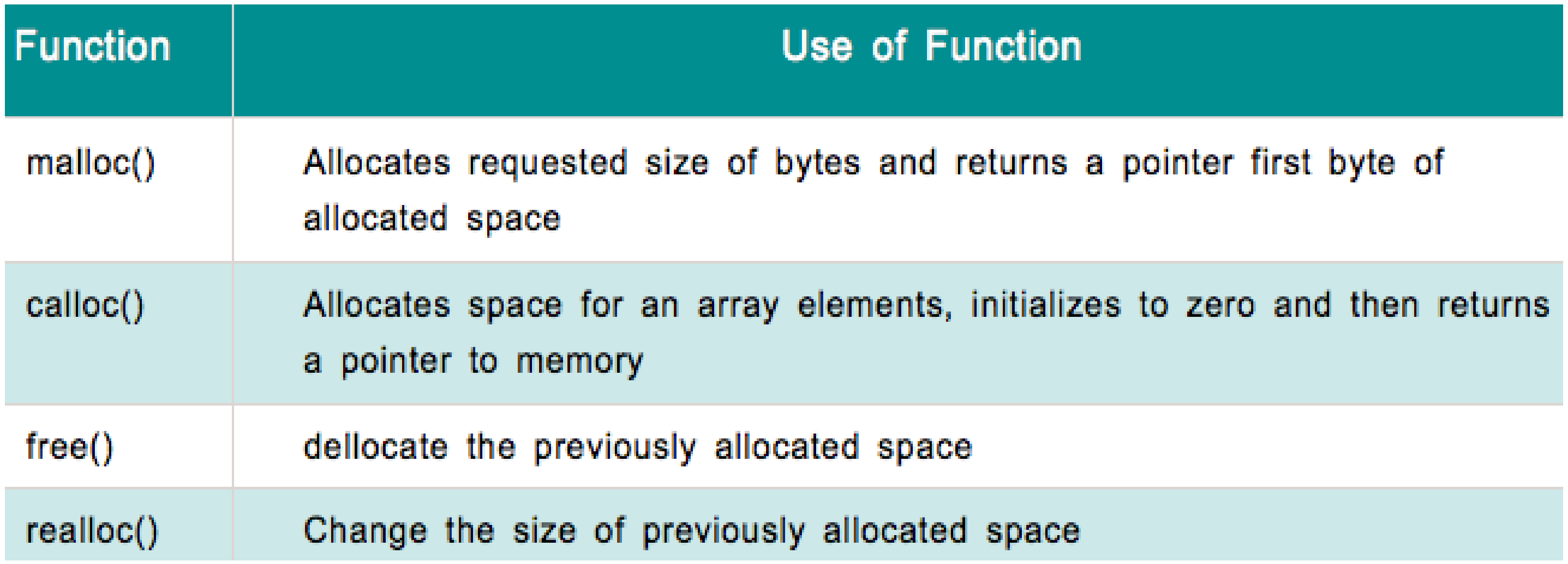
Dynamic Memory Allocation in C

Dr. Sheak Rashed Haider Noori
Associate Professor & Associate Head

Dynamic Memory Allocation in C

- In C language there are 4 library functions under "stdlib.h" for dynamic memory allocation.



Function	Use of Function
malloc()	Allocates requested size of bytes and returns a pointer first byte of allocated space
calloc()	Allocates space for an array elements, initializes to zero and then returns a pointer to memory
free()	dellocate the previously allocated space
realloc()	Change the size of previously allocated space

malloc()

- The name malloc stands for "memory allocation”. The function malloc() reserves a block of memory of specified size and return a pointer of type void which can be casted into pointer of any form.
- **Prototype:**

```
void* malloc(size in byte);
```

- **Syntax:**

```
ptr=(cast-type*)malloc(byte-size);
```

- Here, ptr is pointer of cast-type. The malloc() function returns a pointer to an area of memory with size of byte size. If the space is insufficient, allocation fails and returns NULL pointer.

