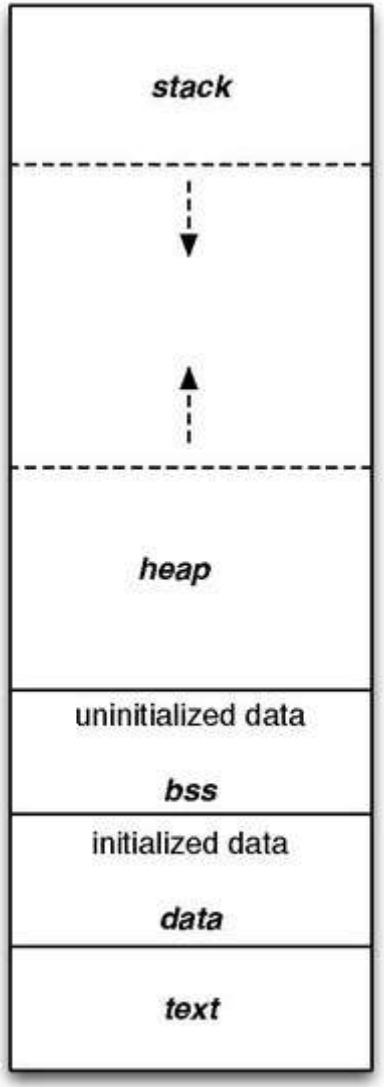
Dynamic Allocation

Jinwoo Choi

Dynamic Allocation(동적할당)

Dynamic Allocation



sizes of memory determined at compile time

sizes of memory determined at run time

sizes of memory determined at compile time

malloc

```
#include  <stdlib.h>

void* malloc(size_t size);
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    .
    .
    .
    return 0;
}
```

Address	Memory
0x10	
0x11	
0x12	
0x13	
0x14	
0x15	
0x16	
0x17	
0x18	
0x19	
0x1A	
0x1B	
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

malloc

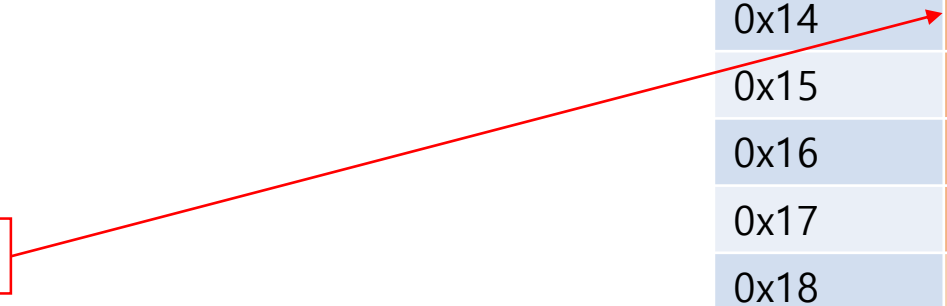
```
#include  <stdlib.h>

void* malloc(size_t size);
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    .
    .
    .
    return 0;
}
```



Address	Memory
0x10	
0x11	
0x12	
0x13	
0x14	
0x15	
0x16	
0x17	
0x18	
0x19	
0x1A	
0x1B	
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

malloc

```
#include  <stdlib.h>

void* malloc(size_t size);
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    .
    .
    .
    return 0;
}
```

Address	Memory
0x10	
0x11	
0x12	
0x13	
0x14	
0x15	
0x16	
0x17	
0x18	
0x19	
0x1A	
0x1B	
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

