CS F111: Computer Programming

(Second Semester 2021-22)

Lect 27: DMA: Malloc(),Calloc(),Realloc().

Nikumani Choudhury

Asst. Professor, Dept. of Computer Sc. & Information System

BITS Pilani

Hyderabad Campus

malloc and calloc

calloc() initializes the allocated memory with 0 value.	malloc() initializes the allocated memory with garbage values.

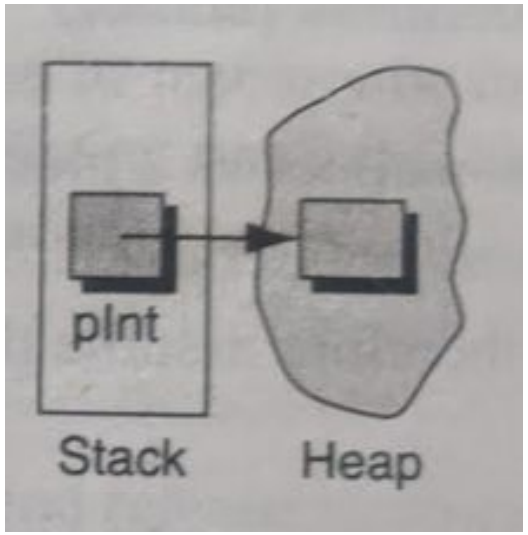
void* malloc(byte-size)

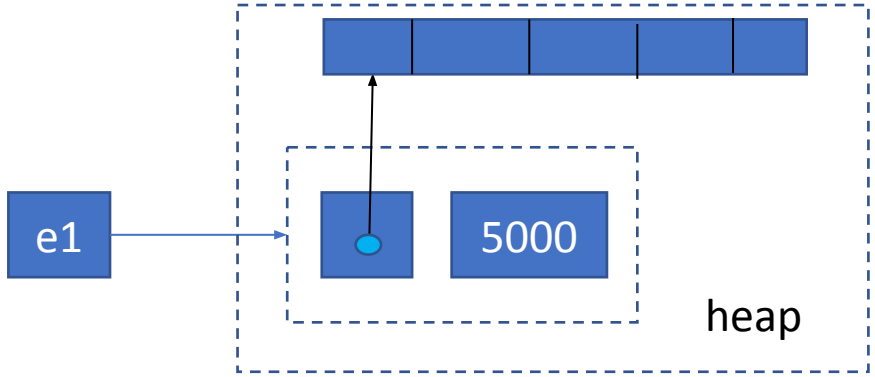
is used for allocating block of memory at runtime

```
int *pInt;
pInt = (int*) malloc (100 * sizeof(int));
```

void *calloc(number of items, element-size)

```
struct employee { char *name; int salary; };
typedef struct employee emp;
emp *e1;
e1 = (emp*)calloc(30,sizeof(emp));
```





```c
#include <stdio.h>
#include<stdlib.h>

int main()
{
    int n, i, *ptr;
    printf("Enter total number of values: ");
    scanf("%d",&n);
    ptr=(int*)malloc(n*sizeof(int));
    printf("Enter the values: ");
    for(i=0;i<n;i++)
    {
        scanf("%d", (ptr+i));
    }
    printf("Entered values are: ");
    for(i=0;i<n;i++)
    {
        printf("%d ",*(ptr+i));
    }
    free(ptr);
    return 0;
}
```

```c
#include <stdio.h>
#include<stdlib.h>

int main()
{
    int n, i, *ptr;
    printf("Enter total number of values: ");
    scanf("%d",&n);
    ptr=(int*)calloc(n, sizeof(int));
     for(i=0;i<n;i++)
    {
        printf("%d ",*(ptr+i));
    }
    printf("Enter the values: ");
    for(i=0;i<n;i++)
    {
        scanf("%d", (ptr+i));
    }
    printf("Entered values are: ");
    for(i=0;i<n;i++)
    {
        printf("%d ",*(ptr+i));
    }
    free(ptr);
    return 0;
}

