**UNIT 6 : Dynamic memory Allocation**

* The process of allocating memory at the time of execution or at the runtime, is called dynamic memory location.
* Two types of problem may occur in static memory allocation.
* If number of values to be stored is less than the size of memory, there would be wastage of memory.
* If we would want to store more values by increase in size during the execution on assigned size then it fails.
* Allocation and release of memory space can be done with the help of some library function called dynamic memory allocation function.
* These library function are called as dynamic memory allocation function. These library function prototype are found in the header file, “alloc.h” where it has defined.
* Function take memory from memory area is called heap and release when not required.
* Pointer has important role in the dynamic memory allocation to allocate memory.

**malloc():**

This function use to allocate memory during run time, its declaration is

void\*malloc(size);

**calloc()**

Similar to malloc only difference is that calloc function use to allocate multiple block of memory .

**Two arguments are there**

* 1st argument specify number of blocks
* 2nd argument specify size of each block.

**realloc()**

The function realloc use to change the size of the memory block and it alter the size of the memory block without loosing the old data, it is called reallocation of memory.

**It takes two argument such as;**

int \*ptr=(int \*)malloc(size);

int\*p=(int \*)realloc(ptr, new size);

**free():-**

Function free() is used to release space allocated dynamically, the memory released by free() is made available to heap again. It can be used for further purpose.

**Syntax for free declaration .**

void(\*ptr)

Or

free(p)

When program is terminated, memory released automatically by the operating system. Even we don’t free the memory, it doesn’t give error, thus lead to memory leak.

We can’t free the memory, those didn’t allocated.

**File handling**

**File:** the file is a permanent storage medium in which we can store the data permanently.

Types of file can be handled ,we can handle three type of file as

(1) sequential file

(2) random access file

(3) binary file

**File Operation**

**opening a file:**

Before performing any type of operation, a file must be opened and for this **fopen()** function is used.

syntax:

file-pointer=fopen(“FILE NAME ”,”Mode of open”);

example:

FILE \*fp=fopen(“ar.c”,”r”);

If fopen() unable to open a file than it will return NULL to the file pointer.

**File-pointer:** The file pointer is a pointer variable which can be store the address of a special file that means it is based upon the file pointer a file gets opened.

**Declaration of a file pointer:-**

FILE\* var;

**Modes of open**

**The file can be open in three different ways as**

1. **Read mode** ’ r’/rt
2. **Write mode** ’w’/wt
3. **Appened Mode** ’a’/at

**Reading a character from a file**

**getc() i**s used to read a character into a file

Syntax:

character\_variable=getc(file\_ptr);

**Writing acharacter into a file**

**putc()** is used to write a character into a file

puts(character-var,file-ptr);

**ClOSING A FILE**

**fclose()** function close a file.

fclose(file-ptr);

fcloseall () is used to close all the opened file at a time

**File Operation**

The following file operation carried out the file

(1)creation of a new file

(3)writing a file

(4)closing a file

Reading and writing a characters from/to a file **fgetc()** is used for reading a character from the file

Syntax:

character variable= fgetc(file pointer);

**fputc()** is used to writing a character to a file

Syntax:

fputc(character,file\_pointer);

Reading and writing a string from/to a file

**getw()** is used for reading a string from the file

Syntax:

gets(file pointer);

**putw()** is used to writing a character to a fileReading and writing a string from/to a file

**fgets()** is used for reading a string from the file

Syntax:

fgets(string, length, file pointer);

**fputs()** is used to writing a character to a file

Syntax:

fputs(string,file\_pointer);

***REFERENCE BOOKS:***

*1 E.Balagurusamy “Programming in C”. Tata McGraw Hill*

*2 Y. Kanetkar “Let Us C”. BPB publication*

*3 Ashok N. Kamthane “Programming with ANSI and TURBO C”. Pearson Education.*