malloc

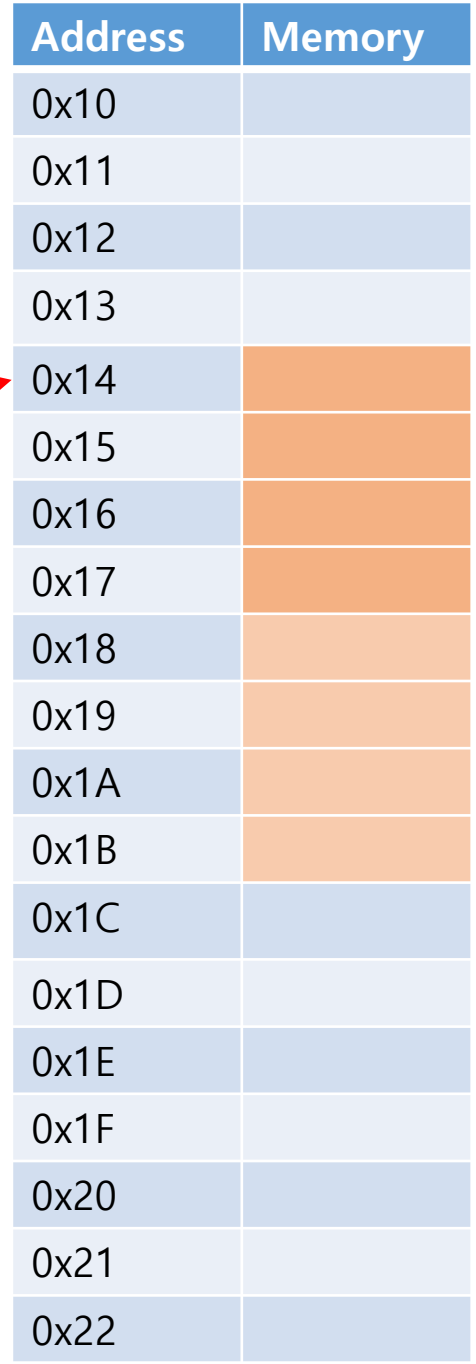
```
#include  <stdlib.h>

void* malloc(size_t size);
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    .
    .
    .
    return 0;
}
```



Address	Memory
0x10	
0x11	
0x12	
0x13	
0x14	
0x15	
0x16	
0x17	
0x18	
0x19	
0x1A	
0x1B	
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

malloc

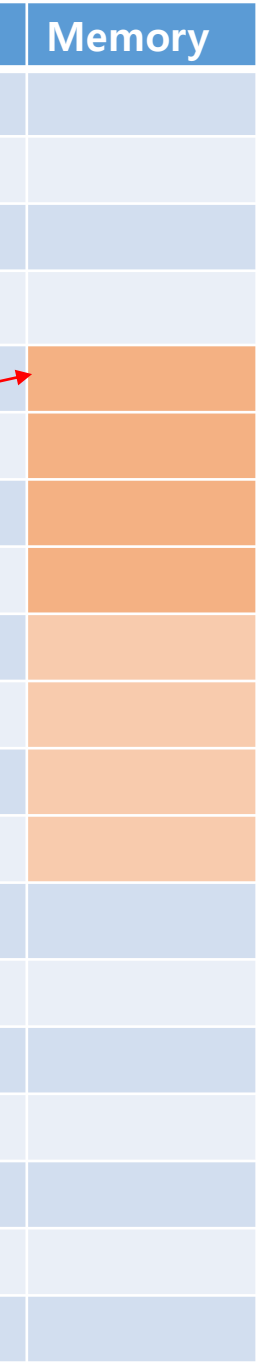
```
#include  <stdlib.h>

void* malloc(size_t size);
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    .
    .
    .
    return 0;
}
```



malloc

```
#include  <stdlib.h>

void* malloc(size_t size);
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    .
    .
    .
    return 0;
}
```

Address	Memory
0x10	
0x11	
0x12	
0x13	
0x14	01011111
0x15	00000100
0x16	00000000
0x17	00000000
0x18	
0x19	
0x1A	
0x1B	
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

malloc

```
#include  <stdlib.h>

void* malloc(size_t size);
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    .
    .
    .
    return 0;
}
```

