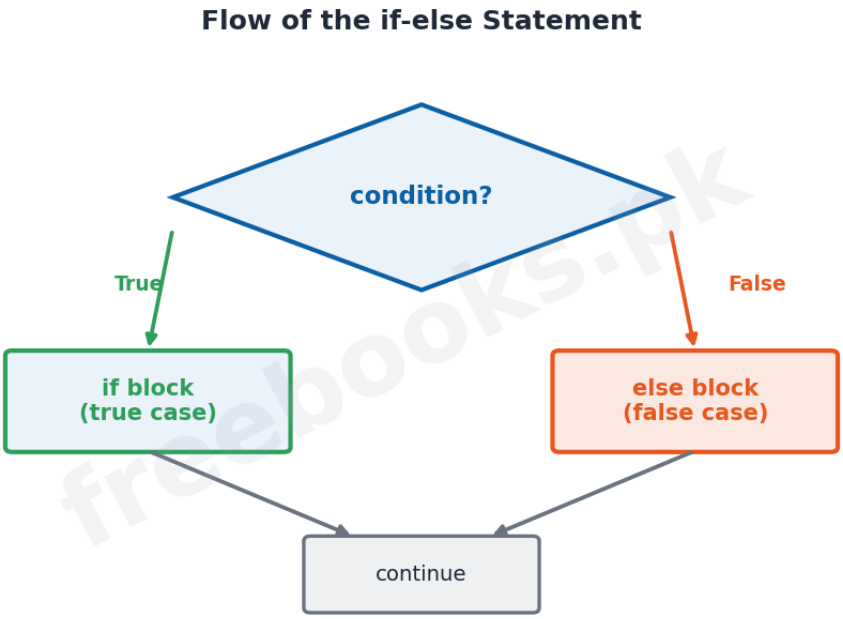
Diagrams & Illustrations

Three control structures: the three control structures used in every program: sequence, selection and repetition.



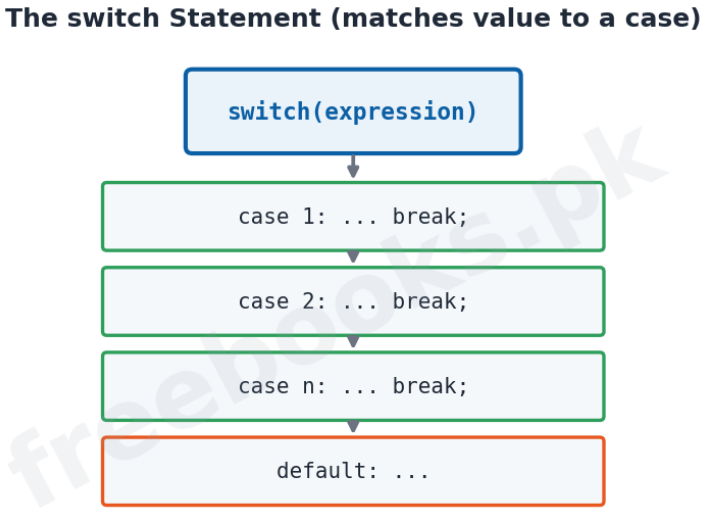
Three control structures

Flow of the if-else statement: how an if-else statement chooses the true block or the false block depending on the condition.



Flow of the if-else statement

The switch statement: how a switch statement matches the expression value to a case and uses break and default.



The switch statement

Solved Examples & Practice

Simple if

if (x > 0) square_root = sqrt(x); calculates the square root only when x is positive.

if-else

if (x >= 0) printf("root exists"); else printf("cannot be calculated"); runs one block or the other.

Nested if for largest

Using if (a > b) then if (a > c) inside it, the program finds the largest of three numbers a, b and c.

switch on a grade

switch(grade){ case 'A': printf("Excellent"); break; default: printf("Try again"); } selects a message by value.

Short Questions & Answers

Q1. Name the three control structures.
Ans. Sequence, selection and repetition (iteration).

Q2. What is a condition?
Ans. An expression that evaluates to true (1) or false (0).

Q3. What is a compound statement?
Ans. A group of statements enclosed in braces { } treated as a single unit.

Q4. What is a nested if statement?
Ans. An if statement written inside another if statement.

Q5. What is the use of the break statement in switch?
Ans. It ends the current case and skips the rest of the switch statement.

Q6. What is the use of the default label in switch?
Ans. It provides code that runs when none of the case labels matches.

Long Questions & Answers

Q1: Explain the simple if, if-else and nested if statements with their general forms.

Decision constructs, also called selection structures, allow a program to choose which statements to execute according to a condition, where a condition is an expression that evaluates to true, represented by 1, or false, represented by 0. The simplest of these is the simple if statement, whose general form is if (condition) followed by a block of statements. The statements in the block are executed only when the condition is true; if the condition is false they are skipped and the program continues with the next statement. When the block contains more than one statement the statements must be enclosed in braces to form a compound statement, although for a single statement the braces may be omitted. The if-else statement extends this idea by using the keyword else to provide a second choice; its

general form is if (condition) { true block } else { false block }. When the condition is true the if block is executed, and when it is false the else block is executed, so that exactly one of the two blocks always runs. It is important to enclose any compound statement in braces, because omitting the braces around a multi-statement if block produces the error message Illegal else without matching if. The nested if statement is an if statement written inside another if statement, and nesting may be carried to any level. It is used when a decision depends on more than one condition, for example when finding the largest of three numbers by testing a against b and then, inside that, a against c. A useful property of the nested if is that once a decision is reached the remaining conditions are skipped, unlike a sequence of separate if statements in which every condition is tested; however, deep nesting makes the code more complex and harder to follow.

Q2: Explain the if-else if statement and compare it with the switch statement.

When a program must choose among more than three alternatives, a nested if statement can become complex and difficult to read, so C provides the if-else if statement, an if statement with multiple alternatives. Its general form is a ladder in which the first condition is tested and, if true, its statements are executed while the rest of the ladder is skipped; if the first condition is false the next else if condition is tested, and so on. If all the conditions are false, the statements following the final else are executed. The conditions are therefore tested in sequence until a true one is found, which makes the structure clear even when there are many alternatives. The switch statement is another way to select one of many alternatives, but it works differently: instead of testing a series of conditions, it compares the value of a single expression against a list of case labels. The expression and the case labels must be of integer or character type; float and double values are not allowed. The value of the expression is compared with each case, and the statements after the first matching case are executed until a break statement is reached, which causes the rest of the switch to be skipped. A default label may be included to provide statements that run when no case matches. If the break statements are omitted, control falls through from the matching case down to the end of the switch, executing all the statements below it. In short, the if-else if ladder is best when the choice depends on ranges or complex conditions, while the switch statement is especially convenient when the selection is based on the exact value of a single integer or character expression.

Q3: Explain the switch statement in detail with its syntax, the role of break and default, and its limitations.

