Introduction

Week 03 – Tutorial

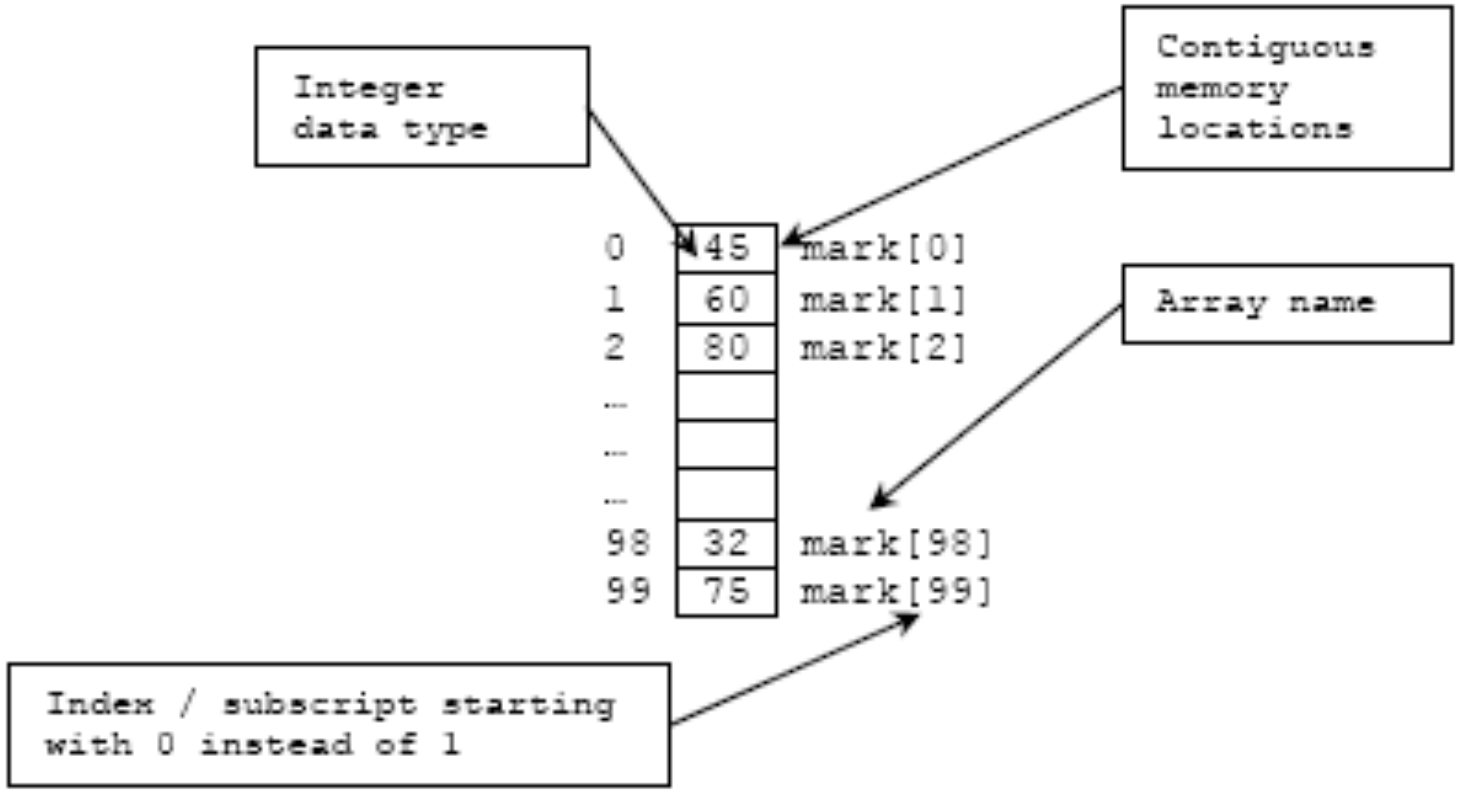
C Language: Pointers, Arrays & Structures

Goal: programming skills

- Able to understand, create and use
  - arrays
  - pointers

Arrays

int mark[100];



Array initialisation

```
int    id[7] = {1, 2, 3, 4, 5, 6, 7};
float  x[5] = {5.6, 5.7, 5.8, 5.9, 6.1};
char   vowel[6] = {'a', 'e', 'i', 'o', 'u', '\0'};
```

Arrays and functions

A function can receive the address of an array by using:

- A pointer.
- A sized array (dimension is explicitly stated), e.g. s[20] or
- An unsized array (dimension is not stated), e.g. p[ ].

