We are here to make our own journey; in this journey finding the courage to be ourselves is not easy, but without ourselves nothing can find the meaning in our lives.

~D. Cuceloglu
Memory Structure

- When a variable defined it is stored somewhere in memory.
- Memory can be thought as block consist of cells.
- When a variable defined, required number of cell from memory is allocated for the variable.
- How many cell will be reserved for the variable depends on the type of variable.
Memory Structure

```c
#include <stdio.h>

int main(void)
{
    // Degiskenler tanımlanıyor:
    int num1, num2;
    float num3, num4;
    char i1, i2;
    // Degiskenlere atama yapılıyor:
    num1 = 5;
    num2 = 12;
    num3 = 67.09;
    num4 = 1.71;
    i1 = 'H';
    i2 = 'p';
    return 0;
}
```
If we illustrate the structure of memory after the code in previous slide.
- Assume that size of int is 2 byte, size of float is 4 byte and size of char is byte.
- Each cell represents 1 byte space.
- Memory portion for defined variables starts from the address 4300.
Memory Structure

- When a variable is defined, a space required for the variable is reserved in the memory.
- E.g. definition `int num1` reserves 2 byte space for variable `num1`.
- After that if the value 5 is assigned on variable num1, 5 is stored in memory location allocated for that variable.
- Actually, all operations taken on variable num1 is the modification of cells in the memory location between 4300 and 4302.
- Variable is actually a memory location reserved for a particular label.
Defining Pointer

- Pointer is a data type that shows the memory address of a data block.
  ```
  data_type *p;
  ```
- Variable `p` stores the **address** of a variable which is in `<data_type>` type
  ```
  int *iptr;
  float *fptr;
  ```
- The only thing that we should pay attention is defining pointer suitable for the data type it points.
- A float variable must only be pointed by a float type pointer.
Defining Pointer

- To make a pointer show the address of a variable, address of the variable should be assigned to the pointer.
- For this purpose we should know the address of the memory location used for the variable.
- It is possible with address operator (&).
  - \&y → gives the address of variable y.

```c
int y = 5;
int *yPtr;
yPtr = &y;
```
Defining Pointer

• After assigning the address of a variable to a pointer, pointer starts to show the address of related variable.

• If we want to access or modify the value of a variable with pointer, we should use * character in the beginning of pointer name.

• All modifications done with * character in the beginning of pointer name effects the original variable.
Defining Pointer

```c
#include <stdio.h>

int main(void)
{
    int i;
    int *iptr;
    i = 5;
    iptr = &i;

    printf("i adresi %p\n", &i);
    printf("iptr degeri %p\n", iptr);

    printf("i degeri %d\n", i);
    printf("*iptr degeri %d\n", *iptr);

    getchar();
    return 0;
}
```
Defining Pointers
(Accessing Variables by Pointers)

- Using pointers, we can change the values of stored variables.
- For accessing the value of a variable with pointer, we should use *
  character in the beginning of pointer name.

```c
#include<stdio.h>
int main()
{
    int i;
    int *iptr;
    iptr = &i;
    *iptr = 8;
    printf("i değişkeninin değeri %d\n", i);
    printf("iptr adresinin içeriği %d\n", *iptr);
    getchar();
    return 0;
}
```
Defining Pointer (Associating Variables with Pointers)

```c
#include<stdio.h>
int main( void )
{
    // int tipinde değişken tanımlıyoruz:
    int xyz = 10, k;
    // int tipinde pointer tanımlıyoruz:
    int *p;

    // xyz değişkeninin adresini pointer'a atıyoruz.
    // Bir değişken adresini '&' işaretliyle alırız.
    p = &xyz;

    // k değişkenine xyz'nin değeri atanır. Pointer'lar değer tutmaz.
    // değer tutan değişkenleri işaret eder.
    // Başına '*' koyulduğunda, işaret ettiği değişkenin değerini gösterir.
    k = *p;

    return 0;
}
```
Defining Pointer

- You can change the variable that pointer shows constantly throughout the program.

```c
#include<stdio.h>

int main( void )
{
    int x, y, z;
    int *int_addr;
    x = 41;
    y = 12;
    // int_addr x degiskenini isaret ediyor.
    int_addr = &x;
    // int_addr' in isaret ettigi degiskenin sakladigi deger aliniyor. (yani x'in degeri)
    z = *int_addr;
    printf( "z: %d\n", z );
    // int_addr, artik y degiskenini isaret ediyor.
    int_addr = &y;
    // int_addr' in isaret ettigi degiskenin sakladigi deger aliniyor. (yani y'in degeri)
    z = *int_addr;
    printf( "z: %d\n", z );
    return 0;
}
```
Defining Pointer

- **Malloc** function is used to show a pointer to an empty block of data.
- Thus, space for data is allocated dynamically.
  - `malloc(n)` → Takes the n byte place from empty memory and returns the starting address.
  - `iptr = (int*) malloc(sizeof(int));`
  - `else iptr = (int*) malloc(4);`
Size of Pointer

- Pointers generally have a fixed size, for example on a 32-bit system they're usually 32-bit.

```c
#include<stdio.h>

int main()
{
    double i;
    double *iptr;

    iptr = &i;
    printf("i boyutu: %d\n", sizeof(i));
    printf("iptr boyutu: %d", sizeof(iptr));

    getchar();
    return 0;
}
```
Pointers that point other Pointers

- As seen that pointers store the memory addresses of variables.
- Pointer is also a variable and an other pointer that shows a pointer can be defined.
- If we define a pointer variable that shows a pointer; we use '***' in the beginning of pointer name.
- Number of * can change. If we define a pointer that points an other pointer that points an other pointer we have to use ‘***’.
Pointers that point other Pointers
We can use increment (++) or decrement (--), addition (+) or subtraction (-) operators with pointers. But this value must be integer.

When we increment the pointer by 1, pointer shows the next data block.

New pointer value depends on the data type that pointer shows.

```c
int i, *iPtr;
iPtr = &i; // Assume iPtr shows address 1000
iPtr += 2; // After this operation new value of iPtr is 1008
```

Because int type occupies 4 bytes of memory space.
#include<stdio.h>

int main( void )
{
    int i, *iPtr;
    double y, *yPtr;

    iPtr = &i;
    printf("iPtr gösterdiği adres: %d \n", iPtr);
    iPtr ++; //int tipi için bir sonraki adres bloğu 4 bayt fazlası.
    printf("iPtr gösterdiği adres: %d \n\n", iPtr);

    yPtr = &y;
    printf("yPtr gösterdiği adres: %d \n", yPtr);
    yPtr ++; //double tipi için bir sonraki adres bloğu 8 bayt fazlası.
    printf("yPtr gösterdiği adres: %d ", yPtr);

    getchar();
    return 0;
}
Pointer Arithmetic

- int i , *iPtr;
- iPtr = &i; // Assume iPtr shows address 1000
- (*iPtr) ++; // Causes to increment value stored in the address 1000.
- iPtr ++; // Causes iPtr to show address 1004 in memory
- (*iPtr) +=2; // Increase value by 2 stored in 1000
- (*iPtr) =7; // Assign 7 in address 1000.
- *(iPtr+2) = 5; // Assign 5 in address 1008.