```

inpu

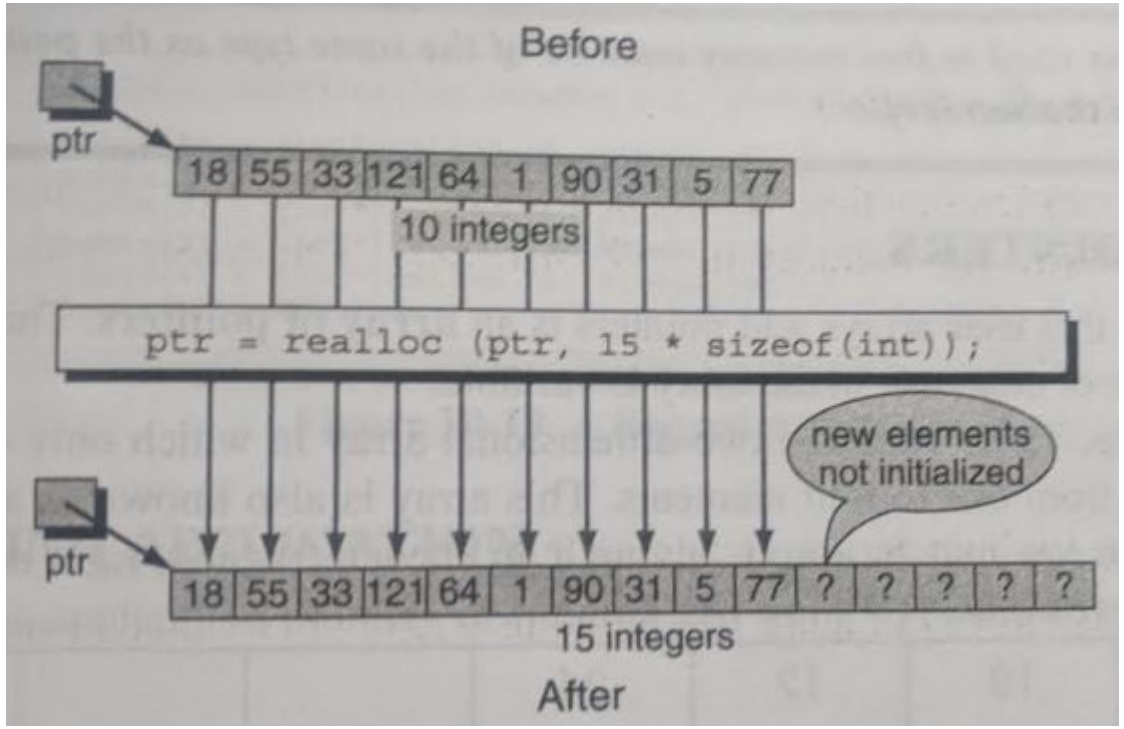
```
Enter total number of values: 3
0 0 0 Enter the values: 2 4 6
Entered values are: 2 4 6

...Program finished with exit code 0
Press ENTER to exit console.
```

realloc

void* realloc(pointer, new-size)

int *x;
x = (int*)malloc(50 * sizeof(int));
x = (int*)realloc(x,100);



```c

int main()
{
    int n, i, *ptr;
    printf("Enter total number of values: ");
    scanf("%d",&n);
    ptr=(int*)calloc(n, sizeof(int));
    printf("Enter the values: ");
    for(i=0;i<n;i++)
    {
        scanf("%d", (ptr+i));
    }
    printf("\n ENter updated size:");
    scanf("%d",&n);

    int *ptr1=(int*)realloc(ptr, n*sizeof(int));
    printf("Previous address= %p, new address = %p", ptr, ptr1);
    printf("\nEntered values are: ");
    for(i=0;i<n;i++)
    {
        printf("%d ",*(ptr+i));
    }
    free(ptr);
    return 0;
}

```

input

```
Enter total number of values: 3
Enter the values: 2 4 6

 ENter updated size:5
Previous address= 0x564f099cbac0, new address = 0x564f099cbac0
Entered values are: 2 4 6 0 0

...Program finished with exit code 0
Press ENTER to exit console.
```

```c
int main()
{
    int i, n;
    int *element;

    printf("Enter total number of elements: ");
    scanf("%d", &n);

    /*
        returns a void pointer(which is type-casted to int*)
        pointing to the first block of the allocated space
    */
    element = (int*) calloc(n,sizeof(int));

    /*
        If it fails to allocate enough space as specified,
        it returns a NULL pointer.
    */
    if(element == NULL)
    {
        printf("Error.Not enough space available");
        exit(0);
    }
    for(i = 0; i < n; i++)
    {
        /*
            storing elements from the user
            in the allocated space
        */
       scanf("%d", element+i);
    }
    for(i = 1; i < n; i++)
    {
       if(*element > *(element+i))
       {
            *element = *(element+i);
        }
    }

    printf("Smallest element is %d", *element);

    return 0;
}
```