The switch statement is a multi-way selection statement in C that chooses one of many alternatives by comparing the value of an expression against a list of possible values. Its syntax begins with the keyword switch followed by an expression in parentheses, and then a block containing several case labels, each written as case value: followed by one or more statements and usually a break statement, and optionally a default label. When the switch statement runs, the value of the expression is evaluated once and then compared to each case label in turn. The statements following the first case label whose value matches the expression are executed. Execution continues until a break statement is encountered; the

break causes the rest of the switch statement to be skipped so that control passes to the statement after the closing brace of the switch. If the break statements are omitted, then once a matching case is found the program continues executing every statement from that point down to the end of the switch, a behaviour known as fall-through, which is occasionally useful, for example when several case labels such as 'a', 'A' should run the same code. The default label provides statements to be executed when none of the case labels matches the value of the expression; its position is not fixed and it may be written before the first case or after the last case. The switch statement has an important limitation: the case labels must be integral or character constants and the controlling expression must evaluate to an integer or a character, so it cannot be used with float or double values or with ranges of values. Because of this, the switch statement is especially useful when the selection is based on the exact value of a single integer or character expression, while decisions that involve ranges or complex conditions are better handled by an if-else if ladder.

MCQs with Answers

1. The three control structures are sequence, selection and:
(a) compilation (b) repetition (c) declaration (d) execution
Correct Answer: (b) repetition. Repetition.

2. A condition evaluates to:
(a) true or false (b) only true (c) only false (d) a string
Correct Answer: (a) true or false. True or false.

3. A group of statements in braces { } is a:
(a) function (b) compound statement (c) header (d) loop
Correct Answer: (b) compound statement. Compound statement.

4. The simplest decision construct is the:
(a) switch (b) for loop (c) if statement (d) while loop
Correct Answer: (c) if statement. if statement.

5. The keyword used for the false case with if is:
(a) then (b) else (c) break (d) case
Correct Answer: (b) else. else.

6. An if statement inside another if is called:
(a) chained if (b) nested if (c) switch (d) default
Correct Answer: (b) nested if. Nested if.

7. The switch case labels must be:
(a) float or double (b) integer or character constants (c) strings only (d) any value
Correct Answer: (b) integer or character constants. Integer or character constants.

8. The statement that ends a case in switch is:
(a) stop (b) return (c) break (d) exit
Correct Answer: (c) break. break.

9. The label that runs when no case matches is:

(a) case (b) else (c) default (d) final
Correct Answer: (c) default. default.

10. Omitting braces around a multi-statement if block causes:
(a) no error (b) Illegal else without matching if (c) syntax stop (d) infinite loop
Correct Answer: (b) Illegal else without matching if. Illegal else without matching if.

Quick Revision Summary

- Three control structures: sequence, selection, repetition (loops).
- Condition = expression that is true (1) or false (0); compound statement = statements in { }.
- Simple if runs a block only if true; if-else runs one of two blocks.
- Nested if = if inside if; if-else if ladder tests conditions until one is true.
- switch matches an integer/char expression to case labels; break exits, default = no match.
- Use if-else if for ranges/complex tests, switch for exact single values. Notes by freebooks.pk.

Exam Tips

- List and define the three control structures.
- Give the general form of simple if, if-else and nested if.
- Explain the if-else if ladder with how conditions are tested.
- Give the syntax of switch and explain break and default.
- State the limitation on switch case labels (integer/char only).
- Know the error message for a missing brace: Illegal else without matching if.

2nd Year Computer Science — Punjab Board

CHAPTER 12

Loop Constructs

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers Loop Constructs from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains iteration and the three loop statements of C, the while loop, the do-while loop and the for loop, together with nested loops, sentinel-controlled loops and type casting. These notes are prepared by freebooks.pk.

Many problems require a statement or set of statements to be executed repeatedly. A loop is a control structure that repeats statements, either a fixed number of times or until a condition is met.

Learning Objectives

- Explain iteration (looping) as the third control structure.
- Use the while loop.
- Use the do-while loop and know why it runs at least once.
- Use the for loop with its three expressions.
- Explain nested loops.
- Explain sentinel-controlled loops and type casting.

Key Concepts

Iteration and the Loop Control Variable

Iteration is the third type of program control structure, after sequence and selection, and the repetition of statements in a program is called a loop. C provides three loop statements: while, do-while and for. Each loop is controlled by a condition and usually by a loop control variable, which is a variable whose value controls the number of iterations. The group of statements repeated by the loop, enclosed in braces, is called the body of the loop.

The while Loop

The while loop keeps repeating its associated statements as long as a specified condition remains true; it is useful when the programmer does not know in advance how many times the loop will run. Its syntax is while (condition) followed by the body. When the condition is true the body is executed, and as soon as the condition becomes false the loop terminates. In a while loop the loop control variable is initialised outside the loop and is incremented or decremented inside the loop body.

The do-while Loop

The do-while loop is similar to the while loop except that the condition is tested at the end of the loop body, which guarantees that the body is executed at least once. Its syntax begins with do, then the body, and ends with while (condition); note that the do-while statement ends with a semicolon, and omitting it causes a syntax error. This loop is useful when statements must run at least once, for example when reading and validating user input until a valid value is entered.

while vs do-while

The key difference is when the condition is tested. In a while loop the condition is tested first, so the body may execute zero times if the condition is false at the start. In a do-while loop the body is executed first and the condition is tested afterwards, so the body always executes at least once. Therefore a while loop is used when the body should run only if the condition is true, while a do-while loop is used when the body must run at least once.

The for Loop

The for loop is the most flexible loop and is preferred by most programmers. Its syntax is for (initialization; test condition; increment/decrement) followed by the body. The three expressions are separated by semicolons: the initialization sets the loop control variable and runs only in the first iteration; the test condition is checked before each pass; and the increment/decrement changes the loop control variable after each pass. The initialization and the increment/decrement are optional, but the test condition is mandatory and cannot be omitted.

Nested Loops

A nested loop is a loop placed inside the body of another loop, and nesting can be done to any level, though complexity increases with each level. There is no restriction on the types of loops that may be nested, so a while, do-while or for loop can be placed inside any other loop. In a nested loop the inner loop completes all its iterations for each single iteration of the outer loop, and each time the outer loop starts a new iteration, the variables used in the inner loop are re-initialised and re-processed.

Sentinel-Controlled Loops

Often a program must read a list of items whose length is not known in advance, for example the marks of every student in a class. A sentinel-controlled loop handles this by asking the user to enter a special value, called a sentinel value, after the last data item. The loop tests each item and exits when the sentinel is read. The sentinel value must be chosen carefully so that it can never occur as normal data; for example, a negative number can be a sentinel for marks because marks are never negative.

Type Casting

Type casting is temporarily treating a variable of one type as another type for a single calculation. It is often needed with division: when two integers are divided the result is an integer and the fractional part is truncated, giving an inaccurate answer. Writing a type in parentheses before a variable, as in (float)total_students, causes that integer variable to act as a float for that one calculation, preserving the fractional part, while it continues to behave as an integer everywhere else.

