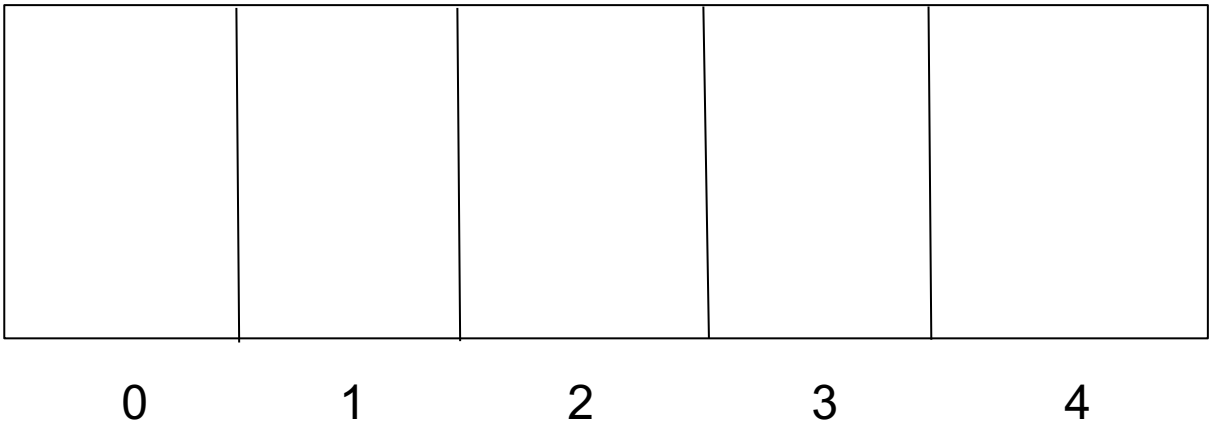
C Programming

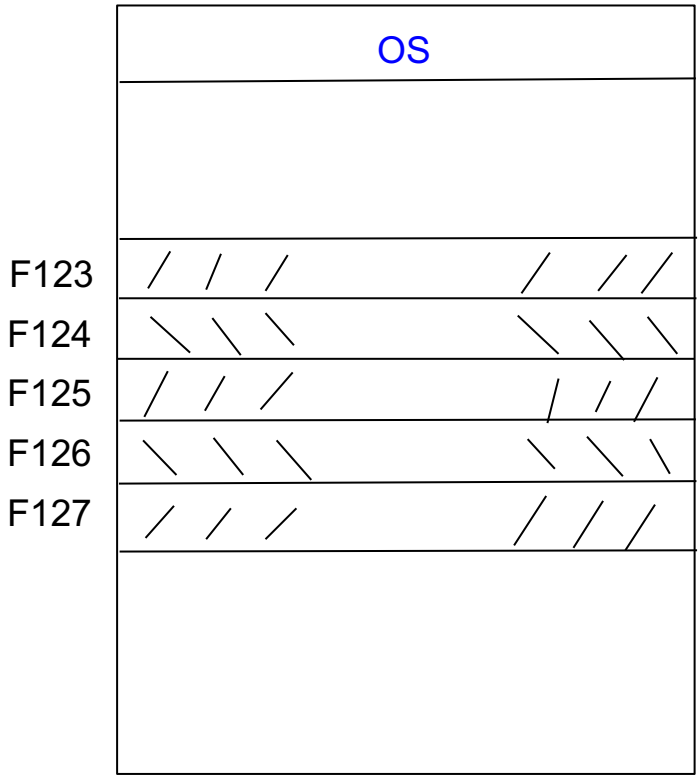
Dynamic Memory Allocation (DMA)

We have seen how to declare an array and how it is stored in memory.

e.g.,

int a[5];





There will be situations when you don't know how much memory is required when a program will run. In these situations, DMA is the best course to allocate memory while the program is running.