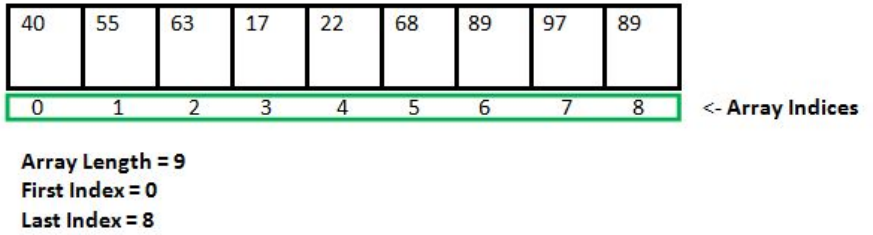
```
Enter total number of elements: 3
3
4
5
Smallest element is 3
```

Advanced Usage of Pointers:
Dynamic Storage Allocation, Linked Lists, and Trees

- Fixed size data structures have the same number of elements from compilation time for the whole structure lifetime:

```
int a[50]; //how many?
```

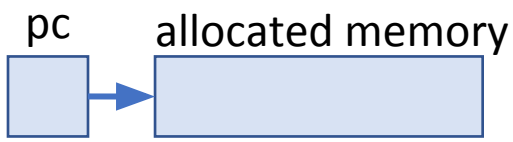
- Dynamic storage allocation: the ability to allocate storage during program execution



Memory allocation functions

- malloc: allocates a block of memory without initializing
- calloc: allocates a block of memory and clears it
- realloc: resizes a previously allocated block of memory
- free: deallocates the memory
- These functions are declared in the <stdlib.h>
- Take as input the number of bytes to allocate

Continued…



- Does malloc return int * or char *?
  - malloc does not know the type of data that will be stored in the block of memory so it returns a generic pointer: void *

```
char *pc = (char *) malloc(100);
int *pi = (int *) malloc (400);
```

- If allocation fails, malloc returns null pointer: NULL

The Null Pointer

• Since NULL is equivalent to 0, it tests false in if, while, do, and for statements:

```
int *p;
…
if (p) … /* true if p is not NULL */
```

• It is illegal to apply the indirection operator (*) to a null pointer:

