Ders 27

2 farklı fonksiyon değeri iletimi var returm ve adres ile

```
int get_value(void)
{

}

x = get_value();


void get_value(int *ptr)
{

}

get_value(&x);
```

2örnek hesaplama 2 farklı yöntem

```c
#include <stdio.h>

double get_circle_area(double radius)
{
    return 3.1415926 * radius * radius;
}

int main()
{
    double r;

    printf("yaricap degerini girin: ");
    scanf("%lf", &r);
    double area = get_circle_area(r);

    printf("area = %f\n", area);

}
```

2.yöntem pointer

```c
void get_circle_area(double radius, double *p_area)
{
    *p_area = 3.1415926 * radius * radius;
}

int main()
{
    double r;

    printf("yaricap degerini girin: ");
    scanf("%lf", &r);
    double area;

    get_circle_area(r, &area);

    printf("area = %f\n", area);

}
```

Bu kodu bu şekilde yazmam çünkü bir avantajı yok burada

Örnek birden fazla geri dönüş varsa

```c
#include <stdio.h>

void get_values(double x, double* p1, double* p2, double* p3);


int main()
{
    double d1, d2, d3;

    get_values(3.844536, &d1, &d2, &d3);

}
```

Örnek maliyet

```c
typedef struct {
    int a[20][20];
    int row, col;
}Matrix;

void get_solution_matrix(Matrix *p);

int main()
{
    Matrix x;
    get_solution_matrix(&x);


}
```

Örnek

```c
#include <stdio.h>

typedef struct {
    int a[20][20];
    int row, col;
}Matrix;

                //out parameter  in(put) param
void add_matrix(Matrix *presult, const Matrix *m1, const Matrix *m2);


int main()
{
    Matrix mx, my;
    Matrix result;

    add_matrix(&result, &mx, &my)


}
```

Burada amaç 2. Ve 3. Parametrede kopyalama maliyetini düşürmek

Örnek pointer +1

```c
int main()
{
    int a[10] = { 0, 1, 2, 3, 4, 5, 6, 7,

    for (int i = 0; i < 10; ++i) {
        printf("%p %p\n", &a[i], &a[0] + i
    }
```

```
Microsoft Visual Studio Debug Console
0133F998 0133F998
0133F99C 0133F99C
0133F9A0 0133F9A0
0133F9A4 0133F9A4
0133F9A8 0133F9A8
0133F9AC 0133F9AC
0133F9B0 0133F9B0
0133F9B4 0133F9B4
0133F9B8 0133F9B8
0133F9BC 0133F9BC
```

Örnek pointer ile dizinin elemanlarını yazmak

```c
int main()
{
    int a[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

    for (int i = 0; i < 10; ++i) {
        printf("%d  %d\n", a[i], *(a + i))
    }
```

İki satıda aynı çıkacak.

Eşitlikler

```
&a[i]               &a[0] + i     a + i     i + a

a[i]                *(&a[0] + i)     *(a + i)     *(i + a)

a[i]                *(a + i)


a[i]            i[a]
*(a + i)        *(i + a)
```

```c
int main()
{
    int a[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    //int* ptr = a;
    //int* ptr = &a[0];

    //int* ptr = &a[5];
    int* ptr = a + 5;

    *ptr
```

Örnek önemli ++ptr

```c
int main()
{
    int a[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    int* ptr = a;

    for (int i = 0; i < 10; ++i) {
        printf("%d  %d   %d  %d\n", a[i], *(a + i), *(i + a), *ptr);
        ++ptr;
    }
```

Eleman türüne bağlı olarak int short önemli değil pointer aritmetiği bir sonraki nesneyi gösterir.

+5 yapsaydık 5 sonraki nesneyi gösterirdi

MÜLAKAT SORUSU ÖNEMLİ !!!!!!!!!!!!!!!!!!!!!!!!!!

```c
int main()
{
    int a[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    int* ptr = a + 5;

    printf("%d\n", *ptr);
    printf("%d\n", ptr[0]);
    printf("%d\n", ptr[3]);
    printf("%d\n", ptr[-3]);
```

```
Microsoft Visual Studio Debug Console
5
5
8
2

D:\KURSLAR\NISAN_2022_C\Debug\NISAN_2022_C.exe (process 20536) exited with code 0.
Press any key to close this window . . .
```

Ekrana hangi yazılar yazar

A 0. İndisi gösteriyor a+5 demek 5 indisli 6. Elemanın adresini tutuyor demek.

Ptr[0] adrese 0 ekledik aynı kaldı değişmedi.

Ptr[3] ekledik yani *(ptr+3) yaptık 8. İndisi tuttu. Bir değeri adres olduğu için syntax hatası vermedi

Ptr[-3] ekledik yani *(ptr-3) yaptık bu seferde 3 çıkarınca 3 eleman geriye gitti ve 2 yazdı.