DMA is provided in 2 ways:

1. **The malloc() function**

malloc() allocates a contiguous block of memory and returns a pointer to the start of the allocated block.

```
pointer = malloc( size );
```

- pointer = pointer contains the address of the **start** of the allocated memory block
- size = the total number of **bytes** required for the memory block

Note: once the memory block is allocated, it contains random data

e.g.,

int *ptr;

ptr = malloc(40);

```
/*
Dynamic Memory Allocation (DMA)

This program uses malloc() to dynamically allocate a block of memory,
enter data into the memory block and display it. The memory block is
freed (released) at the end when the memory block is no longer
required.
*/

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

int main()
{
	int *ptr;
	int numbers = 0;
	int no_bytes = 0;
	int i;


	// Part 1
	// How many numbers, i.e., data items do you wish to enter
	printf("\nHow many numbers will you enter?\n");
	scanf("%d", &numbers);

	// Part 2
	// Calculate the number of bytes required to store the set of
numbers in memory
	no_bytes = numbers * sizeof(int);

	// Part 3
	// Allocate the block of memory required
	ptr = malloc(no_bytes);

	// Part 4
```

```
	// Check if malloc was successful, i.e., check if the memory was
allocated successfully
	if(ptr == NULL)
	{
		printf("\nFailed to allocate memory\n");
	} // end if
	else // memory allocated successfully
	{
		// Part 5
		// memory allocated successfully - go and use it
		printf("\nMemory allocated successfully\n");
		printf("\nEnter the set of %d numbers\n", numbers);

		// Enter data items into the allocated memory block
		for(i = 0; i < numbers; i++)
		{
			scanf("%d", & *(ptr + i));

		} // end for

		printf("\nYou entered:\n");
		// Display the data items entered into the allocated memory
block
		for(i = 0; i < numbers; i++)
		{
			printf("%d\n", *(ptr + i));

		} // end for

		// Part 6
		// Free the allocated memory block once finished using it
		free(ptr);

	} // end else


 return 0;

} // end main()
```

Repl 11.1: https://replit.com/@michaelTUDublin/111-malloc

2. **The calloc() function**

calloc() also allocates a contiguous block of memory and returns a pointer to the start of the allocated block.

pointer = calloc(number_of_data_items, size_of_each_data_item);

- pointer = pointer contains the address of the start of the allocated memory block
- number_of_data_items = total number of data that you wish to be stored
- size = the size of each individual data item, e.g., 4 (integer), 2 (char)

Note: once the memory block is allocated, **the OS initialises the block all to contain 0**

e.g.,

```
int *ptr;
int no_of_numbers = 5;

ptr = calloc(no_of_numbers, 4);
```

```
/*
Dynamic Memory Allocation (DMA)

This program uses calloc() to dynamically allocate a block of memory,
enter data into the memory block and display it. The memory block is
freed (released) at the end when the memory block is no longer
required.
*/

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

int main()
```

```
{
 int *ptr;
 int numbers = 0;
 //int no_bytes = 0;
 int i;


 // Part 1
 // How many numbers, i.e., data items do you wish to enter
 printf("\nHow many numbers will you enter?\n");
 scanf("%d", & numbers);

 // Part 2 NOT NEEDED
 // Calculate the number of bytes required to store the set of numbers
in memory
 //no_bytes = numbers * sizeof(int);

 // Part 3
 // Allocate the block of memory required
 ptr = calloc(numbers, sizeof(int));

 // Part 4
 // Check if malloc was successful, i.e., check if the memory was
allocated successfully
 if(ptr == NULL)
 {
   printf("\nFailed to allocate memory\n");
 } // end if
 else // memory allocated successfully
 {
   // Part 5
   // memory allocated successfully - go and use it
   printf("\nMemory allocated successfully\n");
   printf("\nEnter the set of %d numbers\n", numbers);

   // Enter data items into the allocated memory block
   for(i = 0; i < numbers; i++)
   {
     scanf("%d", & *(ptr + i));
```

```
    } // end for


    printf("\nYou entered:\n");
    // Display the data items entered into the allocated memory block
    for(i = 0; i < numbers; i++)
    {
      printf("%d\n", *(ptr + i));

    } // end for

    // Part 6
    // Free the allocated memory block once finished using it
    free(ptr);

  } // end else

  return 0;

} // end main()
```

