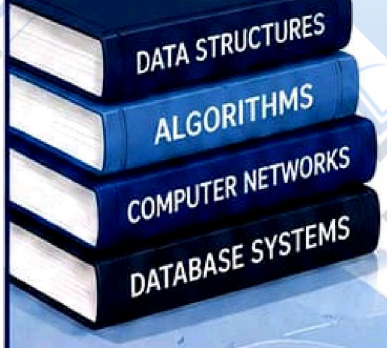
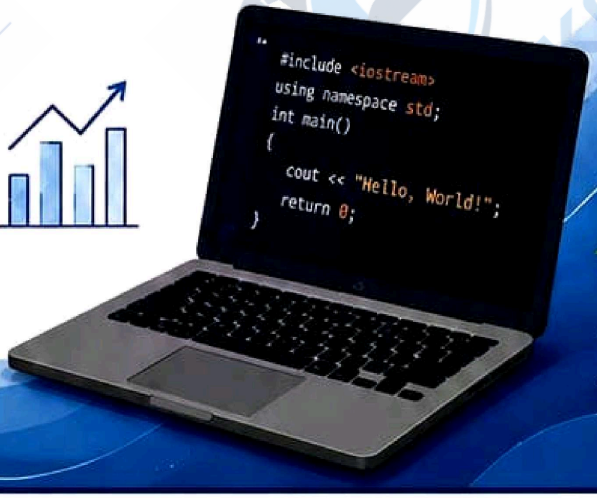
Notes


1010101010
0101010101
1010101010
0101010101
DATA STRUCTURES
ALGORITHMS
COMPUTER NETWORKS
DATABASE SYSTEMS
Good Notes
Better Understanding
Better Results

Unit - III


DATA STRUCTURES
ALGORITHMS
COMPUTER NETWORKS
DATABASE SYSTEMS


#include <iostream>
using namespace std;
int main()
{
cout << "Hello, World!";
return 0;
}

Chapter 3- 8M

One Sentence -

1. What is pointer?
→ A pointer is an variable in C that stores the memory address of another variable.

2. Differentiate between p and *p.
→ p is the pointer variable that stores the address of another variable.

*p is the value at the address stored in pointer ~~variable~~ p.

3. How pointer initialized?
→ A pointer is initialized by assigning it the address of a variable using the address-of operator (&).

4. What is indirection operator and what is its role?
→ The indirection operator in C is the asterisk symbol (*). It is used to access the stored value at a memory address that a pointer is pointing to. This process is called dereferencing.

Role -
It allows the programmer to access or modify the value of a variable using pointer

It plays a key role in dynamic memory pointer arithmetic, and function arguments by references.

5. What is the difference between *p++ and p++.

→ • *p++ means use the value pointed by p, then move p to the next address.
• p++ simply means increment the pointer p to point to the next memory location, without accessing the value.

6. How pointer differ from variable in C?

→ • A variable stores a value directly (eg. i, c, s).
• A pointer stores the address of another variable, not the value itself.

7. Write the names of operator associated with the pointer variable.

→ The operators associated with pointer variable are-

1) Dereference operator (*)-

Accesses the value pointed to by the pointer.

2) Address-of Operator (&)-

Return the memory address of a variable.

8. What is structure in C?

→ A structure in C is a data type that allows grouping variables of different data types under a single name. It is used to represent a record.

9. What is union?

→ A union is a special data type available in C that allows to store different data types in the same memory location.

Long Answer-

1. What is array? Explain declaration and initialization of one-diamentional array with example.

→ Array-

An array in C is a collection of multiple elements of the same data type, stored in ~~con~~ ~~contigous~~ contiguous memory locations. It allows easy access and ~~manipulation~~ handling of a group of values using index numbers.

Declaration of One-dimentional array-

To declare an one-dimentional array in C, you need to specify the type of the elements and the number of elements to be stored in it.