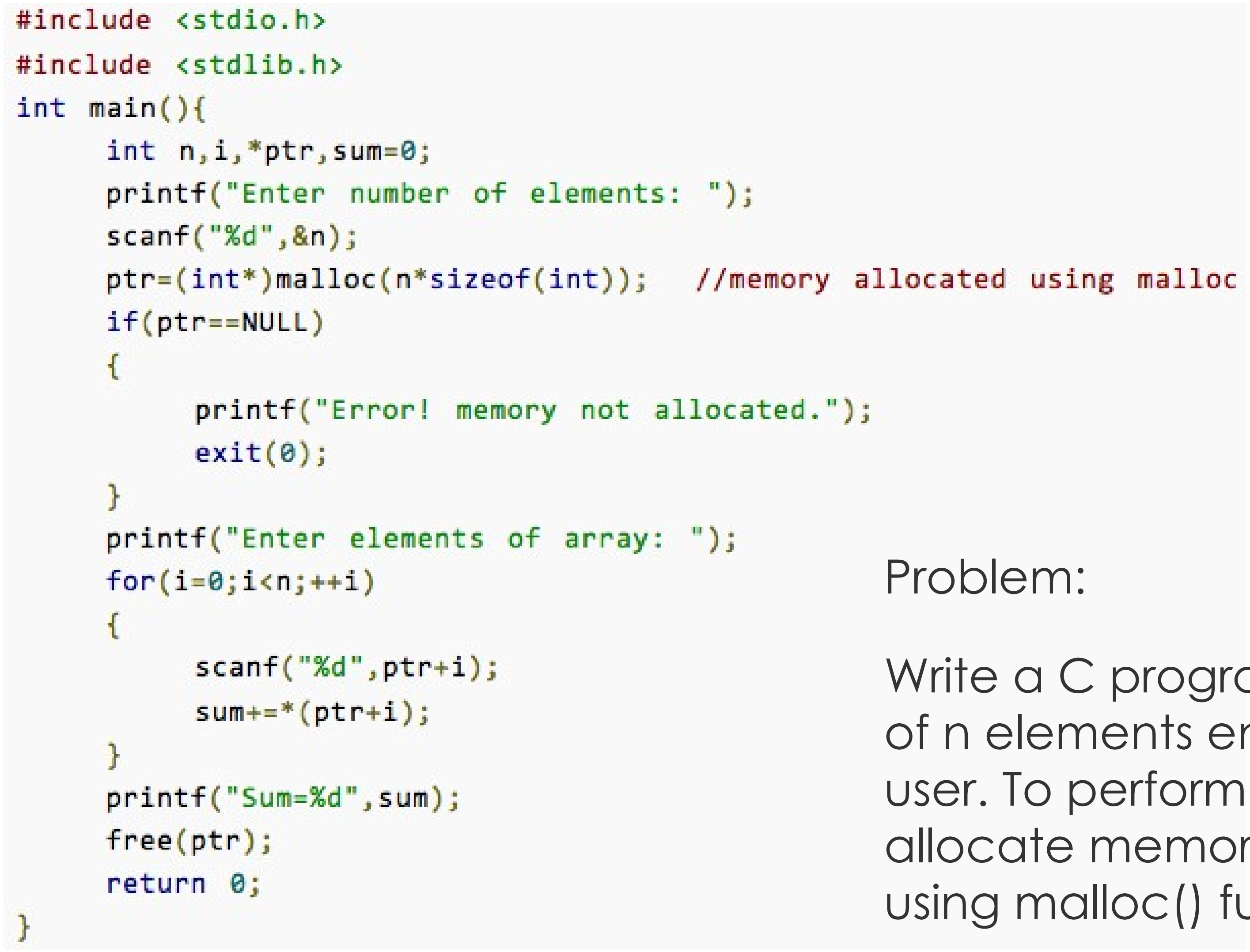
malloc() cont.

- **Example:**

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

- This statement will allocate either 200 or 400 according to size of int 2 or 4 bytes respectively and the pointer points to the address of first byte of memory.

Code Example of malloc()



```c
#include <stdio.h>
#include <stdlib.h>
int main(){
      int n,i,*ptr,sum=0;
      printf("Enter number of elements: ");
      scanf("%d",&n);
      ptr=(int*)malloc(n*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<n;++i)
      {
            scanf("%d",ptr+i);
            sum+=*(ptr+i);
      }
      printf("Sum=%d",sum);
      free(ptr);
      return 0;
}
```

Problem:

Write a C program to find sum of n elements entered by user. To perform this program, allocate memory dynamically using malloc() function.

calloc()

- The name calloc stands for "contiguous allocation".
- The difference between malloc() and calloc() is that, calloc() zero-initializes the memory blocks, while malloc() leaves the memory uninitialized.
- **Prototype:**

  void *calloc (no_of_blocks, size_of_each_block_in_bytes);
- **Syntax:**

  ptr=(cast-type*)calloc(no_of_blocks, size_of_each_block_in_bytes);
- This statement will allocate contiguous space in memory for an array of n elements.