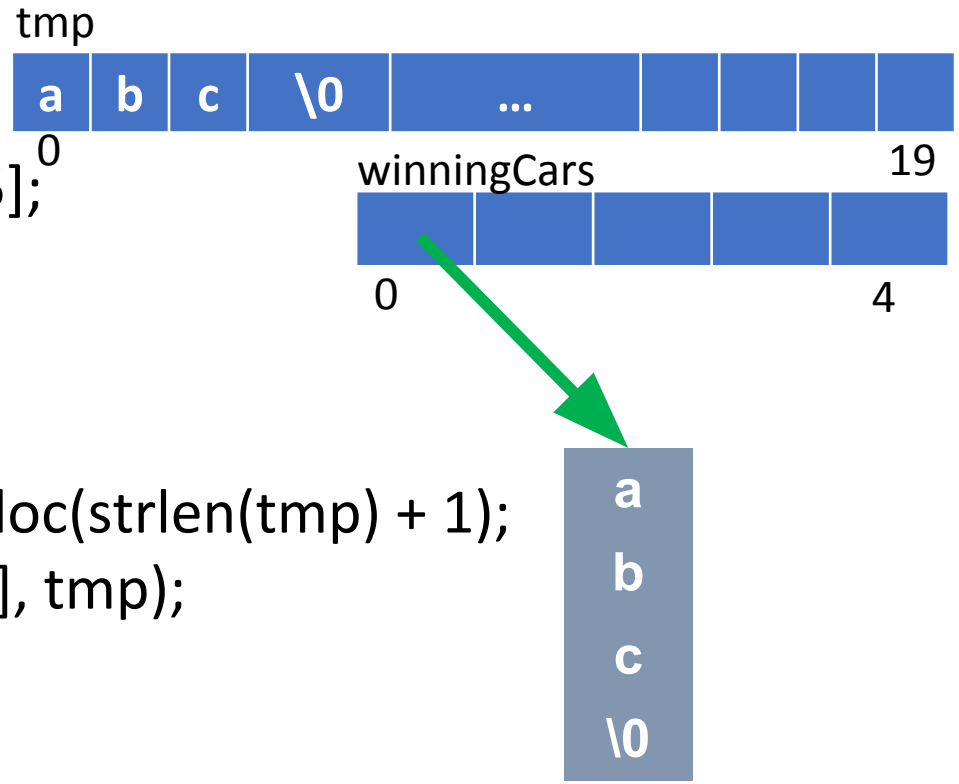
```
p = NULL;
i = *p; /* illegal */
```

Arrays of strings: An Example

```
char tmp[20];
char  *winningCars[5];
for (int i=0; i<5; i++) {
        scanf("%s", tmp);
        //Allocate memory for each pointer
        winningCars[i] = malloc(strlen(tmp) + 1);
        strcpy(winningCars[i], tmp);
}
```

Continued…

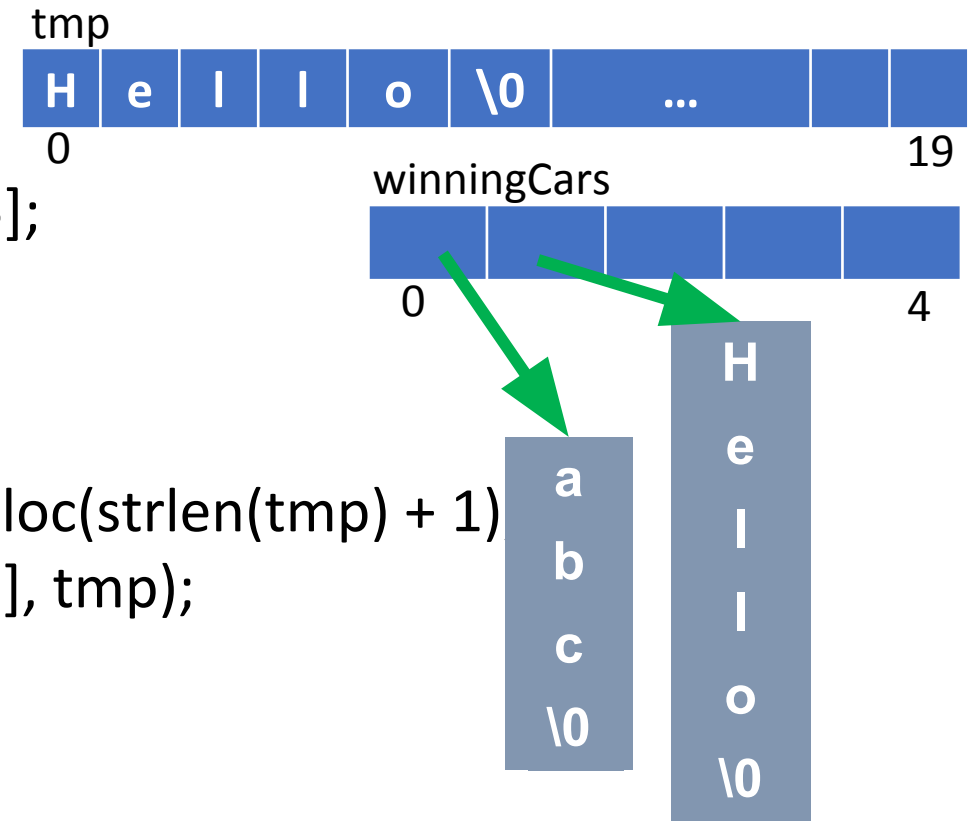
```
char tmp[20];
char * winningCars[5];
```




```
scanf("%s", tmp);
winningCars[0] = malloc(strlen(tmp) + 1);
strcpy(winningCars[0], tmp);
```

Continued…

```
char tmp[20];
char * winningCars[5];
```




```
scanf("%s", tmp);
winningCars[1] = malloc(strlen(tmp) + 1);
strcpy(winningCars[1], tmp);
```