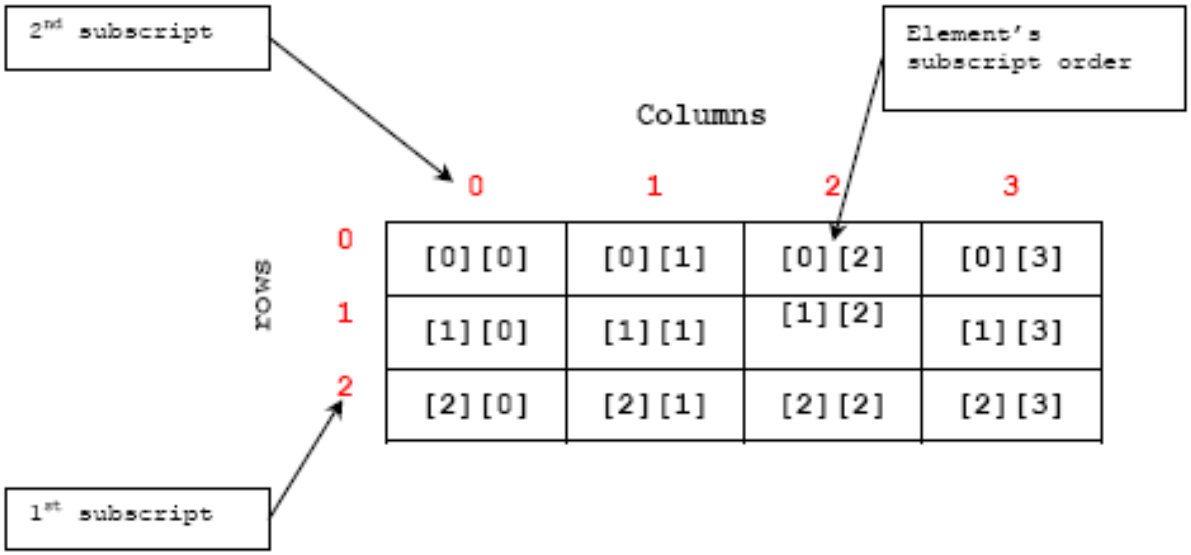
Code

Call a function f with:

- A pointer.
- A sized array (dimension is explicitly stated), e.g. s[20] or
- An unsized array (dimension is not stated), e.g. p[ ].

Multidimensional arrays

```
int     x[3][4];
float   matrix[20][25];
```

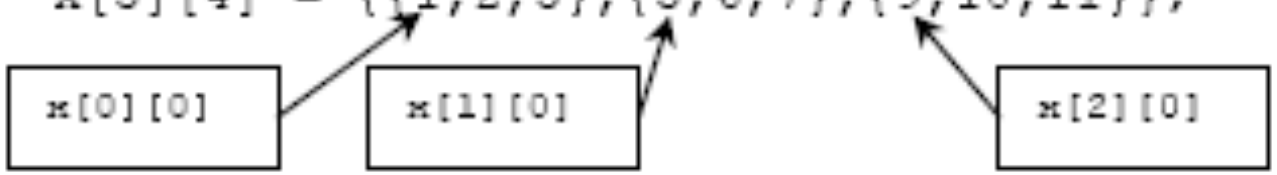


Order of elements in memory

- All multidimensional arrays are stored in linear order row by row:

```
int   x[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