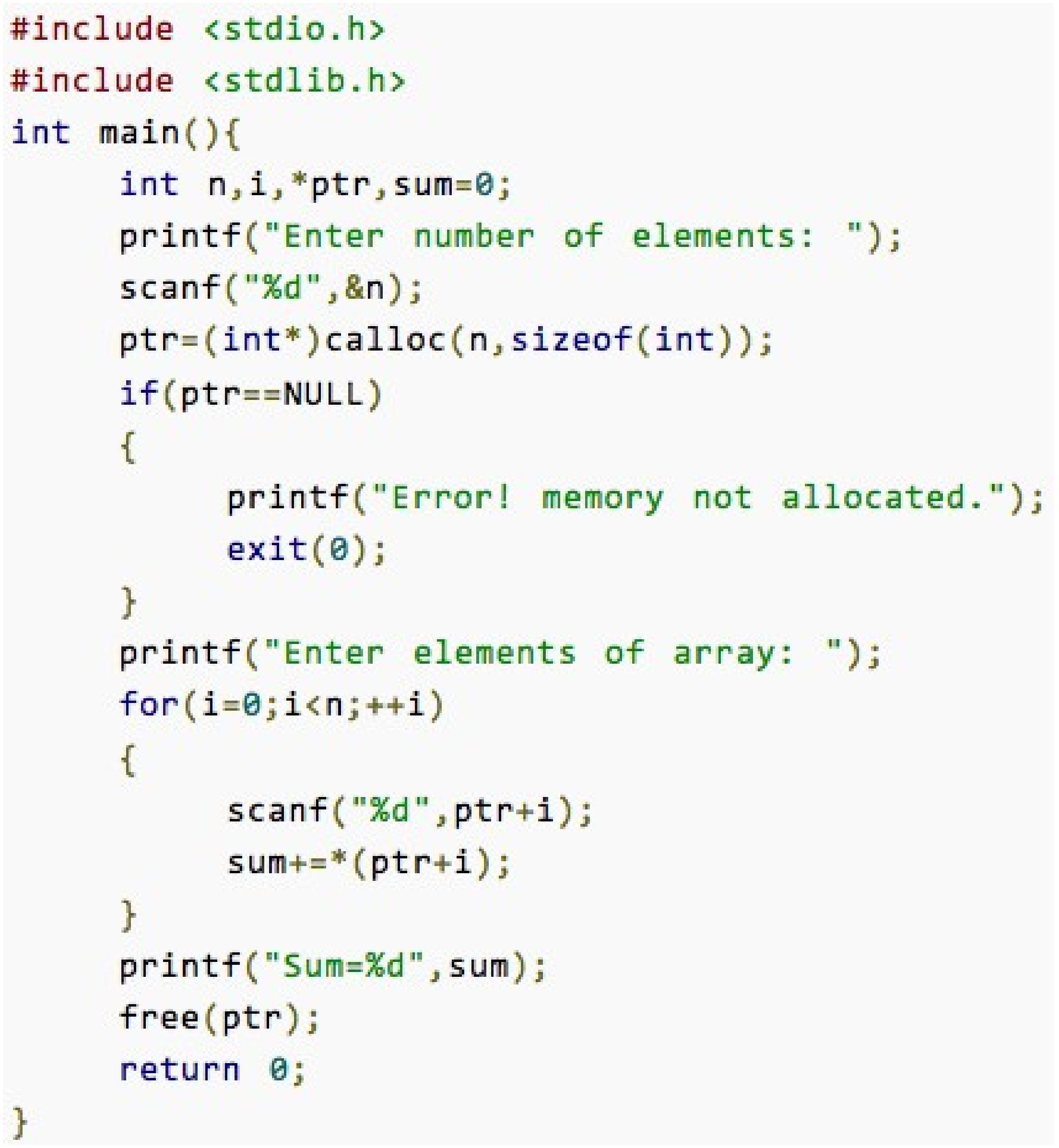
calloc() cont.

- **Example:**

```
float *ptr;
ptr=(float*)calloc(25, sizeof(float));
```

- This statement allocates contiguous space in memory for an array of 25 elements each of size of float, i.e, 4 bytes.

Code Example of calloc()



```c
#include <stdio.h>
#include <stdlib.h>
int main(){
      int n,i,*ptr,sum=0;
      printf("Enter number of elements: ");
      scanf("%d",&n);
      ptr=(int*)calloc(n,sizeof(int));
      if(ptr==NULL)
      {
            printf("Error! memory not allocated.");
            exit(0);
      }
      printf("Enter elements of array: ");
      for(i=0;i<n;++i)
      {
            scanf("%d",ptr+i);
            sum+=*(ptr+i);
      }
      printf("Sum=%d",sum);
      free(ptr);
      return 0;
}
```

Problem:

Write a C program to find sum of n elements entered by user. To perform this program, allocate memory dynamically using calloc() function.

free()

- Dynamically allocated memory with either calloc() or malloc() does not get return on its own.
- The programmer must use free() explicitly to release space taken from the Heap.
- **Prototype:**

```
void  free(pointer to heap memory);
```

- **Example:**

```
free(ptr);
```

- This statement cause the space in memory pointer by ptr to be deallocated.