Important Definitions

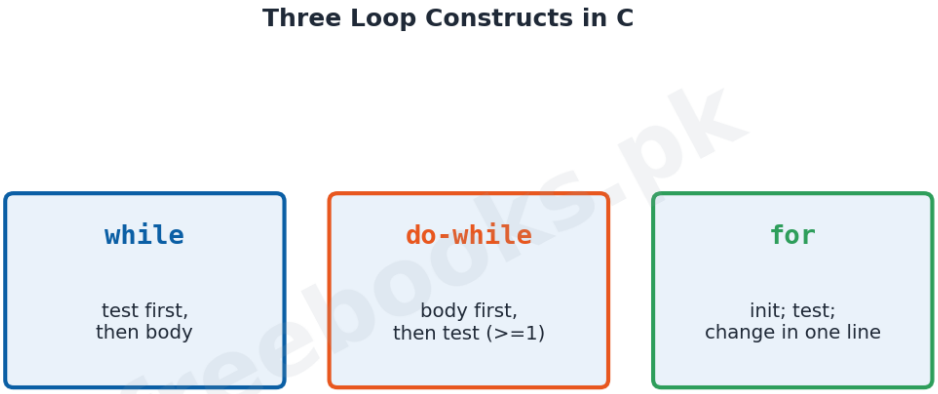
Term	Definition
Iteration (loop)	The repetition of a statement or group of statements in a program.
Loop control variable	A variable whose value controls the number of iterations.
Body of the loop	The statements repeated by the loop, enclosed in braces.
while loop	A loop that tests the condition before executing the body.
do-while loop	A loop that executes the body first, then tests the condition (runs at least once).
for loop	A loop with initialization, test condition and increment/decrement in one statement.
Nested loop	A loop placed inside the body of another loop.
Sentinel value	A special value entered to signal the end of input data.

Key Facts & Rules

Item	Detail
while	while (condition) { body }
do-while	do { body } while (condition); (ends with ;)
for	for (init; test; change) { body }
for optional parts	init and change optional; test is mandatory
Loop control variable	initialised, tested and changed to control iterations
Sentinel loop	read until a special value (e.g. a negative number)
Type cast	(float)total_students keeps the fractional part

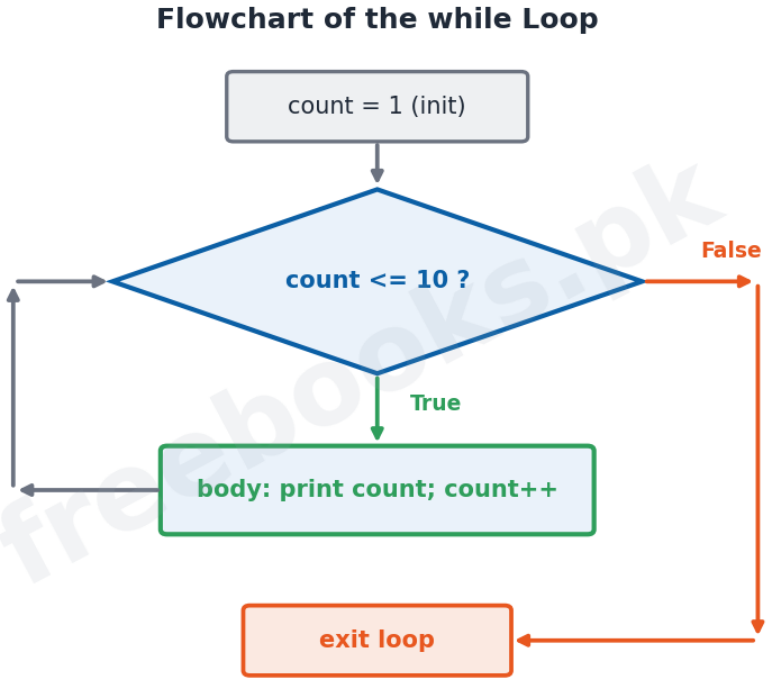
Diagrams & Illustrations

Three loop constructs: the three loop constructs of C, while, do-while and for, and how each tests its condition.



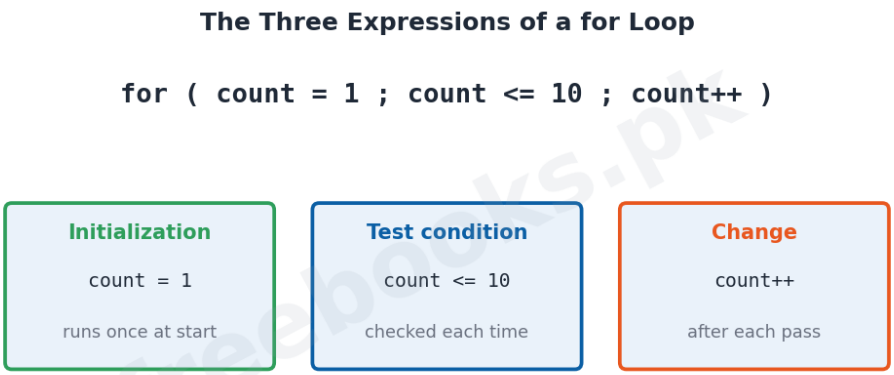
Three loop constructs

Flowchart of the while loop: how a while loop initialises, tests the condition, executes the body and exits when false.



Flowchart of the while loop

The three expressions of a for loop: the initialization, test condition and increment/ decrement parts of a for loop.



The three expressions of a for loop

Solved Examples & Practice

Print 1 to 10 (while)

Initialise count = 1, then while (count <= 10) print count and do count++; the loop prints 1 to 10.

Validate input (do-while)

do { read state } while (state != 'w' && state != 'd'); forces the user to enter a valid value.

Print 1 to 10 (for)

for (count = 1; count <= 10; count++) printf("%d", count); prints 1 to 10 in one compact statement.

Average with sentinel

Read marks in a while loop until a negative number is entered, summing them, then divide using (float) to get the average.

Short Questions & Answers

Q1. What is a loop?
Ans. A control structure that repeats a statement or group of statements in a program.

Q2. What is a loop control variable?
Ans. A variable whose value controls how many times the loop runs.

Q3. Why does a do-while loop always run at least once?
Ans. Because its condition is tested at the end, after the body has already executed once.

Q4. Name the three expressions of a for loop.
Ans. Initialization, test condition and increment/decrement.

Q5. What is a nested loop?

Ans. A loop placed inside the body of another loop.

Q6. What is a sentinel value?

Ans. A special value entered after the last data item to signal the loop to stop.

Long Questions & Answers

Q1: Explain the while loop and the do-while loop, and state the difference between them.

Iteration, the repetition of statements, is the third program control structure after sequence and selection, and C provides three loop statements to carry it out. The while loop keeps repeating its body as long as a specified condition remains true, and its syntax is the keyword while followed by a condition in parentheses and then the body of the loop. It is especially useful when the programmer does not know in advance how many times the loop will run. When execution reaches the while statement the condition is tested; if it is true the body is executed and control returns to test the condition again, and as soon as the condition becomes false the loop terminates and control passes to the statement after the loop. In a while loop the loop control variable, the variable whose value controls the number of iterations, is initialised outside the loop and is incremented or decremented inside the body, for example a variable count initialised to 1 and increased by one each time until the condition count <= 10 becomes false. The do-while loop is very similar but tests its condition at the end of the body rather than at the beginning; its syntax begins with the keyword do, then the body, and ends with while (condition) followed by a semicolon, and forgetting this semicolon causes a syntax error. Because the condition is tested after the body, a do-while loop always executes its body at least once, which makes it ideal for tasks such as reading and validating input, where the program must obtain a value at least once and keep asking until the value is valid. The essential difference between the two is therefore the timing of the test: in a while loop the condition is tested first, so the body may run zero times, whereas in a do-while loop the body runs first and is guaranteed to execute at least once.

Q2: Explain the for loop with its three expressions and describe nested loops.