```
int   x[3][4] = {{1,2,3},{5,6,7},{9,10,11}};
```



Exercise

- Calculate address of the last element of the array.
- Input:
  - int arr[3][3][3][3][3][3][3][3],
  - address of the head element (e.g. 0x0204)

Exercise

- Input:
  - int arr[3][3][3][3][3][3][3][3],
  - address of the head element (e.g. 0x0204)

Answer: Considering a compiler which makes int 4-bytes, the last position is
= 0x6884
= $(204)_{16}$ + $((3^8 - 1) * 4)_{10}$

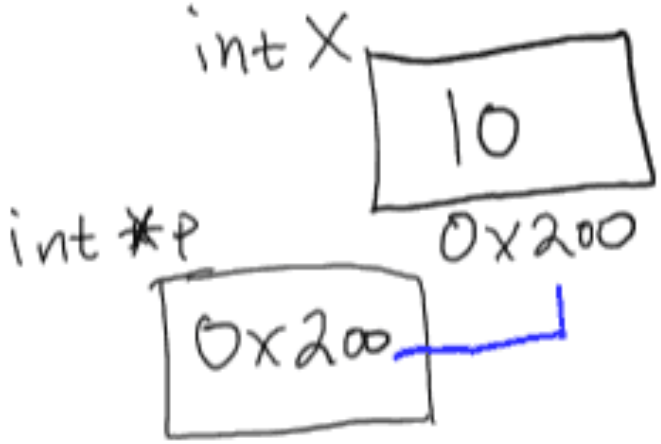
Pointers

Variables that store memory addresses instead of the actual data or values are called pointers.

```
int x, *p;

// storing an integer value
x = 10;

// storing an address of
// an integer variable
p = &x;
```

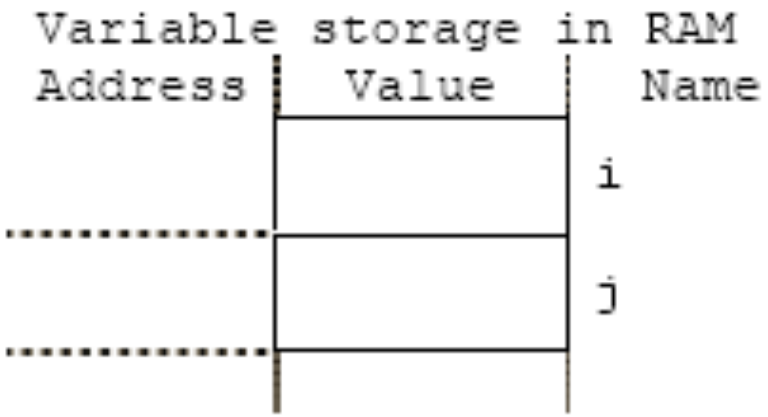


Pointers Practice

Remember that &i will give the address of variable i. Build and run the following program, show the output and answer to the question.

```
#include <stdio.h>

void main(void)
{
    int i = 7, j = 11;
    printf("i = %d, j = %d\n", i, j);
    printf("&i = %p, &j = %p\n", &i, &j);
}
```

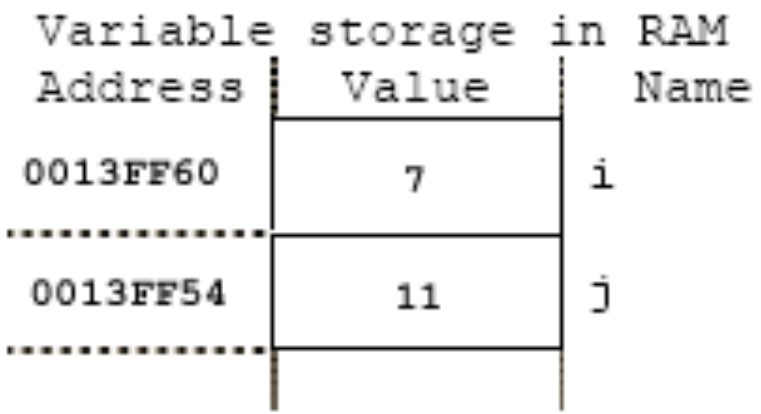


Pointers Practice

Remember that &i will give the address of variable i. Build and run the following program, show the output and answer to the question.