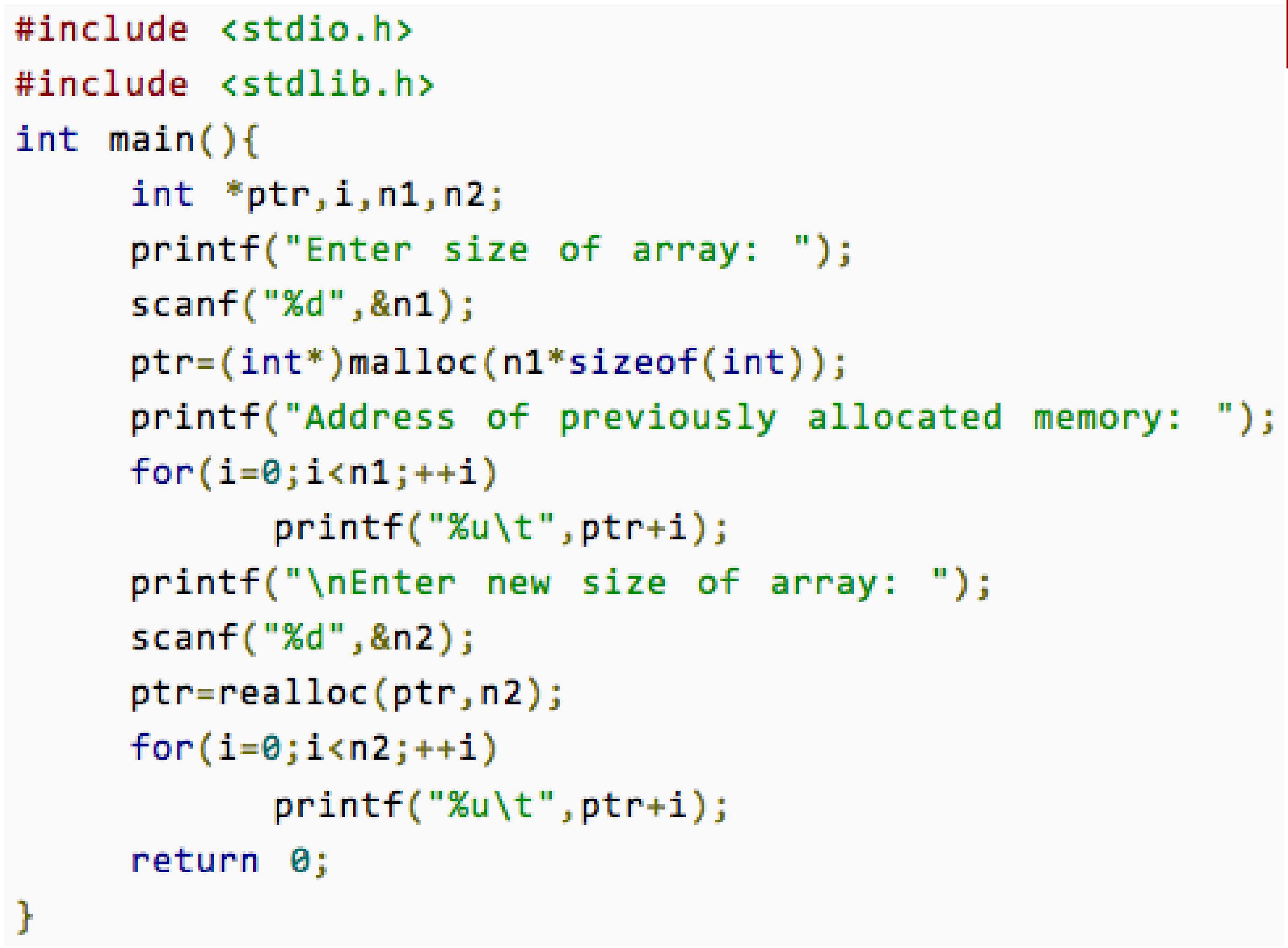
realloc()

- The realloc() function changes the size of a block of memory that was previously allocated with malloc() or calloc().
- If the previously allocated memory is insufficient or more than sufficient. Then, you can change memory size previously allocated using realloc().
- **Prototype:**

```
void *realloc(void *ptr, size_t size);
```

- The ptr argument is a pointer to the original block of memory. The new size, in bytes, is specified by size.

Code Example of realloc()



```
#include <stdio.h>
#include <stdlib.h>
int main(){
      int *ptr,i,n1,n2;
      printf("Enter size of array: ");
      scanf("%d",&n1);
      ptr=(int*)malloc(n1*sizeof(int));
      printf("Address of previously allocated memory: ");
      for(i=0;i<n1;++i)
             printf("%u\t",ptr+i);
      printf("\nEnter new size of array: ");
      scanf("%d",&n2);
      ptr=realloc(ptr,n2);
      for(i=0;i<n2;++i)
             printf("%u\t",ptr+i);
      return 0;
}
```