Address	Memory
0x10	
0x11	
0x12	
0x13	
0x14	01011111
0x15	00000100
0x16	00000000
0x17	00000000
0x18	
0x19	
0x1A	
0x1B	
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

malloc

```
#include  <stdlib.h>

void* malloc(size_t size);
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    .
    .
    .
    return 0;
}
```

Address	Memory
0x10	
0x11	
0x12	
0x13	
0x14	01011111
0x15	00000100
0x16	00000000
0x17	00000000
0x18	
0x19	
0x1A	
0x1B	
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

malloc

```
#include  <stdlib.h>

void* malloc(size_t size);
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    .
    .
    .
    return 0;
}
```

Address	Memory
0x10	
0x11	
0x12	
0x13	
0x14	01011111
0x15	00000100
0x16	00000000
0x17	00000000
0x18	
0x19	
0x1A	
0x1B	
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

malloc

```
#include  <stdlib.h>

void* malloc(size_t size);
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    .
    .
    .
    return 0;
}
```

Address	Memory
0x10	
0x11	
0x12	
0x13	
0x14	01011111
0x15	00000100
0x16	00000000
0x17	00000000
0x18	10011010
0x19	00000011
0x1A	00000000
0x1B	00000000
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

malloc

```
#include  <stdlib.h>

void* malloc(size_t size);
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    .
    .
    .
    return 0;
}
```

Address	Memory	
0x10		
0x11		
0x12		
0x13		
0x14	01011111	arr[0]
0x15	00000100	
0x16	00000000	
0x17	00000000	
0x18	10011010	arr[1]
0x19	00000011	
0x1A	00000000	
0x1B	00000000	
0x1C		
0x1D		
0x1E		
0x1F		
0x20		
0x21		
0x22		

malloc

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d %d\n", arr[0], arr[1]);
```

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(0 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d, %d\n", arr[0], arr[1], arr[2]);
```

Even though this code compiles without error or warning, we can't garuntee the behavior

malloc(0)
assigning value into invalid memory

calloc

```
#include  <stdlib.h>

void* calloc(size_t num_elements, size_t element_size);
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)calloc(2, sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    .
    .
    .
    return 0;
}
```

Address	Memory
0x10	
0x11	
0x12	
0x13	
0x14	00000000
0x15	00000000
0x16	00000000
0x17	00000000
0x18	
0x19	
0x1A	
0x1B	
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

realloc

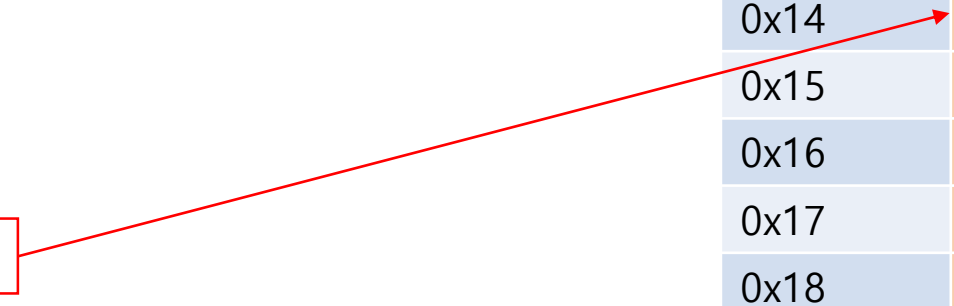
```
#include  <stdlib.h>

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

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    arr = (int*)realloc(arr, 3 * sizeof(int));
    .
    .
    .
    return 0;
}
```



Address	Memory
0x10	
0x11	
0x12	
0x13	
0x14	
0x15	
0x16	
0x17	
0x18	
0x19	
0x1A	
0x1B	
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

realloc

```
#include  <stdlib.h>

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

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    arr = (int*)realloc(arr, 3 * sizeof(int));
    .
    .
    .
    return 0;
}
```