The for loop is another way of writing a loop in C and, because of its flexibility, it is preferred by most programmers for counting loops. Its syntax is the keyword for followed by three expressions in parentheses, separated by semicolons, and then the body: for (initialization expression; test condition; increment/decrement expression). The initialization expression sets the starting value of the loop control variable and is executed only once, during the first iteration. The test condition is then checked; if it is true the body of the loop is executed, after which the increment/decrement expression is evaluated to change the loop control variable, and the condition is tested again. This cycle of test, execute body, and change continues as long as the condition is true, and when the condition becomes false the loop terminates and control passes to the next statement. For example, for (count = 1; count <= 10; count++) prints the numbers 1 to 10. Two of the three expressions, the initialization and

the increment/decrement, are optional and may be omitted, but the test condition is mandatory; if the initialization is omitted the loop control variable must be initialised before the loop and changed inside the body. A nested loop is a loop placed inside the body of another loop, and nesting may be carried to any level, although the complexity grows with each level. Any type of loop may be nested inside any other, so a while, do-while or for loop can appear inside a for loop and so on. In a nested loop the inner loop runs through all its iterations for every single iteration of the outer loop, and each time the outer loop begins a new iteration the variables of the inner loop are re-initialised. Nested loops are commonly used to produce two-dimensional patterns, such as printing rows of asterisks, where the outer loop controls the rows and the inner loop controls the items in each row.

Q3: What is a sentinel-controlled loop? Explain it with the idea of type casting.

Many programs need to read and process a list of items whose length is not known in advance; for instance, to find the average marks of a class the program must read the marks of every student, but the number of students may vary. A sentinel-controlled loop solves this problem by asking the user to enter a special value, called a sentinel value, after the last real data item. The loop reads and processes each item, and the loop condition tests every value so that the loop exits as soon as the sentinel value is read. The sentinel value must be chosen carefully so that it can never occur as valid data: for reading marks, a negative number is a good sentinel because a student can never score negative marks, whereas zero would be a poor choice since a student may score zero. In a typical average program the loop keeps reading marks and adding them to a running sum while counting the students, and stops when a negative number is entered. Before dividing to find the average, the program should check that at least one student was entered, because dividing by a total of zero would cause a runtime division-by-zero error. The average is then found by dividing the sum by the number of students, and here type casting becomes important. Since both the sum and the count are integer variables, their division would be an integer division in which the fractional part is truncated, giving an inaccurate result. To avoid this, the type float is written in parentheses before the count, as in (float)total_students, which causes that integer variable to act temporarily as a float for that one calculation so that the fractional part of the answer is preserved; the variable continues to behave as an integer in all other calculations. This temporary change of type for a single calculation is called type casting.

MCQs with Answers

1. The repetition of statements in a program is called a:
(a) branch (b) loop (c) function (d) header
Correct Answer: (b) loop. Loop.

2. The three loop statements in C are while, do-while and:
(a) switch (b) for (c) if (d) case
Correct Answer: (b) for. for.

3. The loop that tests its condition at the end is the:

(a) while (b) for (c) do-while (d) nested
Correct Answer: (c) do-while. do-while loop.

4. A do-while loop executes its body at least:
(a) zero times (b) once (c) twice (d) ten times
Correct Answer: (b) once. Once.

5. The do-while statement ends with a:
(a) colon (b) semicolon (c) comma (d) brace
Correct Answer: (b) semicolon. Semicolon.

6. In a for loop, the part that runs only once is the:
(a) test condition (b) initialization (c) increment (d) body
Correct Answer: (b) initialization. Initialization.

7. Which expression of a for loop is mandatory?
(a) initialization (b) test condition (c) increment (d) none
Correct Answer: (b) test condition. Test condition.

8. A loop inside another loop is called a:
(a) chained loop (b) nested loop (c) switch (d) sentinel
Correct Answer: (b) nested loop. Nested loop.

9. A special value that signals the end of input is a:
(a) counter (b) sentinel value (c) flag (d) token
Correct Answer: (b) sentinel value. Sentinel value.

10. (float)total_students in an average formula is an example of:
(a) assignment (b) type casting (c) declaration (d) looping
Correct Answer: (b) type casting. Type casting.

Quick Revision Summary

- Iteration (loop) = repetition of statements; C has while, do-while and for loops.
- while tests condition first (may run 0 times); do-while tests at end (runs at least once, ends with ;).
- for (init; test; change) — init runs once, test mandatory, init and change optional.
- Nested loop = loop inside a loop; inner loop completes fully for each outer iteration.
- Sentinel-controlled loop reads until a special value (e.g. a negative number) that can't be real data.
- Type casting: (float)var keeps the fractional part in integer division. Notes by freebooks.pk.

Exam Tips

- Explain iteration and name the three loops.
- Give the syntax of while and do-while and their difference.
- Explain the three expressions of the for loop; note which are optional.
- Define a nested loop and describe how inner/outer loops run.

- Explain a sentinel-controlled loop with a suitable sentinel value.
- Explain type casting with the (float) example and division by zero.

Freebooks.pk

Free Notes • Past Papers • Guides for Students

2nd Year Computer Science — Punjab Board

CHAPTER 13

Functions in C

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers Functions in C from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains structured (modular) programming, the importance and types of functions, the function header, body and return statement, function prototypes, calling a function, and the scope of local and global variables. These notes are prepared by freebooks.pk.

Functions are the building blocks of C programs. A function is a self-contained piece of code with a specific purpose, and dividing a program into functions is the basis of structured programming.

Learning Objectives

- Explain structured (modular) programming and the importance of functions.
- Distinguish built-in and user-defined functions.
- Describe the function header, body and return statement.
- Explain the function prototype.
- Explain how a function is called.
- Explain local and global variables and their scope.

Key Concepts

Structured Programming and Functions

Functions are the building blocks of C programs; a function is a self-contained piece of code with a specific purpose. When the whole program logic is placed inside a single main function, the style is called unstructured programming. Structured programming is a modular approach in which the program logic is divided into a number of smaller modules or functions, and the main function calls these functions where they are needed. This makes programs easier to write, read and maintain.

Importance of Functions

Functions provide many benefits. They make programs easier to understand and maintain, because the main program becomes a series of function calls instead of countless lines of code. They increase the reusability of code, since well-written functions can be used in many programs, as the C standard library shows. They allow different programmers to divide a large project by writing different functions, enabling parallel development. A function can be executed as many times as needed from different places, and when an error occurs only the affected function has to be debugged rather than the whole program.

Types of Functions

There are two types of functions in C: built-in functions and user-defined functions. Built-in (library) functions are predefined functions packaged in libraries that provide ready-made functionality, such as printf and scanf in stdio.h, or getch and getche in conio.h; to use one, the header file containing its declaration must be included. User-defined functions are

functions that the programmer writes, according to the nature of the problem, when the built-in functions are not sufficient.

The Function Header

Every function has a function header, which is its first line and identifies the function. Its general form is return_type FunctionName (parameter_list). The header states three things: the type of the return value, the name of the function, and the parameters enclosed in parentheses. The return_type may be any valid data type; if the function returns no value the return type is the keyword void, and a function with no parameters may write void, or leave the parentheses empty, as its parameter list.

The Function Body and return Statement

The function body follows the header and is enclosed in curly braces; it contains the variable declarations and the program logic and makes use of the arguments passed to the function. The return statement, written as return expression;, specifies the value returned by the function. When it is executed the expression is evaluated and returned, and the function stops immediately even if statements remain. If the return type is void, no return statement is needed.