Syntax-

```
datatype arrayName [size];
```

eg-

```
int marks [10];
```

Here,

int is array type, marks is array name and [10]; is size of array.

Initialization of One-dimentional array-

At the time of declaring an array, you can initialize it by providing the set of comma-separated values enclosed within the curly braces {}.

Syntax-

```
data_type array_name[size]={value1,v2,v3...
```

eg-

```
int num[6] = {2,4,6,8,12};
int num[] = {2,4,8};
```

eg-

```
#include<stdio.h>
int main()
{
  int marks[5]={90,85,78,92,88};

  for(int i= 0; i<5; i++)
{
  printf("Mark %d= %d\n", i+1, mark[i]);
                                    }
    return 0;
          }
```

2. Explain how to access elements in a one-dimentional arrays.

→

An array in C is a collection of multiple elements stored in a contiguous memory location. It allows to access or handle values by using index numbers.

Syntax Decleration of One D array

To decleare one d array in C, we need to specify the number of elements and types of the elements.

Syntax -

```
array_type array_name [size];
```

Example -

```
int marks [10];
```

Initialization of one D array.

One d array is initialize at the time of decleration by providing the values seperated by comma in curly braces { }.

Syntax

```
type array_name[size] = {value1 ... valuen}
```

3. Explain declaration and initialization of two dimentional arrays.

→ A two-dimentional array is an array of arrays. It is used to represent tables, matrices, or grid like data, with rows and columns.

Declaration of 2-D array:

Syntax-

datatype arrayname[rowsize][columnsize];

In 2-D array, the elements are organized in form of row and columns, where the first row is represented by a[0][0]. The first number of first index represent the number of rows and the second index number represent the number of column.

Initialization of 2D-array-

We can initialize a 2D array by using a list of values enclosed inside '{ }' and seperated by a comma.

datatype arrayname[row][column] = {v1, v2};

eg- int x[4][3] = {0,1,2,3,4,5,6,7,8,9,10,11};

int x[4][3] = {{0,1,2}, {3,4,5}, {6,7,8}

, {9,10,11}};

5. What is pointer? Explain declaration and initialization of pointer with example.

→ A pointer is an variable in C that stores the memory address of the another variable.

Declaration of Pointer-

Every variable must be declared for its types. Since pointer variables contain addresses that belong to a separate data type, they must be declared as pointers before we use them.

Syntax- data_type *pt_name;

eg- int *p;

pointer can decleare as

```
int* p;
int * p;
int *p;
```

Initialization of Pointer-

The assigning of the adress of a variable to a pointer variable is known as initializati-on. Once a pointer variable has been decleared we can use the assignment operator to variable.

```
eg- int quantity;
    int *p;          // decleration of pointer
    p=&quantity;     // initialization of pointer
```

eg-

```
#include <stdio.h>
int main ()
{
  int x = 25;
 int *ptr = &x;

printf(" Value of x : %d", x);
printf("Address of x:%d" ,&x);

printf(" Pointer ptr holds: %p \n",ptr);

printf(" Value pointed by ptr : %d " , *ptr);

        return 0; }
```

6. What is pointer? Explain pointer operator.

→ A pointer is a variable that stores the memory adderess of other variable.

1. Declaration of a pointer-

Used to declare a pointer that can store the address of a variable of a specific data type.

Syntax- int *ptr;

This means ptr is a pointer to an integer.

2. Initialization-

Assigning the address of a variable to a pointer using the address-of (&) operator.

eg-

```
int a = 10;
int *ptr = &a;
```

Here, ptr stores the address of a.

3. Dereferencing (Indirection)-

Accessing the value at the address stored by the pointer using the dereference (*) operator.

eg- printf("%d", *ptr);

4. Pointer Arithmetic.

You can move the pointer to the next or previous memory location using arithmetic operations.