```
#include <stdio.h>

void main(void)
{
    int i = 7, j = 11;
    printf("i = %d, j = %d\n", i, j);
    printf("&i = %p, &j = %p\n", &i, &j);
}
```

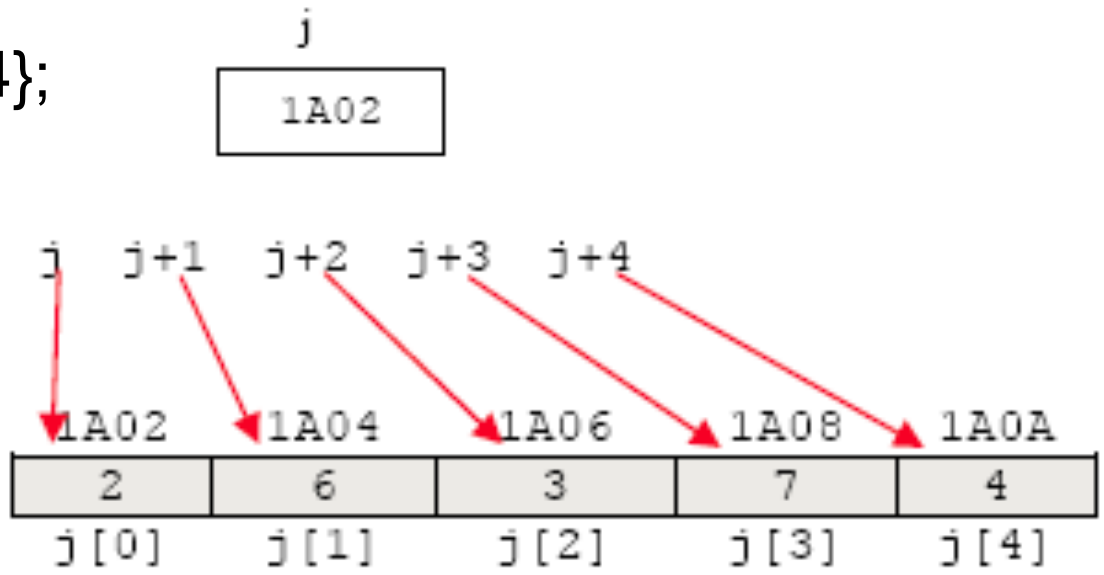


For 32 bits system the 0012FF60 hexadecimal will be converted to **32 bits binary**

(one hex character represented by 4 binary digits).

Pointer Arithmetic

```
int *p, j[ ] = {2, 6, 3, 7, 4};
```



Pointer variable can be changed.

However **pointer constant** cannot be changed.

Pointer Arithmetic Example

```
#include <stdio.h>

int main()
{
    int a[10] = {1 ,2 ,5 ,3 , 1, -3};
    double b[20] = {4, 3, 2.0, 2.5, -15.66};
    int i;

    for (i = 0; i < sizeof(a) / sizeof(int); i++)
    {
        printf("%d ", *(a+i));
    }
    for (i = 0; i < sizeof(b) / sizeof(double); i++)
    {
        printf("%f ", *(b+i));
    }
    return 0;
}
```

Code

Build and run the following program, show the output and answer the questions.

```
#include <stdio.h>

void main(void)
{
    int i = 7, j = 11;
    printf("i = %d, j = %d\n", i, j);
    printf("&i = %p, &j = %p\n", &i, &j);
    // reassign new values...
    &i = 4;
    j = 5;
    // reprint...
    printf("\ni = %d, j = %d\n", i, j);
    printf("&i = %p, &j = %p\n", &i, &j);
}
```

Questions

a) After the initialization statement, were we able to change the values of the variables?

b) After the assignment statements, were we able to change the addresses of the variables? Try &i = 4;

c) Now at the end of main(), add the following statement and try running it: `j = &i;` Were we able to store the address of i into j?

Answers

a Yes.

b No we can't. The &i = 4; generate the following error during the build time.

"...error C2106: '=' : left operand must be l-value..."

c We add the following code at the end of main().