Function Prototype

The compiler must know about a function before it is used, which is why header files are included for built-in functions. A function prototype is a statement that gives the compiler the basic information it needs to check and use a function correctly: the return type, the function name and the parameters. Its general form is the same as the function header but with a semicolon at the end, return_type FunctionName (parameter_list);. Prototypes for functions must appear before the function call, and are usually placed at the top of the source file just before the main function.

Calling a Function

A function call is the mechanism used to invoke a function to perform its task, and it can be made at any point in the program. To call a function, the function name, the required arguments and the statement terminator (a semicolon) are written. When a function call is executed, control is transferred to the called function, memory is allocated to the variables declared in it, and the statements in its body are executed; after the last statement, control returns to the calling function.

Local and Global Variables and Scope

The scope of a variable is the region of a program in which it can be accessed; a variable cannot be used outside its scope, and doing so causes a compiler error. Variables declared inside a block (a pair of curly braces) are local variables with local scope, valid from their declaration to the end of that block; their lifetime is the period during which control stays in that block. A global variable is declared outside all functions and has global scope, so it is accessible to the whole program from its declaration onwards.

Important Definitions

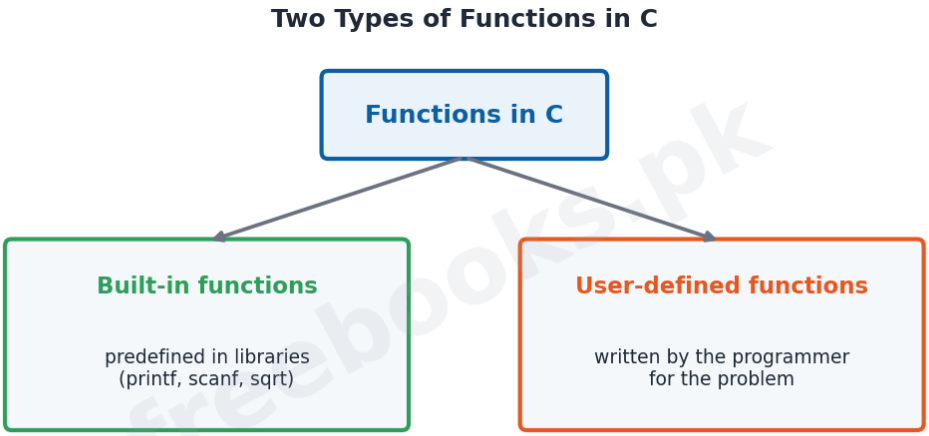
Term	Definition
Function	A self-contained piece of code that performs a specific task.
Structured programming	Dividing a program into smaller modules or functions.
Built-in function	A predefined function packaged in a library (e.g. printf).
User-defined function	A function written by the programmer for a specific problem.
Function header	The first line stating return type, name and parameters.
return statement	A statement that returns a value and ends the function.
Function prototype	A declaration of a function's return type, name and parameters, ending with a semicolon.
Scope	The region of a program in which a variable can be accessed.

Key Facts & Rules

Item	Detail
Function definition	return_type FunctionName(parameter_list) { body }
No value / no parameter	void FunctionName(void) or void FunctionName()
return statement	return expression;
Function prototype	return_type FunctionName(parameter_list); (ends with ;)
Function call	FunctionName(arguments);
Local variable	declared in a block; scope = that block only
Global variable	declared outside all functions; scope = whole program

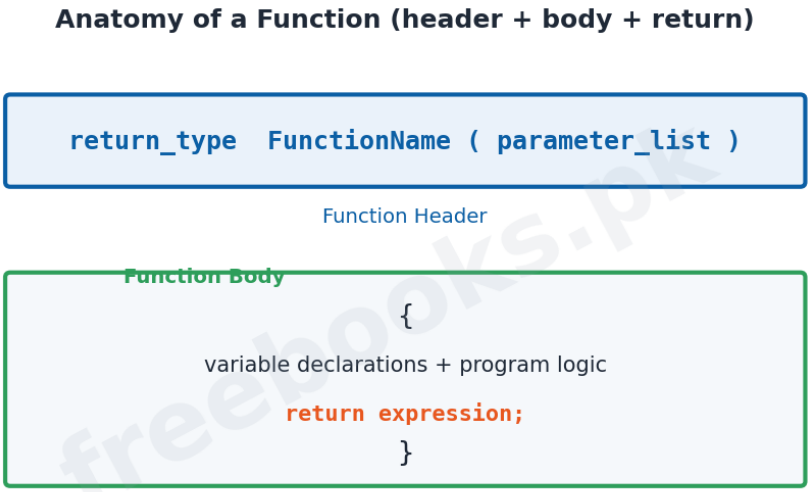
Diagrams & Illustrations

Two types of functions: the two types of functions in C, built-in (library) functions and user-defined functions.



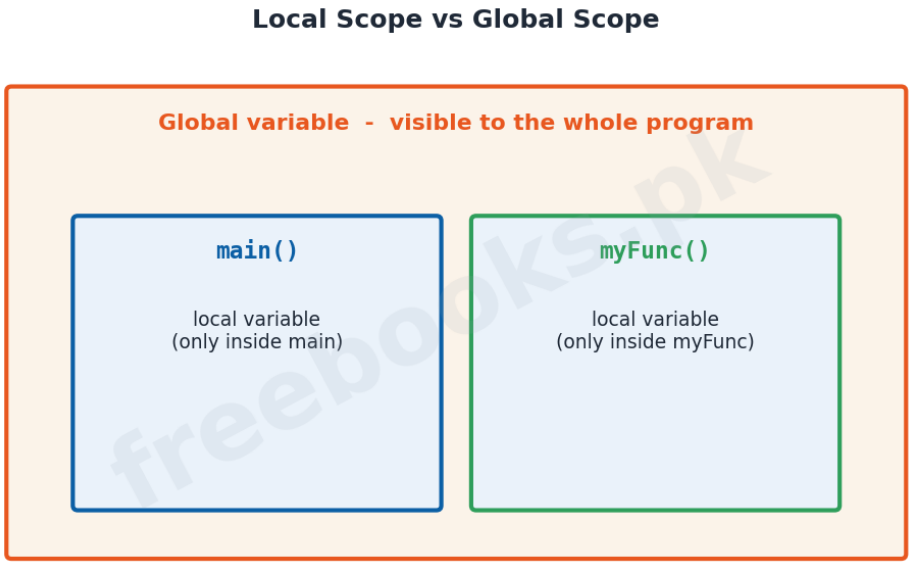
Two types of functions

Anatomy of a function: the parts of a function, the header and the body with its return statement.



Anatomy of a function

Local vs global scope: how a local variable is visible only in its block while a global variable is visible to the whole program.



Local vs global scope

Solved Examples & Practice

A function header

int add(int a, int b) is a header for a function named add that takes two integers and returns an integer.

A void function

void greet(void) is a function that takes no parameters and returns no value.

A prototype

int add(int, int); written before main is the prototype of the add function.

A function call

sum = add(5, 3); calls add with arguments 5 and 3 and stores the returned value in sum.

Short Questions & Answers

Q1. What is a function?
Ans. A self-contained piece of code that performs a specific task.

Q2. Name the two types of functions in C.
Ans. Built-in (library) functions and user-defined functions.

Q3. What is a function header?
Ans. The first line of a function stating its return type, name and parameters.

Q4. What is the use of the return statement?

Ans. It returns a value from the function and stops the function's execution.