- ptr ++ : moves to the ~~previous~~ next memory location.
- ptr + n : moves forward by n elements.
- ptr -- : moves to the previous memory location.
- ptr - n : moves back ward by n elements.

eg- ptr++;

5. Null Pointer-

A pointer that does not point to any valid memory location.

eg- int *ptr = NULL;

It is used for safety to check if a pointer is assinged before using it.

7. What is pointer? Explain pointer arithmetic with suitable example?

→ A pointer is an variable in C that stores the memory address of another variable.

Pointer arithmetic.

Pointer arithmetic involves performing operations like addition, substraction, increment or decrement on pointers. Since pointer store addresses these operations move the pointer to different memory locations, depending on the size of data type.

Types of Pointer Arithmetic.

1. Increment (ptr++) -

Moves the pointer to the next memory location of its type.

2. Decrement (ptr--) -

Moves the pointer to the previous memory location.

3. Addition (ptr+n) -

Moves the pointer to n elements forward.

4. Substraction (ptr-n) -

Moves the pointer n elements backward.

eg-

```
#include <stdio.h>
int main ()
{
  int arr[] = {10, 20, 30, 40, 50};

int *ptr = arr;

printf ("First value : %d \n", *ptr);
    ptr++;

printf (" Secod value : %d \n", *ptr);

    ptr = ptr + 2;

printf(" Fourth value : %d \n", *ptr);

    return 0;
        }
```

8. Explain the address (&) and dereference (*) operators in pointer.

→ Pointers are special variables that stores memory addresses of other variables. To use pointers effectively, C provides two key operators.

1. Address - of Operator (&).

The address - of operator (&) is a unary operator used to obtain the memory address of a variable. It tells the compiler to return the address where a ~~variou~~ variable is stored in memory.

Purpose

- It is used to initialize a pointer with the address of a variable.
- Allows indirect access to a variable through pointers.

eg-

```
int a = 5;
int *ptr = &a;
```

- ptr now contains the address of a.
- If a is stored at memory location 1000, then ptr = 1000.

2. Dereference Operator (*).

The dereference operator (*), also called the indirection operator, is used to access or modify the value at the address a pointer is pointing to.

Porpose.

- To read or change the value stored at a specific memory location.
- Enables indirect manipulation of data.

eg-

```
int a=5;
int *ptr = &a;

printf ("%d", *ptr);
    *ptr = 10;
```

- *ptr access the value stored at the address ptr points to.
- *ptr = 10 changes the value at the address which is the value of a.

10. What is structure? Explain the defining structure

→

A structure is a user-defined data type that allows you to group variables of different variables of different data types under a single name. It is used to represent record.

11. Explain initialization of structure in C.

→ Structure initialization means assigning initial values to the members of a structure at the time of declaration or later during runtime.
It helps to define and set default data for a structure in a clear and organized way.

You can initialize a structure when defining the variable using curly braces { } with values in the order of deelaratio

Syntax -

```
struct StructureName
{
    data_type member1;
    data_type member2;
                };
struct StructureName variable = {value1, V2}
```

eg -

```
struct Student
{
    int rollNo;
    float marks;
                };
struct Student s1 = {101, 8.5};
```

12. Explain difference between structure and array in C.

→

- **Structure-**
  A structure is a user-defined data type that allows you to group different types of variables under one name.

- **Array-**
  An array is a collection of similar data types stored in contiguous memory locations under a single name.

Difference

Feature	Structure	Array
Data Type	Can store different data types	Stores same data type only
Defination	Defined using struct keyword	Declared with data type and size.
Memory	Allocated memory for each member	Allocates continuous memory for all elements
Access	Members accessed by dot operator (.)	Elements accessed using index [i]
Usage	Used to model a real-world entity eg. student with name age	Used to store list of values (eg - marks).

13. Explain how do you access of structure members?

→ Once a structure is defined and a variable of that structure type is declared, you can access its members using the dot (.) operator or the arrow (->) operator.

1. Access using dot operator (.)

Used when you are working with a normal structure variable (not a pointer).

