Khulna University Of Engineering & Technology

KUET

SESSIONAL REPORT

Course No. CSE 1102

Department Of Computer Science and Engineering

Experiment No. _____

Name of the Experiment Introduction to dynamic memory allocation, enumeration command line parameters and makefile

Remarks

Name _____

Roll _____

Date of Performance ..... Group No _____

Date of Submission..... ..... Year first (2022-23)

Semester First

Title: Dynamic memory allocation, enumeration, command line parameters and makefile in c programming

Objectives:

(I) To understand memory management using dynamic memory allocation

(II) To learn ~~enumer~~ enumeration to increase code readability.

(III) To learn command line parameters to make program flexible.

(IV) To understand make files for project management.

Introduction:

Dynamic memory allocation, enumeration, command line parameters and make files in c programming are very important to make the code more efficient.

Dynamic memory allocation:

The process of allocating memory at run time is known as dynamic memory allocation. In other words, procedure of changing the size of a data structure. There are four library routines known as "memory management functions" that can be allocating and freeing memory during program execution. They are:-

- malloc
- calloc
- realloc
- free

Malloc:

The 'malloc" or "memory allocation" method in C is used to dynamically allocate a single large block of memory with the specified size. It returns a pointer of type void which can be cast into a pointer of any form. It doesn't initialize memory at execution time so that it has initialized each block with the default garbage value initially.

Syntax:

```
ptr = (cast_type*) malloc(byte_size)
```

Example:

```
#include<stdio.h>
void main()

int num;
int *ptr;
printf("How many numbers:");
scanf("%d",&num);
ptr=(int*) malloc(num* sizeof(int));
for (int i= 0; i<num; i++){
        *ptr=i;
         ptr++;
}
ptr--;
```

```
for (int i=num; i>0; i--){
    printf("\n%d",*ptr);
        ptr--;
    }
    free (ptr);
    getch();
}
```

Calloc:

Calloc or contiguous allocation method in C is used to dynamically allocate the specified number of blocks of memory of the specified type. It's similar to malloc() but has two different points and these are:-

- It initializes each block with a default value '0'.
- It has two parameters or arguments as compare to malloc().

Syntax:

```
ptr=(cast_type*) calloc(n, element_size);
```

(P.T.O)

Example:

```
#include <stdio.h>

struct student
{
char name[10];
int age;
};
int main(){
typedef struct student record;
record *st_ptr, *end_ptr;
int class_size;
printf("take size\n");
scanf("%d",&class_size);
st_ptr=(record*) calloc(class_size; size of(record));
  end_ptr=st_ptr;
for(int i=0; i<class_size; i++)
{
scanf("%s %d", st_ptr->name, &st_ptr->age);
  st_ptr++;
}
printf("The output is :\n");
for (int i=0; i<class_size; i++)
{
printf("%s %d\n", end_ptr->name, end_ptr->age);
end_ptr++;
}
return 0;
}
```

Realloc:

"realloc" or "re-allocation" method in C is used to dynamically change the memory allocation of a previously allocated memory. In other words, if the memory previously allocated with the help of malloc or calloc is insufficient, realloc can be used to dynamically re-allocate memory. reallocation of memory maintains the already present value and new blocks will be initialized with the default garbage value.

Syntax:

```
ptr = realloc(ptr,newsize);
```

Example:

```
#include<stdio.h>
#include<string.h>
int main(){
  char *buffer;
  if(buffer=(char*)malloc(10)==NULL)
  {
   printf("malloc failed:\n");
   exit(1);
  }
 strcpy(buffer,"hydrabad");
printf("buffer contains : %s\n",buffer);
if((buffer=(char*)realloc(buffer,15))==NULL)
{
  printf("realloc failed.\n");
} exit(1);
```

(P.T.O)

```
printf("\n buffer size modified \n");
printf("buffer still contains:%s\n", buffer);
strcpy(buffer,"sundenabad");
printf("\nbuffer now contains: %s\n", buffer);
free(buffer);
return 0;
}
```

free():

"free" method in C is used to dynamically de-allocate the memory. The memory allocated using functions malloc() and calloc() is not de-allocated on their own. Hence the free() method is used. whenever the dynamic memory allocation takes place. It helps to reduce wastage of memory by freeing it.

Syntax:

```
free(ptr);
```

Enumeration:

Enumeration (or enum) is a user defined data-type in C. It's mainly used to assign names to integral constants, the names make a program easy to read and maintain

Syntax:

```
enum state{working = 1, failed = 0};
```

Example:

```
#include <stdio.h>
enum colors{black, blue, cyan, green};

int main()
{
enum colors foreground, background;
 foreground = black;
 background = green;
printf("%d", foreground);
return 0;
}
```

Command line parameters:

The most important function of C is the main() function. It's mostly defined with a return type of int and without parameters as shown:

```
int main{
    ....
}
```

User can give command-line arguments in c. Command line arguments are the values given after the name of the program in the command-line shell of Operating Systems. The first of these, argc, must be an integer ~~value~~ variable. The second, argv, is an array of pointers to characters i.e. an array of string.

Syntax:

```
void main(int argc, char *argv[])
{
....
}
```

Makefile: general:

MAKE is driven by a make file, which contains a list of target files, dependent files and commands. A target file requires its dependent-files to produce it.

Syntax:

```
target_file1 : dependent_file list
'command_sequence
:
:
target :fileN : dependent_file list
    command_sequence.
```

Discussion:

For dynamic memory allocation, it should be checked that the allocation is successful or not. And the allocated memory should be made free to prevent memory leaks. Enumeration can be used for named constants, improving readability. Arbitrary values shouldn't be assigned to enumeration constants without a clear reason. Without proper checks availability of command line parameters shouldn't be assumed. For make file, dependencies and build rules should be defined for efficient compilation.
These should be maintained while using these things.

Conclusion:

Dynamic memory allocation facilitates efficient memory usage, enabling flexible data structures and preventing fixed-size limitations. Enumeration enhances code readability by assigning meaningful names to integral values, reducing errors and aiding maintainability. Command line parameters enables runtime customization, making programs versatile and configurable without modifying the source code. Make file automates the build process, ensuring efficient compilation, managing dependencies and simplifying project organization and maintenance.

Reference:

(I) C lass Slides.

(II) 'Programming in ANSI C' by Balaguruswamy

(III) https://www.geeksforgeeks.org