Q5. What is a function prototype?

Ans. A statement giving the compiler the return type, name and parameters of a function, ending with a semicolon.

Q6. What is the scope of a local variable?

Ans. From its declaration to the end of the block in which it is declared.

Long Questions & Answers

Q1: Explain structured programming, the importance of functions and the two types of functions.

A function is a self-contained piece of code with a specific purpose, and functions are the building blocks of C programs. In the simplest programs the whole logic is placed inside a single main function, a style known as unstructured programming; structured programming, by contrast, is a modular approach in which the program logic is divided into a number of smaller modules called functions, and the main function calls these functions wherever they are needed. This idea is inspired by hardware manufacturing, where a device is built from replaceable components that can be combined and, if faulty, replaced individually. Dividing a program into functions brings several important benefits. First, it makes programs much easier to understand and maintain, because the main program becomes a short series of function calls rather than countless lines of code. Second, it increases the reusability of code, since a well-written function can be used again in many programs, and the C standard library is itself a large collection of reusable functions. Third, it supports parallel development, because different programmers working on one large project can divide the work by each writing different functions. Fourth, a function can be executed as many times as necessary from different places in the program, avoiding repeated code. Finally, it simplifies debugging, because when an error arises only the affected function needs to be examined rather than the whole program. There are two types of functions in C. Built-in or library functions are predefined functions packaged in libraries, such as printf and scanf in the standard input/output library and getch and getche in the console input/output library; to use one, the header file that declares it must be included in the program. User-defined functions are functions that the programmer writes when the built-in functions are not sufficient for the problem at hand.

Q2: Describe the structure of a function in C: the header, the body, the return statement and the function prototype.

Every function in C has almost the same basic structure, consisting of a function header followed by the function body enclosed in curly braces. The function header is the first line of the definition and identifies the function; its general form is return_type FunctionName (parameter_list), and it states three things: the type of the value the function returns, the name of the function, and the parameters of the function enclosed in parentheses. The return type may be any valid data type, but if the function does not return a value the return type is written as the keyword void, and a function that takes no parameters may write void

as its parameter list or simply leave the parentheses empty, so that a function returning nothing and taking nothing may be written void FunctionName(). The function body follows the header and is enclosed in curly braces; it contains the declarations of the variables used by the function and the program logic, and it makes use of the arguments passed to the function. Inside the body the return statement, whose general form is return expression;, is used to specify the value the function sends back to the caller; when it is executed the expression is evaluated and returned as the value of the function, and the function stops immediately even if further statements remain, while a void function needs no return statement. Because the compiler must know about a function before it is used, C also requires a function prototype, which is a statement that provides the compiler with the return type, the name and the parameter list of the function. The prototype looks exactly like the function header but ends with a semicolon, as in return_type FunctionName (parameter_list);, and it must appear before the function is called, usually at the top of the source file just before the main function.

Q3: Explain how a function is called, and describe local and global variables and their scope.

A function call is the mechanism used to invoke a function so that it performs its task, and a call can be made at any point in the program. To call a function, the programmer writes the function name, the arguments the function requires, and the statement terminator, which is a semicolon. When a function call statement is executed, control is transferred from the calling function to the called function; memory is then allocated to the variables declared inside the called function, and the statements in its body are executed one by one. After the last statement of the function, or when a return statement is reached, control returns to the calling function, which continues from where it left off. Closely related to functions is the idea of the scope of a variable, which is the region of a program in which the variable can be accessed; the name of a variable is only valid within its scope, and any attempt to refer to it outside its scope causes a compiler error. Variables declared inside a block, that is within a pair of curly braces such as the body of a function or of an if statement, are called local variables and have local scope, which extends from the point where the variable is declared to the end of the block containing the declaration. The lifetime of a local variable is the period during which control remains inside that block; as soon as control leaves the block the variable is destroyed, meaning the memory allocated to it is returned to the operating system. A global variable, in contrast, is declared outside all functions and has global scope, so it is accessible to the whole program from the point of its declaration onwards. Local variables keep functions independent of one another, while global variables allow information to be shared across the whole program.

MCQs with Answers

1. The building blocks of C programs are:
(a) loops (b) functions (c) arrays (d) headers
Correct Answer: (b) functions. Functions.

2. Dividing a program into smaller modules is called:

(a) unstructured programming (b) structured programming (c) looping (d) casting
Correct Answer: (b) structured programming. Structured programming.

3. printf and scanf are examples of:
(a) user-defined functions (b) built-in functions (c) global variables (d) prototypes
Correct Answer: (b) built-in functions. Built-in functions.

4. A function written by the programmer is a:
(a) built-in function (b) user-defined function (c) library function (d) header
Correct Answer: (b) user-defined function. User-defined function.

5. If a function returns no value its return type is:
(a) int (b) void (c) float (d) char
Correct Answer: (b) void. void.

6. The statement that returns a value from a function is:
(a) break (b) return (c) exit (d) stop
Correct Answer: (b) return. return.

7. A function prototype ends with a:
(a) brace (b) semicolon (c) comma (d) colon
Correct Answer: (b) semicolon. Semicolon.

8. A variable declared inside a block is a:
(a) global variable (b) local variable (c) constant (d) parameter
Correct Answer: (b) local variable. Local variable.

9. A variable accessible to the whole program is a:
(a) local variable (b) global variable (c) temporary variable (d) register
Correct Answer: (b) global variable. Global variable.

10. Using a variable outside its scope causes a:
(a) warning only (b) compiler error (c) new variable (d) loop
Correct Answer: (b) compiler error. Compiler error.

Quick Revision Summary

- Function = self-contained code with a specific purpose; building block of C programs.
- Structured programming divides logic into functions called by main.
- Two types: built-in (library, e.g. printf) and user-defined functions.
- Function = header (return_type name(params)) + body { ... return expr; }.
- Prototype = header ending with a semicolon, placed before the call.
- Local variable: scope = its block; global variable: scope = whole program. Notes by freebooks.pk.

Exam Tips

- Define a function and explain structured programming.
- List the benefits (importance) of functions.

- Differentiate built-in and user-defined functions with examples.
- Describe the header, body and return statement.
- Explain the function prototype and where it is placed.
- Differentiate local and global variables by their scope.

Freebooks.pk

Free Notes • Past Papers • Guides for Students

2nd Year Computer Science — Punjab Board

CHAPTER 14

File Handling in C

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers File Handling in C from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains why files are needed, the concept of a stream, text and binary streams, the newline and EOF markers, opening and closing files, file opening modes, the file pointer, and reading and writing characters. These notes are prepared by freebooks.pk.

Programs so far worked with temporary data that had to be re-entered each time. To store data permanently we use files. A file is a set of related records saved on secondary storage such as a disk.

Learning Objectives

- Explain the need for files and define a file.
- Explain the concept of a stream and its two types.
- Describe the newline and EOF markers in a text file.
- Open and close files using fopen() and fclose().
- State the file opening modes.
- Explain the file pointer and read/write characters with getc() and putc().

Key Concepts

The Need for Files

The programs studied so far worked only with temporary data, so the user had to enter the data every time the program ran, and no data or results could be kept permanently. To store data permanently it is saved on secondary storage in the form of files. A file is a set of related records. File handling gives a C program the ability to save its data to a file and to read the data back later, so that information is not lost when the program ends.