```
// assigning the address of variable i to normal variable j
j = &i;
printf("\ni = %d, j = %d\n", i, j);
printf("&i = %p, &j = %p\n", &i, &j);
```

Yes, we can store the address of i into j.

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Exercise On Pointers 1

```
[1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	-2	h
0x29ff0c	int	1963357002	c
0x29ff10	int	1963357245	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff60	j
0x29ff1c	int *	0x40192e <__do_global_ctors+46>	e
0x29ff20	int *	0x4018d0 <__do_global_dtors>	d
0x29ff24	int	0	b
0x29ff28	int *	0x29ff0c	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	-2	h
0x29ff0c	int	1963357002	c
0x29ff10	int	5	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff60	j
0x29ff1c	int *	0x40192e <__do_global_ctors+46>	e
0x29ff20	int *	0x4018d0 <__do_global_dtors>	d
0x29ff24	int	0	b
0x29ff28	int *	0x29ff0c	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	-2	h
0x29ff0c	int	1963357002	c
0x29ff10	int	5	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff60	j
0x29ff1c	int *	0x40192e <__do_global_ctors+46>	e
0x29ff20	int *	0x4018d0 <__do_global_dtors>	d
0x29ff24	int	4	b
0x29ff28	int *	0x29ff0c	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
[4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	-2	h
0x29ff0c	int	1963357002	c
0x29ff10	int	5	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff60	j
0x29ff1c	int *	0x40192e <__do_global_ctors+46>	e
0x29ff20	int *	0x29ff0c	d
0x29ff24	int	4	b
0x29ff28	int *	0x29ff0c	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	-2	h
0x29ff0c	int	1963357002	c
0x29ff10	int	5	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff60	j
0x29ff1c	int *	0x29ff10	e
0x29ff20	int *	0x29ff0c	d
0x29ff24	int	4	b
0x29ff28	int *	0x29ff0c	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	-2	h
0x29ff0c	int	1963357002	c
0x29ff10	int	5	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff60	j
0x29ff1c	int *	0x29ff10	e
0x29ff20	int *	0x29ff0c	d
0x29ff24	int	4	b
0x29ff28	int *	0x7	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	-2	h
0x29ff0c	int	1963357008	c
0x29ff10	int	5	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff60	j
0x29ff1c	int *	0x29ff10	e
0x29ff20	int *	0x29ff0c	d
0x29ff24	int	4	b
0x29ff28	int *	0x7	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	-2	h