Address	Memory
0x10	
0x11	
0x12	
0x13	
0x14	01011111
0x15	00000100
0x16	00000000
0x17	00000000
0x18	10011010
0x19	00000011
0x1A	00000000
0x1B	00000000
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

realloc

```
#include  <stdlib.h>

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

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    arr = (int*)realloc(arr, 3 * sizeof(int));
    .
    .
    .
    return 0;
}
```

Address	Memory
0x10	01011111
0x11	00000100
0x12	00000000
0x13	00000000
0x14	10011010
0x15	00000011
0x16	00000000
0x17	00000000
0x18	
0x19	
0x1A	
0x1B	
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

free

```
#include  <stdlib.h>

void free(void* ptr);
```

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    arr = (int*)realloc(arr, 3 * sizeof(int));

    free(arr);

    return 0;
}
```

Address	Memory
0x10	01011111
0x11	00000100
0x12	00000000
0x13	00000000
0x14	10011010
0x15	00000011
0x16	00000000
0x17	00000000
0x18	
0x19	
0x1A	
0x1B	
0x1C	
0x1D	
0x1E	
0x1F	
0x20	
0x21	
0x22	

free

```
#include  <stdlib.h>

void free(void* ptr);
```

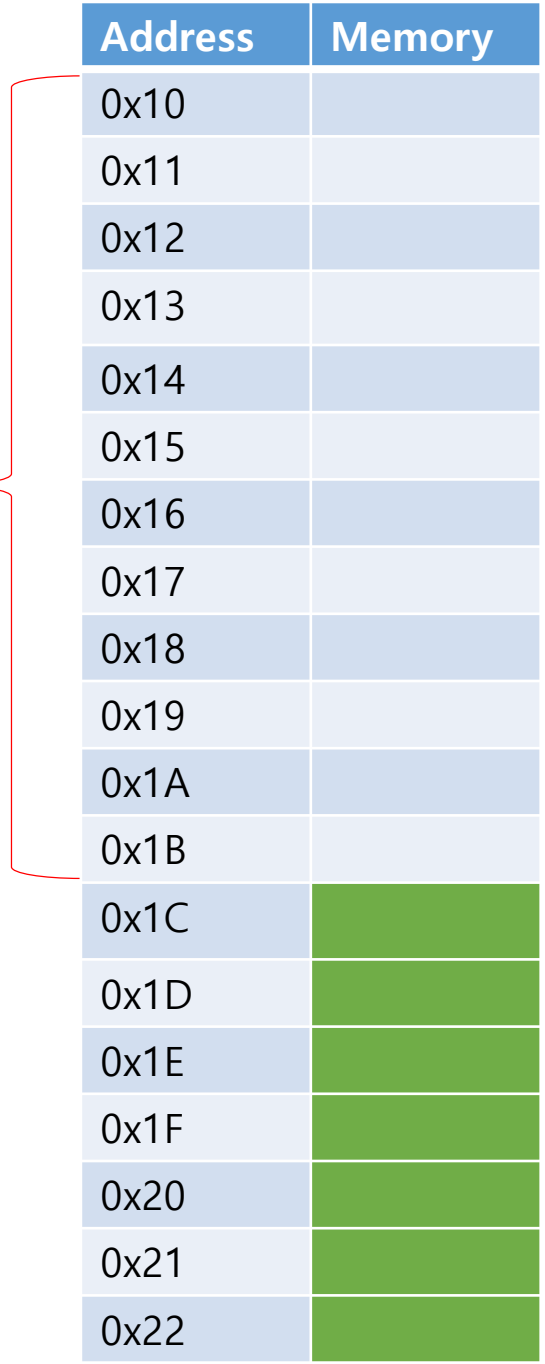
```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int* arr = (int*)malloc(2 * sizeof(int));