Repl 11.2: https://replit.com/@michaelTUDublin/112-calloc#main.c

3. The realloc() function

The realloc() function is used to change the size of an **already dynamically allocated block of memory**.

realloc() is used as follows:

```
pointer = realloc( pointer, new_total_size_of_block);
```

- pointer = pointer that is pointing at the start of the existing memory block
- new_total_size_of_block = total size of the increased/decreased block in bytes

Let's take the code above with calloc() and increase the size of the block to allow additional numbers to be entered

```
/*
Dynamic Memory Allocation (DMA)
This program uses calloc() to dynamically allocate a block of memory,
enter data into the memory block and display it.

realloc() is then used to increase the size of the block and allow the
user to enter additional numbers

The memory block is freed (released) at the end when the memory block
is no longer required.
*/
#include <stdio.h>
#include <stdlib.h>

int main()
{
     int *ptr;
     int numbers = 0;
     //int no_bytes = 0;
     int i;

     //Needed for realloc()
     char answer = 'n';
     int extra_data = 0;
     int new_block_size = 0;


     // Part 1
     // How many numbers, i.e., data items do you wish to enter
     printf("\nHow many numbers will you enter?\n");
     scanf("%d", & numbers);

     // Part 2 NOT NEEDED
```

```
	// Calculate the number of bytes required to store the set of
numbers in memory
	//no_bytes = numbers * sizeof(int);

	// Part 3
	// Allocate the block of memory required
	ptr = calloc(numbers, sizeof(int));

	// Part 4
	// Check if malloc was successful, i.e., check if the memory was
allocated successfully
	if(ptr == NULL)
	{
		printf("\nFailed to allocate memory\n");

	} // end if
	else // memory allocated successfully
	{

	// Part 5
		// memory allocated successfully - go and use it
		printf("\nMemory allocated successfully\n");
		printf("\nEnter the set of %d numbers\n", numbers);

		// Enter data items into the allocated memory block
		for(i = 0; i < numbers; i++)
		{
			scanf("%d", & *(ptr + i));
		} // end for

		printf("\nYour memory block contains:\n");

		// Display the data items entered into the allocated memory
	 block
		for(i = 0; i < numbers; i++)
		{
			printf("%d  %p\n", *(ptr + i), (ptr + i) );
		} // end for
```

```
            //Part 6
            //Ask the user if they wish to enter additional numbers
            printf("\nEnter more numbers (y/n)\n");
            scanf("%1s", & answer);

            //Check answer
            if(answer == 'n')
            {
                  printf("\nNo changes - memory block remains the
same\n");
            } // end if
            else
            {
                  printf("\nHow many extra numbers to enter?\n");
                  scanf("%d", & extra_data);

                  // Calculate the total size of the new memory block
needed to store the extra data
                  new_block_size = (numbers + extra_data) * sizeof(int);

                  //change the size of the allocated memory block to
include the extra numbers
                  ptr = realloc(ptr, new_block_size);

                  //Check if the memory can be expanded
                  if(ptr == NULL)
                  {
                        printf("\nFailed to EXPAND memory block foe new
data\n");

                  } // end if
                  else
                  {
                        printf("\nEnter the additional data items\n");

                        // Enter data items into the allocated memory
block
                        for(i = numbers; i < numbers + extra_data; i++)
                        {
```

```
                        scanf("%d", & *(ptr + i));

                    } // end for

                    printf("\nYour memory block contains:\n");

                    // Display the data items entered into the re-
                allocated memory block
                    for(i = 0; i < numbers + extra_data; i++)
                    {
                        printf("%d  %p\n", *(ptr + i), (ptr + i) );
                    } // end for

                    } // end else

            } // end else

            // Part 7
            // Free the allocated memory block once finished using it
            free(ptr);
        } // end else

return 0;

} // end main()
```

Repl 11.3: https://replit.com/@michaelTUDublin/113-realloc