UNIT: 6
Structure, Union and Files

❖ Structure

❖ Introduction to Structures and Declaration, Initialization and accessing of Structures:

- Structure is a collection of logically related data items with different data type.
- Structure must be defined first, that may be used later to declare structure.

Syntax:

```
struct structure_name
{
datatype member1;
datatype member2;
---
datatype memberN;
} ;
```

- struct is a keyword, it declares structure to hold data.
- Structure_Name is the name of structure (structure tag).
- member1, member2,… are members of structure.

Example:

```
struct student
{
char name[20];
int roll_no;
float per;
}s1;
```

Here **s1** is a structure variable.

Structure Initialization:

Example:

```
struct student
{
char name[20];
int roll_no;
float per;
}s1={"Arav",20,85.5};
```

Example:

```
#include<stdio.h>
 struct student
```

```
{
   char name[20];
   int roll_no;
   float per;
}s1;
void  main()
{
   struct student s1={"Arav",20,85.5};

   printf("\nName       : %s",s1.name);
   printf("\nRollNo      : %d",s1.roll_no);
   printf("\nPercent    : %f",s1.per);
   }
```

❖ Introduction to union and Declaration, Initialization and accessing of union:

- Union is a group of memory that is used to store variables of different datatypes.
- Union are same as structure, but main difference between structure and union is in **terms of storage.**

Syntax :

```
union union-name
        {
datatype member1;
datatype member2;
---
datatype memberN;
        };
```

Example:

```
union  student
{
char name[20];
int roll_no;
float per;
};
```

Example:

```
union  student
{
char name[20];
int roll_no;
float per;
}u1;
```

```
void main()
{
   u1.roll_no=65;
   printf("\nRollNo       : %d",u11.roll_no);
}
```

Difference between structure and union

	Structure	union
1	In structure, each member has separate memory allocated.	In union all member shares common memory.
2.	Any member can be accessed at any time.	Only one member can be accessed at a time.

3.	struct data
	{
	int a;
	float b;
	char c;
	};
		union data
		{
		int a;
		float b;
		char c;
		};

4.	Size of structure is total memory allocated to each member. Size of student= total sizeof(a,b,c);	Size of union is maximum memory allocated to any member. Size of student= maximum memory from (a,b,c);
5	Structure is defined using **struct** keyword.	Union is defined using **union** keyword.

Difference between Array and structure.

	Array	Structure
1	An array is a collection of variables of same data type known by same name.	A structure is a collection of variables of different data type known by same name.
2.	An array is a derived data type.	A structure is a programmer defined.
3.	For array we have to declare an array variable and use it.	For structure we have to design and declare a data structure before the variables of that type are declared and used.

4.	Example:
	int a[5];
		Example:
		struct data
		{
		int a;
		float b;
		char c;
		};

❖ User-defined Data types: enum, typedef

Enumerated Data Type:

- Enumerated data type is user defined data type.
- Using enumerated data type we can create more than one symbolic constant at a time.

Syntax: **enum identifier {value1, value2,value3};**

Here, enum is a keyword to declare enumerated data type.
Identifier is a user defined enumerated data type.

For Example:

```
#include<stdio.h>
enum week {Mon=1, Tue, Wed, Thur, Fri, Sat, Sun};
void main()
{
 enum week day;
 day = Wed;
 printf("%d",day);
getch();
}
```

Output:2

typedef :

The typedef is a keyword that is used in C programming to provide existing data types with a new name.
typedef keyword is used to redefine the name already the existing name.

When names of datatypes become difficult to use in programs, typedef is used with user-defined datatypes.

Syntax:
typedef <existing_name> <alias_name>

Example:
typedef long mylong

❖ Files

Introduction to text Files

File:

A file is a group of related data which is stored in a disk.

File operations:

(1) Naming a file
(2) Opening a file
(3) Reading data from a file
(4) Writing data to a file
(5) Closing a file

Basic file operations:

fopen() - create a new file or open a existing file
fclose() - close a file
getc() - reads a character from a file
putc() - writes a character to a file
fscanf() - reads a set of data from a file
fprintf() - writes a set of data to a file
getw() - reads a integer from a file
putw() - writes a integer to a file
fseek() - set the position to desire point
ftell() - gives current position in the file
rewind() - set the position to the beginning point

fopen() and fclose():

fopen(): Create a new file or open an existing file.

Syntax:

```
FILE *fp;
fp=fopen("filename", "mode");
```

Here,fp is a file pointer to the datatype FILE.

Second statement open a file named filename and mode specify the opening of this file.

Example:

```
FILE *fp;
fp=fopen("E:\\abc.txt", "r");
```

fclose(): Close the file

A file must be closed as soon as all operation on it have been completed.

Syntax:

```
fclose(file pointer);
```

Example:

```
FILE *fp;
```

```
fp=fopen("E:\\abc.txt", "r");
----------
fclose();
```

fgetc() and fputc():

fputc(): Write character to the file

Syntax: fputc(c,fp);

Here c is a character type variable and fp is a file pointer.

This statement write the character stored in variable c to the file.

Example:

```
void main()
{
char c='A';
FILE *fp;
Fp= fopen(" E:\\ abc.txt", "w");
 fputc(c,fp);
fclose(fp);
getch();
}
```

fgetc(): Read character from file

Syntax: c=fgetc(fp);

Here c is a character type variable and fp is a file pointer.

Example:

```
void main()
{
char c;
FILE *fp;
Fp= fopen(" E:\\ abc.txt", "r");
 c = fgetc(fp);
printf("%c", c);
fclose(fp);
getch();
}
```

fscanf() and fprintf():

These functions are used for reading and writing the data to the file.

fprintf() : Write all data written in list to the file.

Syntax: fprintf(fp , "Control String ", List);

Where fp is a file pointer.

List may include variable, constants and string.

Example:

```
void main()
{
```

```
char name[10]=”abc”;
int  id=1;
FILE *fp;
Fp= fopen(“ E:\\ abc.txt”, “w”);
      fprintf(fp,”%s %d”, name,id);
fclose(fp);
getch();
}
```

fscanf() : Read all data written in file and store in variables.

Syntax: fscanf(fp , “Control String “, List);

Where fp is a file pointer.

List may include variable, constants and string.

Example:

```
void main()
{
char name[10];
int  id;
FILE *fp;
Fp= fopen(“ E:\\ abc.txt”, “r”);
      fscanf(fp,”%s %d”, name, &id);
      printf(“Name : %s and id : %d”, name, id);
fclose(fp);
getch();
}
```

Example of reading data from a file and writing to it(in text mode):

```
#include<stdio.h>
#include<conio.h>
void main()
{
FILE *fp;
char ch;
fp = fopen("hello.txt", "w");
printf("Enter data");
while( (ch = getchar()) != EOF) {
  putc(ch,fp);
}
fclose(fp);
fp = fopen("hello.txt", "r");

while( (ch = getc(fp)! = EOF)
  printf("%c",ch);
```

```
fclose(fp);
}
```