    *arr = 1103;
    *(arr + 1) = 906;

    printf("%d, %d\n", arr[0], arr[1]);
    arr = (int*)realloc(arr, 3 * sizeof(int));

    free(arr);

    return 0;
}
```



LAB – DynamicArray

- Create a file named 'DynamicArray_YourName.c'
- Dynamically allocate memory for an array of 5 integers using malloc
- Assign values to the array elements using pointer arithmetic (e.g. *(arr+1))
- Print the values using both pointer and array notation
- Free the allocated memory at the end of the program

LAB – DynamicStrings

- Create a file named 'DynamicStrings_YourName.c'
- Use calloc to allocate memory for a string of 15 characters
- Copy a string value into that memory(e.g. "Hello World")
- Print the string using the pointer
- Use realloc to resize the memory to hold a longer string
- Copy a longer string (e.g. "Hello Longer String!") into the resized memory
- Print the new string and free the memory at the end

Double Pointer

```
int rows = 3;
int cols = 4;

int** arr = (int**)malloc(rows * sizeof(int*));

for (int i = 0; i < rows; i++)
{
    arr[i] = (int*)malloc(cols * sizeof(int));
}
```

Address	Memory	Address	Memory
0x10		0x5C	
0x14		0x60	
0x18		0x64	
0x1C		0x68	
0x20		0x6C	
0x24		0x70	
0x28		0x74	
0x2C		0x78	
0x30		0x7C	
0x34		0x80	
0x38		0x84	
0x3C		0x88	
0x40		0x8C	
0x44		0x90	
0x48		0x94	
0x4C		0x98	
0x50		0x9C	
0x54		0xA0	
0x58		0xA4	

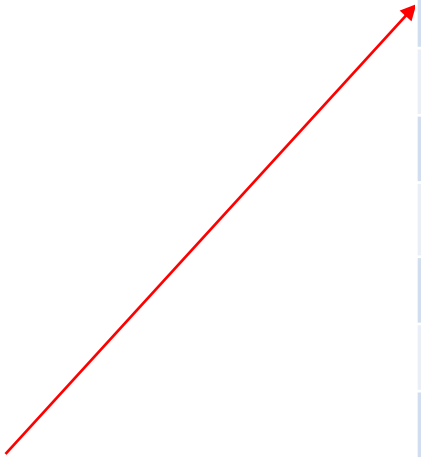
Double Pointer

```
int rows = 3;
int cols = 4;

int** arr = (int**)malloc(rows * sizeof(int*));

for (int i = 0; i < rows; i++)
{
    arr[i] = (int*)malloc(cols * sizeof(int));
}
```

int **
int *
int *
int *



Address	Memory	Address	Memory
0x10		0x5C	
0x14		0x60	
0x18		0x64	
0x1C		0x68	
0x20		0x6C	
0x24		0x70	
0x28		0x74	
0x2C		0x78	
0x30		0x7C	
0x34		0x80	
0x38		0x84	
0x3C		0x88	
0x40		0x8C	
0x44		0x90	
0x48		0x94	
0x4C		0x98	
0x50		0x9C	
0x54		0xA0	
0x58		0xA4	

Double Pointer

```
int rows = 3;
int cols = 4;

int** arr = (int**)malloc(rows * sizeof(int*));

for (int i = 0; i < rows; i++)
{
    arr[i] = (int*)malloc(cols * sizeof(int));
}
```

int **
int *
int *
int *

Address	Memory	Address	Memory
0x10		0x5C	
0x14		0x60	
0x18		0x64	
0x1C		0x68	
0x20		0x6C	
0x24		0x70	
0x28		0x74	
0x2C		0x78	
0x30		0x7C	
0x34		0x80	
0x38		0x84	
0x3C		0x88	
0x40		0x8C	
0x44		0x90	
0x48		0x94	
0x4C		0x98	
0x50		0x9C	
0x54		0xA0	
0x58		0xA4	

Double Pointer

