Dynamic Memory Allocation

- malloc

malloc()

- ***Dynamic memory allocation*** allows your program to obtain **more memory space** while running, or to release it if it's not required.
- You can **manually handle memory space** for a program.
- **malloc()** **allocates** requested size of bytes and returns a pointer first byte of allocated space.
- **free()** **deallocates** the previously allocated space.

Dynamic Memory Allocation in C
(as well as in C++)

- **Dynamic** memory **allocation** with the ***malloc()***, and **deallocation** with ***free***. For example,

```
#include <stdlib.h>
    .......
    scanf_s("%d", &i);
    char *buffer = (char*)malloc(i + 1);
     .......
    free(buffer);
```

malloc also allocates memory on the *heap*.

✓ scanf_c() can be used instead scanf().

Example

```
#include "stdafx.h"
#include <iostream>  // <stdio.h> in C
#include "conio.h"
using namespace std;
#include <stdlib.h>
                // C/C++ malloc, free, rand

void main()
{
  int i, n;
  char * buffer;

printf("How long do you want the string? ");
scanf_s("%d", &i);
```

```
buffer = (char*)malloc(i + 1);
if (buffer == NULL) exit(1);

for (n = 0; n<i; n++)
   buffer[n] = rand() % 26 + 'a';
buffer[i] = '\0';
  // '\0' the null character. The array is
  // terminated with a '\0' to mark the end.
printf("Random string: %s\n", buffer);
free(buffer);

while (!_kbhit());
}
```

0	1	2	3	4	5	6			63
H	E	L	L	O	\n	\0	.....		

Example

```
#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));
       //memory allocated using malloc
   if(ptr == NULL)
    {
     printf("Error! memory not allocated.");
     exit(0);
    }
```

```
printf("Enter elements of array: ");
    for(i = 0; i < num; ++i)
     {
       scanf("%d", ptr + i);
       sum += *(ptr + i);
     }

    printf("Sum = %d", sum);
    free(ptr);
    return 0;
}
```

Discussions

- *malloc* allocates memory on the **heap**. It is the old **C-style way of dynamically allocating memory**. In C++, it is essentially deprecated in favour of the *new* operator.

- *alloca* dynamically creates memory on the **stack**. It's not part of standard C either, so I would avoid using it.

- *main()* local variables are on the **stack.**