0x29ff0c	int	1963357008	c
0x29ff10	int	5	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff60	j
0x29ff1c	int *	0x29ff10	e
0x29ff20	int *	0x29ff0c	d
0x29ff24	int	4	b
0x29ff28	int *	0xb	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	-2	h
0x29ff0c	int	1963357008	c
0x29ff10	int	6	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff60	j
0x29ff1c	int *	0x29ff10	e
0x29ff20	int *	0x29ff0c	d
0x29ff24	int	4	b
0x29ff28	int *	0xb	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	-2	h
0x29ff0c	int	1963357008	c
0x29ff10	int	6	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff08	j
0x29ff1c	int *	0x29ff10	e
0x29ff20	int *	0x29ff0c	d
0x29ff24	int	4	b
0x29ff28	int *	0xb	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	10	h
0x29ff0c	int	1963357008	c
0x29ff10	int	6	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff08	j
0x29ff1c	int *	0x29ff10	e
0x29ff20	int *	0x29ff0c	d
0x29ff24	int	4	b
0x29ff28	int *	0xb	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	11	h
0x29ff0c	int	1963357008	c
0x29ff10	int	6	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff08	j
0x29ff1c	int *	0x29ff10	e
0x29ff20	int *	0x29ff0c	d
0x29ff24	int	4	b
0x29ff28	int *	0xb	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	11	h
0x29ff0c	int	1963357008	c
0x29ff10	int	6	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff08	j
0x29ff1c	int *	0x29ff10	e
0x29ff20	int *	0x29ff0c	d
0x29ff24	int	79	b
0x29ff28	int *	0xb	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;                    ///// Here we run a quiz
[15] *d = a + *j;
[16] *k = 13;
```

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	11	h
0x29ff0c	int	1963357008	c
0x29ff10	int	7	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff08	j
0x29ff1c	int *	0x29ff10	e
0x29ff20	int *	0x29ff0c	d
0x29ff24	int	79	b
0x29ff28	int *	0xb	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

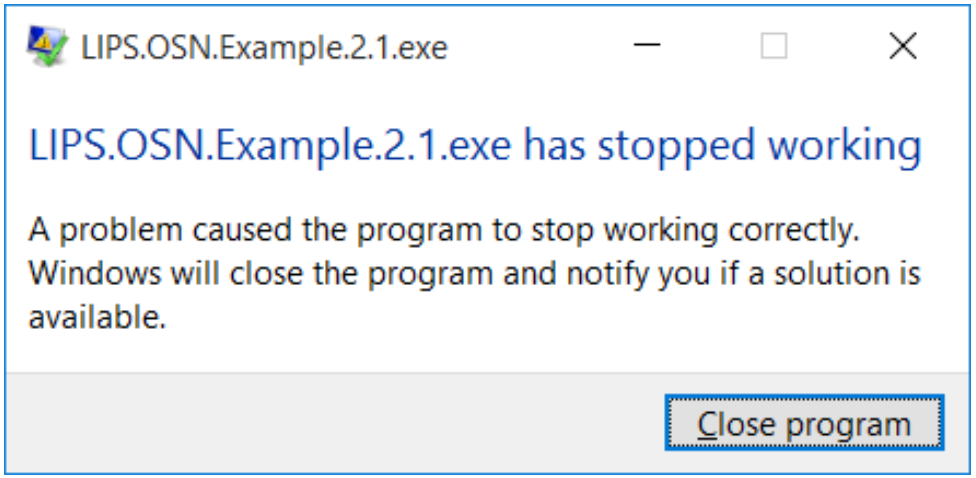
Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff08	int	11	h
0x29ff0c	int	18	c
0x29ff10	int	7	a
0x29ff14	int *	0x4018d0 <__do_global_dtors>	k
0x29ff18	int *	0x29ff08	j
0x29ff1c	int *	0x29ff10	e
0x29ff20	int *	0x29ff0c	d
0x29ff24	int	79	b
0x29ff28	int *	0xb	g
0x29ff2c	int *	0x29ff10	f
0x29ff38	start address		

Exercise On Pointers 1