The Stream

C has no built-in method of performing file input/output, but the C standard library stdio provides a rich set of I/O functions for the purpose. A very important concept is the stream, which is a logical interface to a file. A stream is associated with a file by an open operation and disassociated from it by a close operation. Using streams lets C treat many different devices, such as a disk file, the screen, the keyboard or a port, in a uniform way.

Text and Binary Streams

There are two types of streams. A text stream is a sequence of characters in which certain translations may occur, for example a newline may be converted to a carriage-return/line-feed pair, so there may not be a one-to-one match between the characters written and those stored on the device. A binary stream is a sequence of bytes that correspond one-to-one with the bytes on the external device, with no translation, so the number of bytes read or written is the same as the number on the device.

Newline and EOF Marker

A text file is a named collection of characters saved in secondary storage and has no fixed size. To mark the end of a text file a special end-of-file character, denoted by EOF in C, is placed after the last character. When a text file is created with an editor such as Notepad, pressing the Enter key places a newline character, denoted by \n, at the end of each line, and an EOF marker is placed at the very end of the file.

Opening a File

Before reading from or writing to a file it must be opened. All standard file handling functions are declared in stdio.h, so this header is included in almost every such program. The fopen() function opens a file and associates it with a stream; its prototype is FILE* fopen(const char* filename, const char* mode). Its first argument is the file name (with the absolute path, using \\ for each backslash, if the file is not in the current directory) and its second argument is the mode written as a string in double quotes. If fopen() fails, for example because the file does not exist, it returns the NULL pointer, so the result should always be checked.

File Opening Modes

A file can be opened in several modes. The mode "r" opens a text file for reading and the file must already exist; "w" opens a file for writing, overwriting an existing file or creating a new one; and "a" opens a file for appending, adding data to the end of an existing file or creating it. The combined modes "r+", "w+" and "a+" open a file for both reading and writing, differing in whether the file must exist and whether its existing contents are kept or overwritten.

The File Pointer

A file pointer is a variable of type FILE, defined in stdio.h, and is declared as FILE* fp. Here the symbol * does not mean multiplication; used with a type it means a pointer to a variable of that type, so int* is a pointer to an integer and FILE* is a pointer to a FILE. A pointer is a memory cell whose content is the address of another memory cell. The file pointer points to the file information that defines the file's properties, including its name, status and current position, and fopen() returns this pointer.

Closing and Reading/Writing Characters

When a program no longer needs a file it should close it with fclose(), whose prototype is int fclose(FILE* fp); this disassociates the stream from the file and frees the information stored about it, returning 0 on success or EOF on error. Once a file is open, a character can be read with getc(FILE* fp), which returns the next character as an integer or EOF at end of file or on error, and a character can be written with putc(int ch, FILE* fp), which writes the character and returns it, or EOF if an error occurs.

Important Definitions

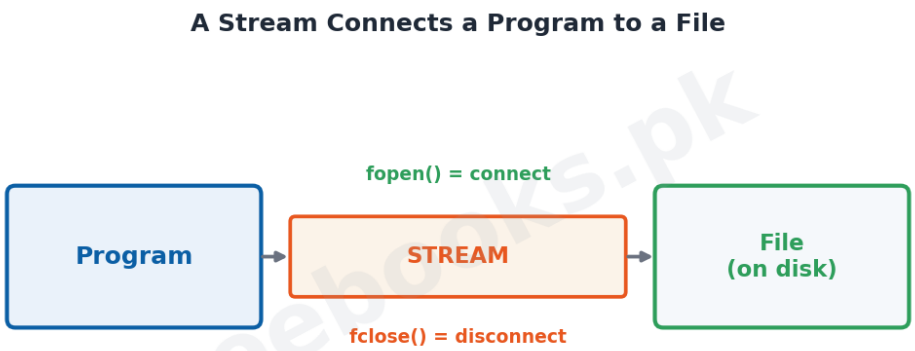
Term	Definition
File	A set of related records saved on secondary storage.
Stream	A logical interface between a program and a file.
Text stream	A sequence of characters, with possible character translations.
Binary stream	A sequence of bytes matching the device one-to-one, with no translation.
EOF	The end-of-file marker placed after the last character of a file.
fopen()	A function that opens a file and returns a file pointer.
File pointer	A variable of type FILE* that points to a file's information.
fclose()	A function that closes a file and disassociates its stream.

Key Facts & Rules

Item	Detail
Include header	#include <stdio.h>
Open a file	FILE* fp = fopen(filename, mode);
Modes	"r" read, "w" write, "a" append, "r+" "w+" "a+"
Check failure	if (fp == NULL) { error }
Close a file	fclose(fp); (returns 0 or EOF)
Read a character	ch = getc(fp);
Write a character	putc(ch, fp);

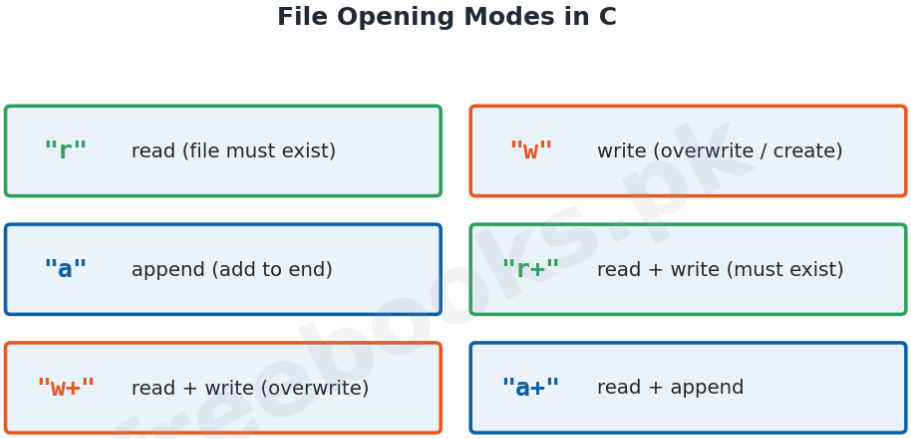
Diagrams & Illustrations

A stream connects a program to a file: how a stream links a program to a file, opened with fopen() and closed with fclose().



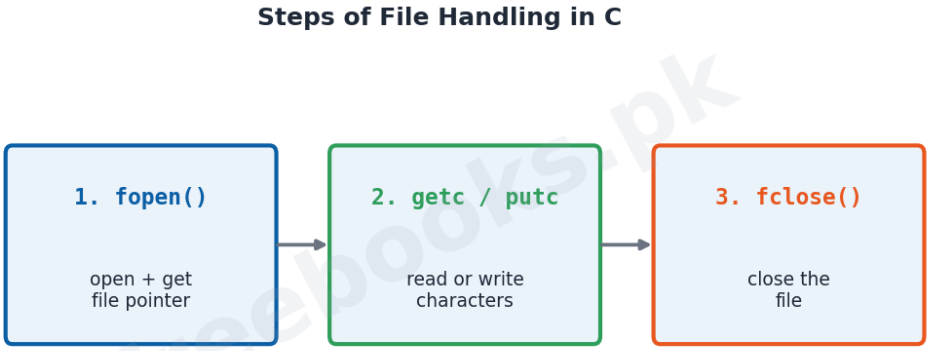
A stream connects a program to a file

File opening modes: the file opening modes in C, r, w, a, r+, w+ and a+, and what each does.



