Pointer

Call by value

- int p;
- Formal parameter p is a local variable
- It cannot change the actual parameter

Actual parameter may be a constant, a variable, or an expression

Call by reference

- int &p
- Formal parameter p is a local reference
- It can change the actual parameter
- Actual parameter must be a variable

Memory Allocation

Static Memory Allocation

- Allocation take place at **Compile Time**
- Memory wastage

Dynamic Memory Allocation

- Allocation take place at **RunTime**
- No memory wastage.

Structure

1. It is used to create new user define data type
2. Structure representing a collection of variables of same or different datatype together under a single name.
3. The variables inside the structure are called as data members

Difference between Structure and Array

1. Array demands a homogeneous(same) data type
2. Elements in an array are referred by their positions
3. Structure is a heterogeneous(different) data type
4. Where as members are referred by their unique names.

Union

1. All the members of a union use the same memory space, unlike structure will allocate independent memory location for each data members

2. Less memory space is required since all members are stored in the same memory locations

Disadvantage:

- This will not suite for all applications

File Handling

File is a place on a disk where group of related data stored as permanently

Eg:

Hard Disk,

Floppy Disk,

CD-ROM,

DVD

Type of Files

1. **ASCII File Format**
2. **Character Oriented**

BINARY File Format

Byte Oriented(Bits)

Basic File Operations

1. Open a file
2. Naming a file
3. Read from a file
4. Write to a file
5. Close a file

Opening a file

First Create a File_pointer Variable

Syntax:

FILE *file_prt_var;

Eg:

FILE *fp;

To Open a file

Syntax:

File_ptr_Var = fopen("Filename",mode of file");

Eg:

fp=fopen("file1.dat","w");

Naming a File

- Name file as two parts
- Primary part
- Minimum one char. To Maximum 8 chars
- Extension Part (optional)
- Minimum no char. Maximum 3 Chars

Eg:

Student.dat

Student

Mode of File

r – read from a file

w – write to a file

a – append to a file

r+ - read and write to file

w+ - write and read from file

a+ - append and read from file

Write a Character to a file

Syntax:

fputc(char_var, file_ptr_var);

Eg:

fputc(ch,fp);

Read a character from a file

Syntax:

char_var=fgetc(file_ptr_var);

Eg:

ch=fgetc(fp);

Writing Different data type

Syntax:

fprintf(file_ptr,"format",var1,var2…");

Eg:

fprintf(fp,"%d\t%s\t%d",s.rno,s.name,s.m1..);

Reading different data type

Syntax:

fscanf(file_ptr,"format",&var1,&var2…");

Eg:

fscanf(fp,"%d%s%d", &s.rno,&s.name,
&s.m1..);

Binary File Format

To Write

Syntax:

fwrite(address of structure,size of structure, No. of structure,
file_ptr_var);

Example:

fwrite(&s,sizeof(s),1,fp);

Binary File Format

To Read

Syntax:

fread(address of structure,size of structure, No. of structure,
file_ptr_var);

Example:

fread(&s,sizeof(s),1,fp);

Preprocessor

1. Is a collection of special statements called directives that executed at the beginning of the compilation process.
2. Which permits the insertion of file in to a C program.
3. Macro definition
4. File inclusion
5. Conditional compilation

Points to be noted

- All preprocessor directives should begin with an # sign
- Should not end with semi colon.