```
 [1] int a, b, c, *d, *e, *f=&a, *g=&c;
 [2] a = 5;
 [3] b = 4;
 [4] d = &c;
 [5] e = &a;
 [6] g = 7;
 [7] *d = *d + *e + 1;
 [8] g++;
 [9] (*f)++;
[10] int  h, *k, *j=&h;
[11] h = 10;
[12] (*j)++;
[13] b = *f + g + h;
[14] (*f)++;
[15] *d = a + *j;
[16] *k = 13;
```



Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	2752352	f
0x29ff1c	int	4200798	a
0x29ff20	int *	0x401900 <__do_global_dtors>	d
0x29ff24	int *	0x0	b
0x29ff28	int	2752280	g
0x29ff2c	int	2752284	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	2752352	f
0x29ff1c	int	4200798	a
0x29ff20	int *	0x401900 <__do_global_dtors>	d
0x29ff24	int *	0x0	b
0x29ff28	int	2752280	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	2752352	f
0x29ff1c	int	4200798	a
0x29ff20	int *	0x401900 <__do_global_dtors>	d
0x29ff24	int *	0x4	b
0x29ff28	int	2752280	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	2752352	f
0x29ff1c	int	4200799	a
0x29ff20	int *	0x401900 <__do_global_dtors>	d
0x29ff24	int *	0x4	b
0x29ff28	int	2752280	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	16803200	f
0x29ff1c	int	4200799	a
0x29ff20	int *	0x401900 <__do_global_dtors>	d
0x29ff24	int *	0x4	b
0x29ff28	int	2752280	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	16803200	f
0x29ff1c	int	4200799	a
0x29ff20	int *	0x4b0d48	d
0x29ff24	int *	0x4	b
0x29ff28	int	2752280	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	16803200	f
0x29ff1c	int	4200799	a
0x29ff20	int *	0x4b0d48	d
0x29ff24	int *	0x4	b
0x29ff28	int	2752280	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	16803200	f
0x29ff1c	int	4200799	a
0x29ff20	int *	0x4b0d48	d
0x29ff24	int *	0x4	b
0x29ff28	int	6953079	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	16803200	f
0x29ff1c	int	4200799	a
0x29ff20	int *	0x4b0d48	d
0x29ff24	int *	0x29ff1c	b
0x29ff28	int	6953079	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	16803200	f
0x29ff1c	int	6953099	a
0x29ff20	int *	0x4b0d48	d
0x29ff24	int *	0x29ff1c	b
0x29ff28	int	6953079	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	16803201	f
0x29ff1c	int	6953099	a
0x29ff20	int *	0x4b0d48	d
0x29ff24	int *	0x29ff1c	b
0x29ff28	int	6953079	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	16803201	f
0x29ff1c	int	6953099	a
0x29ff20	int *	0x29ff18	d
0x29ff24	int *	0x29ff1c	b
0x29ff28	int	6953079	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	16803201	f
0x29ff1c	int	6953099	a
0x29ff20	int *	0x29ff18	d
0x29ff24	int *	0x29ff18	b
0x29ff28	int	6953079	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	23756300	f
0x29ff1c	int	6953099	a
0x29ff20	int *	0x29ff18	d
0x29ff24	int *	0x29ff18	b
0x29ff28	int	6953079	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	30709399	f
0x29ff1c	int	6953099	a
0x29ff20	int *	0x29ff18	d
0x29ff24	int *	0x29ff18	b
0x29ff28	int	6953079	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	68371907	f
0x29ff1c	int	6953099	a
0x29ff20	int *	0x29ff18	d
0x29ff24	int *	0x29ff18	b
0x29ff28	int	6953079	g
0x29ff2c	int	10	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	68371907	f
0x29ff1c	int	6953099	a
0x29ff20	int *	0x29ff18	d
0x29ff24	int *	0x29ff18	b
0x29ff28	int	6953079	g
0x29ff2c	int	11	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	68371907	f
0x29ff1c	int	6953099	a
0x29ff20	int *	0x29ff18	d
0x29ff24	int *	0x29ff1c	b
0x29ff28	int	6953079	g
0x29ff2c	int	11	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	68371907	f
0x29ff1c	int	6953099	a
0x29ff20	int *	0x29ff18	d
0x29ff24	int *	0x29ff1c	b
0x29ff28	int	6953080	g
0x29ff2c	int	11	c
0x29ff38	start address		

Exercise On Pointers 2