```
int rows = 3;
int cols = 4;

int** arr = (int**)malloc(rows * sizeof(int*));

for (int i = 0; i < rows; i++)
{
    arr[i] = (int*)malloc(cols * sizeof(int));
}
```

arr[0]

int **				
int *	int	int	int	int
int *				
int *				

Address	Memory	Address	Memory
0x10	0x28	0x5C	
0x14		0x60	
0x18		0x64	
0x1C		0x68	
0x20		0x6C	
0x24		0x70	
0x28		0x74	
0x2C		0x78	
0x30		0x7C	
0x34		0x80	
0x38		0x84	
0x3C		0x88	
0x40		0x8C	
0x44		0x90	
0x48		0x94	
0x4C		0x98	
0x50		0x9C	
0x54		0xA0	
0x58		0xA4	

Double Pointer

```
int rows = 3;
int cols = 4;

int** arr = (int**)malloc(rows * sizeof(int*));

for (int i = 0; i < rows; i++)
{
    arr[i] = (int*)malloc(cols * sizeof(int));
}
```

arr[1]

int **				
int *	int	int	int	int
int *	int	int	int	int
int *				

Address	Memory	Address	Memory
0x10	0x28	0x5C	
0x14		0x60	
0x18	0x40	0x64	
0x1C		0x68	
0x20		0x6C	
0x24		0x70	
0x28		0x74	
0x2C		0x78	
0x30		0x7C	
0x34		0x80	
0x38		0x84	
0x3C		0x88	
0x40		0x8C	
0x44		0x90	
0x48		0x94	
0x4C		0x98	
0x50		0x9C	
0x54		0xA0	
0x58		0xA4	

Double Pointer

```
int rows = 3;
int cols = 4;

int** arr = (int**)malloc(rows * sizeof(int*));

for (int i = 0; i < rows; i++)
{
    arr[i] = (int*)malloc(cols * sizeof(int));
}
```

arr[2]

int **				
int *	int	int	int	int
int *	int	int	int	int
int *	int	int	int	int

Address	Memory	Address	Memory
0x10	0x28	0x5C	
0x14		0x60	
0x18	0x40	0x64	
0x1C		0x68	
0x20	0x80	0x6C	
0x24		0x70	
0x28		0x74	
0x2C		0x78	
0x30		0x7C	
0x34		0x80	
0x38		0x84	
0x3C		0x88	
0x40		0x8C	
0x44		0x90	
0x48		0x94	
0x4C		0x98	
0x50		0x9C	
0x54		0xA0	
0x58		0xA4	

Double Pointer

```
int rows = 3;
int cols = 4;

int** arr = (int**)malloc(rows * sizeof(int*));

for (int i = 0; i < rows; i++)
{
    arr[i] = (int*)malloc(cols * sizeof(int));
}
```

int **				
int *	int	int	int	int
int *	int	int	int	int
int *	int	int	int	int

Address	Memory	Address	Memory
0x10	0x28	0x5C	
0x14		0x60	
0x18	0x40	0x64	
0x1C		0x68	
0x20	0x80	0x6C	
0x24		0x70	
0x28		0x74	
0x2C		0x78	
0x30		0x7C	
0x34		0x80	
0x38		0x84	
0x3C		0x88	
0x40		0x8C	
0x44		0x90	
0x48		0x94	
0x4C		0x98	
0x50		0x9C	
0x54		0xA0	
0x58		0xA4	

LAB – Dynamic2DArray

- Create a file named 'Dynamic2DArray_YourName.c'
- Define two variables(*rows, columns*) with MACRO(preprocessor)
- Dynamically allocate memory for a 2D integer matrix using int**
  - Allocate memory for an array of int*(number of *rows*)
  - For each row, allocate memory for *columns* number of int
- Write a function **void fillMatrix(int** matrix, int rows, int cols)** that
  - Fills the matrix such that element [i][j] = (i + 1) * (j + 1)
- Write a function **void printMatrix(int** matrix, int rows, int cols)** that
  - Prints the matrix in table format
- Free all allocated memory at the end of the program