Indian Institute of Information Technology Kottayam

Dynamic Memory Allocation

BY

Arun Cyril Jose

Dynamic Memory Allocation

- There are four library functions under <stdlib.h> for dynamic memory allocation.

- malloc()
- calloc()
- free()
- realloc()

malloc()

- Reserves a block of memory of specified size and return a pointer of type ‘void’ (i.e. *void**) which can be type casted into a pointer of any form.

- ptr = (cast_type*) malloc (byte_size);
- Here, ptr is a pointer of cast_type

- Returns a pointer to an area of memory with size of *byte_size* specified in the argument.

malloc()

- Example:

```
int *ptr;
ptr = (int*) malloc (10 * sizeof(int));
```

- Returns a (void*) pointer [which is type casted into (int*) pointer] pointing to the address of the first byte of memory that is allocated.
- If the allocation fails, it returns a NULL pointer.

calloc()

- Dynamic memory allocation.
- ptr = (cast_type*) calloc (n, element_size);
- Here, ptr is a pointer of cast_type

- Allocates multiple (here it is ‘n’) blocks of memory each of same size specified by element_size.
- Sets all byes to zero.

calloc()

- Example:

```
float *ptr;
ptr = (float*) calloc (5, sizeof(float));
```

- Allocates 5 blocks of memory each of size 4 bytes.
  - If float takes 4 bytes to be stored in the memory.
- Sets all byes to zero.

free()

- Dynamically allocated memory created using malloc() or calloc() does not get freed on its own.
- We have to explicitly use free() to release the allocated space.
- Example:

```
int *ptr;
ptr = (int*) malloc (10 * sizeof(int));
//—Rest of your program—
//When the pointer ptr is no longer needed free it.
free(ptr);
```

Question 1

- Sum of *n* elements using Dynamic memory allocation.

Dynamic Memory Allocation

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int num, i, *ptr, sum = 0;
    printf("Enter number of Elements: ");
    scanf("%d", &num);
    ptr = (int*) malloc(num * sizeof(int));
    if (ptr == NULL)
    {
        printf("Memory not allocated");
        exit(0);
    }
    printf("Enter the Array Elements: \n");
    for(i = 0; i < num; i++)
    {
        scanf("%d", (ptr + i));
        sum += *(ptr + i);
    }
    printf("\n Sum = %d: ", sum);
    free(ptr);
    return 0;
} //main() ends
```

realloc()

- If you find that previously allocated memory is insufficient or more than required, you can change the previously allocated memory size using realloc().
- Example:

```
int *ptr;
ptr = (int*) malloc (10 * sizeof(int));
//Assigns 10 * 2 = 20 bytes to integer pointer ptr
ptr = (int*) realloc (ptr, (20 * sizeof(int)));
//Increases the size of integer pointer ptr to 20 * 2 = 40 bytes
```

Question 1

- Sum of *n* elements using Dynamic memory allocation.

Dynamic Memory Allocation

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int num, i = 0, *ptr, sum = 0;
    ptr = (int*) malloc(sizeof(int));
    if (ptr == NULL)
    {
        printf("Memory not allocated");
        exit(0);
    }
    printf("Enter the Array Elements, -999 to stop: \n");
    scanf("%d", ptr);
    while (*(ptr+i) != -999)  {
        sum += *(ptr + i);
        i++;
        ptr = (int*) realloc(ptr, ((i+1) * sizeof(int)));
        if (ptr == NULL)  {
            printf("Memory not allocated");
            exit(0);
        } //if block ends
        scanf("%d", (ptr + i));
    } //while loop ends
    printf("\n Sum = %d \n", sum);
    free(ptr);
    return 0;
} //main() ends
```

Question 2

- Read a String from the user, make it completely Dynamic.

Dynamic Memory Allocation

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
      char *str, c;
      int i = 0;
      str = (char*) malloc(sizeof(char));
      if (str == NULL)
      {
           printf("Memory not allocated");
           exit(0);
      }
      printf("Enter your String: ");
      c = getc(stdin);
      while (c != '\n')
       {
           *(str + i) = c;
            i++;
            str = (char*) realloc(str, ((i+1) * sizeof(char)));
            if (str == NULL)
            {
                 printf("Memory not allocated");
                 exit(0);
            }
            c = getc(stdin);
       }
       *(str + i) = '\0';
       printf("\nEntered String is: %s ", str);
       free(str); return 0;
} //main() ends
```

Structure Pointer

- **Example:**

```
struct student
{
    int rollno;
    char name[30];
    int age;
 } student1;
struct student *p;
p = &student1;
```

Structure Pointer

- Allocate memory to store 10 elements of the structure ‘student’

```
struct student *p;
p = (struct student*) malloc(10 * sizeof(struct student));
```

- Access each element of the structure like this:

(*p).rollno;	p->rollno;
(*p).name[30];	p->name[30];
(*p).age;	p->age;

Question 3

- Read details of ‘n’ students using dynamic memory allocation. Each record consists of name, rollno, age and marks.

Dynamic Memory Allocation

```
#include <stdio.h>
#include <stdlib.h>
 struct student {
        char name[30];
        int rollno;
        int age;
        float marks;
 } *p;
int main() {
       int i, num;
       printf("Enter the number of Students: ");
       scanf("%d", &num);
       p = (struct student*) malloc(num * sizeof(struct student));
       if (p == NULL) {
            printf("Memory not allocated");
            exit(0);
       } //end of if block
       printf("Enter student details: \n");
       for (i = 0; i < num; i++) {
              fgets((p+i)->name, 30, stdin);
              scanf("%d", &((p+i)->rollno));
              scanf("%d", &((p+i)->age));
              scanf("%f", &((p+i)->marks));
        }
        for (i = 0; i < num; i++) {
          puts((p+i)->name);
          printf("Roll: %d\n", (p+i)->rollno);
          printf("Age: %d\n", (p+i)->age);
          printf("Marks: %f\n", (p+i)->marks);
        }
        free(p); return 0;
} //main() ends
```