Syntax-

```
structure_variable.member_name;
```

eg-

```
#include <stdio.h>
struct Student
{ int roll;
  float marks; };

int main()
{
  struct Student s2={102,90.5};
   struct Student s1;
    s1.roll = 101;
    s1.marks = 85.5;

 printf("Roll : %d", s1.roll);
printf("Mark : %f", s1.marks);

      return 0;
          }
```

Access Using Arrow Operator (->)

Used when you have a pointer to a structure.

Syntax-

```
pointer_to_structure -> member_name;
```

eg-

```
#include <stdio.h>
struct Student
{
  int roll;
float marks; };

int main()
{
  struct Student s2 = { 100, 90.5};

  struct student *ptr = &s2;

printf("Roll : %d, ptr -> roll);
printf("Mark : %d, ptr -> marks);

      return 0;
      }
```

Q4. What is union? Explain its syntax with example.

→ A union is a user defined data type in C that allows storing different types of data in the same memory location. But only one member can hold a value at a time because all members share the same memory.

1. Declaration-

To declare a union, use the union keyword followed by the union name and its member inside curly braces {}.

Syntax -

```
union UnionName
{
    data_type member1;
    data_type member2;
    ...
};
```

eg-

```
#include<stdio.h>
union Data
{   int i;
    float f;
    char ch; };

int main()
{   union Data d;
    d.i = 100;
    printf("Integer - %d", d.i);
    d.f = 3.14;
    printf("float - %f", d.f);
    d.ch = 'A';
    printf("Char - %c", d.ch);
    return 0;
}
```

15. Explain the defference between structure and union with example.

→ Both structure and union are user-defined data types that group variables of different data types under one name.

Feature	Structure	Union
Keyword	struct	union
Memory Allocation	Each member gets separate memory	All members share the same memory.
Size	Total size = sum of sizes of members	Size = size of the largest member
All Memory Active	Can store values in all members simultaneously	Can store value in only one member at a time.
Usage	When all data members are needed	When only one data member is used at a time

Example of Structure-

```
#include<stdio.h>
struct Student
{
    int roll;
float marks;
char grade; };

int main()
{
  struct Student s={1, 90.5, 'A'};

  printf("Roll %d, Marks %f, Grade %c, s.roll.
          s.marks, s.grade);
        return 0; }
```

Example of Union.

```
#include <stdio.h>
union Student
{  int roll;
 float mark;
char grade; };

 int main()
{ union Student s;
    s.roll = 1;
    s.mark = 88.5;
    s.grad = 'A';
printf("Roll %d, s.roll.);
printf("Mark %f, s.mark);
printf("Grade %c, s.grade);
          return 0; }
```

17. What are advantages and disadvantages of using union in C?

→

Advantages-

1. Memory Efficiency -
   All members share the same memory location. Saves memory when only one member is used at a time.

2. Useful in Embedded Systems.
   Ideal for memory-constrained environment like microcontrollers.

3. Efficient Type Reuse.
   Allow storing multiple data types in the same location, one at a time.

4. Helps in Handling Raw Data
   Can be used to access the same memory areas as different data types. Eg convert int to byte.

Disadvantages-

1. Only One Member Active-

At any point, only the value of the most recently assigned member is valid.

2. Risk of Data Loss-

Assigning a value to one member overwrites the previous members data.

3. No Type Safety-

Difficult to track which member is currently valid, leading to errors.

4. Limited Uses Cases-

Not suitable when multiple member need to hold values simultaneously. (use struct insted).

18. Explain how to initialize a union in C?

→ A union in C can be initialized at the time of declaration. Since all members of union share the same memory, only one member can be initialized at a time.

Syntax-

```
union UnionName
{
    data_type member1;
    data_type member2;
    ...... };

union UnionName variable_name
= {value_for_first_member};
```

Only the first member in the union can be directly initialized using this method.

→ Assinging other members. (After declaration)

Other members must be assingned separately after declaration.

```
d.f = 3.14;
```