File opening modes

Steps of file handling: the three steps of file handling, open with fopen(), read/write with getc()/putc(), then close with fclose().



Steps of file handling

Solved Examples & Practice

Open for reading

fp = fopen("myfile.txt", "r"); opens an existing text file for reading.

Check the open

if (fp == NULL) printf("Error opening file"); reports failure when fopen returns NULL.

Read a character

ch = getc(fp); reads the next character from the file; it returns EOF at the end.

Write and close

putc('A', fp); writes the character A to the file, then fclose(fp); closes it.

Short Questions & Answers

Q1. Why are files needed?
Ans. To store data and results permanently so they are not lost when the program ends.

Q2. What is a stream?
Ans. A logical interface between a program and a file.

Q3. Name the two types of streams.
Ans. Text stream and binary stream.

Q4. What does EOF mark?
Ans. The end of a file; it is placed after the last character.

Q5. What is the use of the fopen() function?
Ans. It opens a file, associates it with a stream and returns a file pointer.

Q6. What does fopen() return if it fails?
Ans. The NULL pointer.

Long Questions & Answers

Q1: Explain the concept of a stream and describe text and binary streams.

The programs studied in earlier chapters worked only with temporary data, which the user had to enter each time the program ran, because they could not store data permanently; permanent storage of data is achieved by saving it on secondary storage in the form of files, where a file is a set of related records. C itself has no built-in method of performing file input and output, but its standard library, stdio, provides a rich, powerful and flexible set of I/O functions for the purpose, and the key concept behind them is the stream. A stream is a logical interface to a file: it is the channel through which data flows between the program and the file. A stream is associated with a file by an open operation and is disassociated from it by a close operation, and because C works through streams it can treat many different things, such as a disk file, the screen, the keyboard, a port or a file on tape, in the same uniform way. There are two types of streams. A text stream is a sequence of characters in which certain character translations may take place; for example, a newline character may be converted into a carriage-return and line-feed pair, which means there may not be a one-to-one relationship between the characters written by the program and the characters actually stored on the external device. A binary stream, on the other hand, is a sequence of bytes that correspond one-to-one with the bytes on the external device, with no translation at all, so that the number of bytes read or written is exactly the same as the

number of bytes on the device, although an implementation may append a few extra bytes to pad the data to fill a disk sector. Text streams are convenient for human-readable data, while binary streams preserve data exactly as it is.

Q2: Explain how a file is opened and closed in C, including fopen(), the file opening modes and fclose().

Before a program can read from or write to a file, the file must first be opened, and all the standard file handling functions needed for this are declared in the header file stdio.h, which is therefore included in almost every file handling program. A file is opened, and associated with a stream, by the library function fopen(), whose prototype is FILE* fopen(const char* filename, const char* mode). The first argument is the name of the file; if the file is not in the current directory its absolute path must be given, and because a single backslash has a special meaning in C strings each backslash in the path must be written as a double backslash. The second argument is the opening mode, which must be written as a string in double quotes rather than as a single character. The mode determines what may be done with the file: "r" opens an existing text file for reading and fails if the file does not exist; "w" opens a file for writing, creating it if necessary and overwriting its contents if it already exists; and "a" opens a file for appending, so that new data is added to the end of an existing file, again creating the file if it does not exist. In addition, the combined modes "r+", "w+" and "a+" open a file for both reading and writing, differing in whether the file must already exist and whether its current contents are preserved or overwritten. The fopen() function returns a file pointer that is used in all later operations on the file, but if it fails, most commonly because the file to be read does not exist, it returns the NULL pointer, so a program should always test the returned value against NULL before using the file. When the program has no further use for a file it should close it with the fclose() function, whose prototype is int fclose(FILE* fp); this function closes the file associated with the valid file pointer fp, disassociates the stream from the file and frees the information that was stored about the file, returning 0 if it succeeds and EOF if an error occurs.

Q3: Explain the file pointer and describe how characters are read from and written to a file.

To work with a file after opening it, C uses a file pointer, which is a variable of the type FILE that is defined in the header file stdio.h and is declared with a statement such as FILE* fp. In this declaration the symbol star does not mean arithmetic multiplication; when it is used with a data type it indicates a pointer to a variable of that type, so that int star means a pointer to an integer, float star means a pointer to a float, and FILE star means a pointer to a variable of type FILE. Conceptually a pointer is a memory cell whose content is the address of another memory cell, so a pointer does not store a value directly but rather stores the location where a value is kept. The file pointer, in particular, points to the block of file information that describes the properties of the file, including its name, its status and its current position, and this pointer is exactly what the fopen() function returns when a file is opened successfully. Once a file has been opened, single characters can be read from it or written to it, depending on the mode in which it was opened, using two library functions. The getc() function, with the prototype int getc(FILE* fp), reads the next character from the file

and returns it as an integer; it returns the special value EOF when the end of the file is reached or when an error occurs, which is why its return type is int rather than char. The putc() function, with the prototype int putc(int ch, FILE* fp), writes the character stored in ch to the file associated with fp as an unsigned char; although ch is declared as an int a char may be used, and the function returns the character written if it succeeds or EOF if an error occurs. Together the file pointer and these character functions allow a program to move through a file and transfer data one character at a time.

MCQs with Answers

1. Data is stored permanently in the form of:
(a) variables (b) files (c) arrays (d) streams
Correct Answer: (b) files. Files.

2. A set of related records is called a:
(a) stream (b) file (c) pointer (d) buffer
Correct Answer: (b) file. File.

3. A logical interface between a program and a file is a:
(a) file (b) stream (c) pointer (d) record
Correct Answer: (b) stream. Stream.

4. The two types of streams are text and:
(a) numeric (b) binary (c) string (d) logical
Correct Answer: (b) binary. Binary.

5. The end of a file is marked by:
(a) \n (b) EOF (c) NULL (d) \t
Correct Answer: (b) EOF. EOF.

6. The function used to open a file is:
(a) fclose() (b) fopen() (c) getc() (d) putc()
Correct Answer: (b) fopen(). fopen().

7. If fopen() fails it returns:
(a) 0 (b) EOF (c) the NULL pointer (d) 1
Correct Answer: (c) the NULL pointer. The NULL pointer.

8. The mode that appends data to a file is:
(a) "r" (b) "w" (c) "a" (d) "x"
Correct Answer: (c) "a". "a".

9. A variable of type FILE* is a:
(a) record (b) file pointer (c) stream name (d) mode
Correct Answer: (b) file pointer. File pointer.

10. The function used to read a character from a file is:
(a) putc() (b) getc() (c) fclose() (d) fopen()
Correct Answer: (b) getc(). getc().

Quick Revision Summary

- Files store data permanently; a file = a set of related records (stdio.h needed).
- A stream is a logical interface to a file; two types: text and binary.
- Text file: each line ends with \n; the file ends with the EOF marker.
- fopen(filename, mode) opens a file and returns a FILE* (NULL if it fails).
- Modes: "r" read, "w" write (overwrite), "a" append, plus "r+" "w+" "a+".
- getc() reads a character, putc() writes one; fclose() closes the file. Notes by freebooks.pk.

Exam Tips

- Explain why files are needed and define a file.
- Define a stream and differentiate text and binary streams.
- Explain the newline and EOF markers in a text file.
- Give the fopen() prototype and list the file opening modes.
- Explain the file pointer (FILE*) as a pointer to file information.
- Describe fclose(), getc() and putc() with their return values.