```
 [1] int a, *b, c=&a, *d, f, g=&f;
 [2] c = 10;
 [3] b = 4;
 [4] a = a + 1;
 [5] f = a + b;
 [6] d = malloc(sizeof(int));
 [7] *d = 20;
 [8] g = g + a;
 [9] b = &a;
[10] *b = *d + g;
[11] f++;
[12] d = &f;
[13] b = d;
[14] *b = a + f;
[15] *d = *d + a;
[16] f = a + *b + *d + c;
[17] c++;
[18] b++;
[19] g++;
[20] f = a + b + *d;
```

Address	Type	Value	Name
0x29ff00	end address		
0x29ff18	int	304052308	f
0x29ff1c	int	6953099	a
0x29ff20	int *	0x29ff18	d
0x29ff24	int *	0x29ff1c	b
0x29ff28	int	6953080	g
0x29ff2c	int	11	c
0x29ff38	start address		

Few typical errors

```
#include <stdio.h>

int main()
{
    int a = 1025;
    int *b = a;
    *b = 1000;
    /*
        Segmentation Fault

    */
    return 0;
}
```

Few typical errors

```
#include <stdio.h>

int main()
{
    int user_input = 0;
    scanf("%d", user_input);
    printf("%d", user_input);
    /*
        SegFault.
        Halt after scanf
    */
    return 0;
}
```

Problem 3

1. How to find the size of an array?
2. Why is it important to initialize an array beforehand?
3. Is it valid to refer to the i^{th} array element like this?

```
*(my_array + i)
```

Problem 3 - Solution (1/3)

1. To find the size of an array we use sizeof() function. However, it's a little bit tricky since sizeof(arr) will return the size of memory allocated for an array. So, we need to divide it by the size of an array type:

```
  int arr_size =
sizeof(arr) / sizeof(*arr);
```

Problem 3 - Solution (2/3)

2. When we declare an array, the memory is allocated, but is not cleared or filled by default values for us. Instead, it may contain some values remained after other program execution. To avoid unexpected results, we should use calloc() function (will be explained later) or run the following code:

```
for (i = 0; i < arr_size; i++) {
 arr[i] = …;
}
```

Problem 3 - Solution (3/3)

3. Yes, it's a valid way which is called pointer arithmetics. We can add and subtract integers to and from pointers. The value of the pointer is adjusted by the **amount multiplied by the size (in bytes) of the type** to which the pointer refers

Problem 4

- Write a program that will contain the definition for a structure representing complex numbers and will perform some simple operations on them

Problem 4 - Solution (1/5)

- Definition of a structure:

```
struct <struct-type>{
  <type> <identifier_list>;
  <type> <identifier_list>;
  …
} ;
```

Each identifier defines a *member* of the structure

- Example:

```
struct Date {
  int day;
  int month;
  int year;
};
```

The Date structure has 3 members: day, month & year.

Problem 4 - Solution (2/5)

- Declaration of a variable of struct type:

```
struct <struct-type> <identifier_list>;
```

- Example:

```
struct Date d1, d2;
```

- *d1* and *d2* are variables of *Date* type

Problem 4 - Solution (3/5)

- The structure definition:

```
typedef struct comp_num {
  float r;
  float i;
} comp_num;
```

Problem 4 - Solution (4/5)

- The addition function definition:

```
comp_num add(comp_num cn1,
comp_num cn2){
    comp_num result;
    result.r = cn1.r + cn2.r;
    result.i = cn1.i + cn2.i;
    return(result);
  }
```

Problem 4 - Solution (5/5)

- main function():

```
int main(void) {
    comp_num cn1, cn2;
    // struct comp_num cn1, cn2;
    scanf("%f%f",&cn1.r,&cn1.i);
    scanf("%f%f",&cn2.r,&cn2.i);
    comp_num result = add(cn1,
cn2);
    return 0;
}
```

Problem 5

1. Define a structure called *node*, which contains an integer element called *data*, and a pointer to a structure of type *node* called *next_node*
2. Declare three structures called *node1*, *node2*, *node3*, of type *node*
3. Write C statements which will link the three nodes together, with *node1* at the head of the list, *node2* second, and *node3* at the tail of the list. Assign the value NULL to *node3.next* to signify the end of the list
4. Using a pointer *list*, of type node, which has been initialised to the address of *node1*, write C statements which will cycle through the list and print out the value of each nodes data field
5. Assuming that pointer list points to *node2*, what does the following statement do?

```
list->next_node = (struct node *) NULL;
```

End

Week 03 – Tutorial

References

- http://www.gdsw.at/languages/c/programming-bbrown/c_088.htm
- https://en.wikibooks.org/wiki/C_Programming/Pointers_and_arrays#Pointers_and_Arrays
- http://www.programiz.com/c-programming/examples/complex-number-add
- http://www.tenouk.com/Module11.html