Örnek dizi yazdırmak

sorula

```
void array_func(int* ptr, int size)
{
    for (int i = 0; i < size; ++i) {
        ptr[i]
    }
}

void array_func2(int* ptr, int size)
{
    while (size--) {
        //*ptr
        //++ptr;

    }
}
```

Sorular

```c
#include <stdio.h>
#include "nutility.h"

#define         SIZE        20

int main()
{
    int a[SIZE];

    set_array_random(a, SIZE);
    print_array(a, SIZE);
    print_array(a, 4);
    print_array(a + 4, 3);
    print_array(a + SIZE - 3, 3);

}
```

Print ifadelerinde ne yazar

1 ve 2 bütün diziye işlem yaptır.

3. print sadece ilk 4 değişken ekrana yazar.

4.printte 4. İndexten itibaren 3 eleman yazılır.

5 de ise son 3 elemanı yazdırdık.

2 eleman adresi toplanamazzzzzz

Soru 1 dizinin aritmetik hesabbı.

toplamı

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <math.h>
#include "nutility.h"

#define         SIZE        100

//int sum_array(const int* p, int size)
//{
//  int sum = 0;
//  for (int i = 0; i < size; ++i) {
//      sum += p[i];
//  }
//
//  return sum;
//}

int sum_array(const int* p, int size)
{
    int sum = 0;

    while (size--) {
        sum += *p;
        ++p;
    }

    return sum;
}
```

Toplamının fonksiyonla aritmetik ortalaması ve standart sapması.

```
double get_mean(const int* p, int size)
{
    return (double)sum_array(p, size) / size;
}

double get_std_dev(const int* p, int size)
{
    double mean = get_mean(p, size);
    double sum_square = 0.;

    for (int i = 0; i < size; ++i) {
        sum_square += (p[i] - mean) * (p[i] - mean);
    }

    return sqrt(sum_square / (size - 1));
}


int main()
{
    int a[SIZE];

    randomize();

    set_array_random(a, SIZE);
    print_array(a, SIZE);

    printf("dizi toplami    = %d\n", sum_array(a, SIZE));
    printf("dizi ortalamasi = %f\n", get_mean(a, SIZE));
    printf("std sapma = %f\n", get_std_dev(a, SIZE));

}
```

Aynı koda ait hepsi incele

Soru 2 büyük ve küçük elemanı bulan fonksiyon

```c
#define         SIZE        20


//öyle bir fonksiyon yazın ki
//bir dizinin hem en büyük hem de en küçük elemanlarının değerlerini hesaplasın

void get_minmax(const int *p, int size, int *pmin, int *pmax)
{
    //*pmin = *pmax = p[0];
    *pmin = *pmax = *p;

    for (int i = 1; i < size; ++i) {
        if (p[i] < *pmin) {
            *pmin = p[i];
        }
        else if (p[i] > *pmax)
            *pmax = p[i];
    }

}

int main()
{
    int a[SIZE];

    randomize();

    set_array_random(a, SIZE);
    print_array(a, SIZE);

    int min, max;

    get_minmax(a, SIZE, &min, &max);

    printf("min = %d\n", min);
    printf("max = %d\n", max);

}
```

Burada 2 farklı değeri bulmak istediğimiz için return yerine istediğimiz değişkenlerin adreslerini değişkene argüman olarak bildirdik.

Soru 3 diziyi ters çeviren fonksiyon

```
void reverse_array(int* p, int size)
{
    for (int i = 0; i < size / 2; ++i) {
        int temp = p[i];
        p[i] = p[size - 1 - i];
        p[size - 1 - i] = temp;
    }
}

int main()
{
    int a[SIZE];

    randomize();

    set_array_random(a, SIZE);
    print_array(a, SIZE);
    reverse_array(a, SIZE);
    print_array(a, SIZE);
}
```

Bunu hazır swap ile yapabilirim.

```
void reverse_array(int* p, int size)
{
    for (int i = 0; i < size / 2; ++i) {
        //swap(&p[i], &p[size - 1 - i]);
        swap(p + i, p + size - 1 - i);
    }
}

int main()
```

Burada daha sade bir şekilde yazdık.

2 swap() adres vermeside olur ancak alttaki p+i daha çok tercih edilir.

Pointer aritmetiğini göstermek amaçlı bir ters çevirme örneği amaç öğretmek üretimde yazmazdım.

```
void reverse_array(int* p, int size)
{
    int n = size / 2;
    int* pe = p + size;

    while (n--) {
        swap(p++, --pe);
    }
}
```

--pe dizinin son elemanın değerini aldı pe—yapsaydık hata olurdu.

Daha sonra daha şık bir şekilde yazıcaz.

Soru 4 bir dizinin elemanlarını başka bir diziye kopyalayan kod.

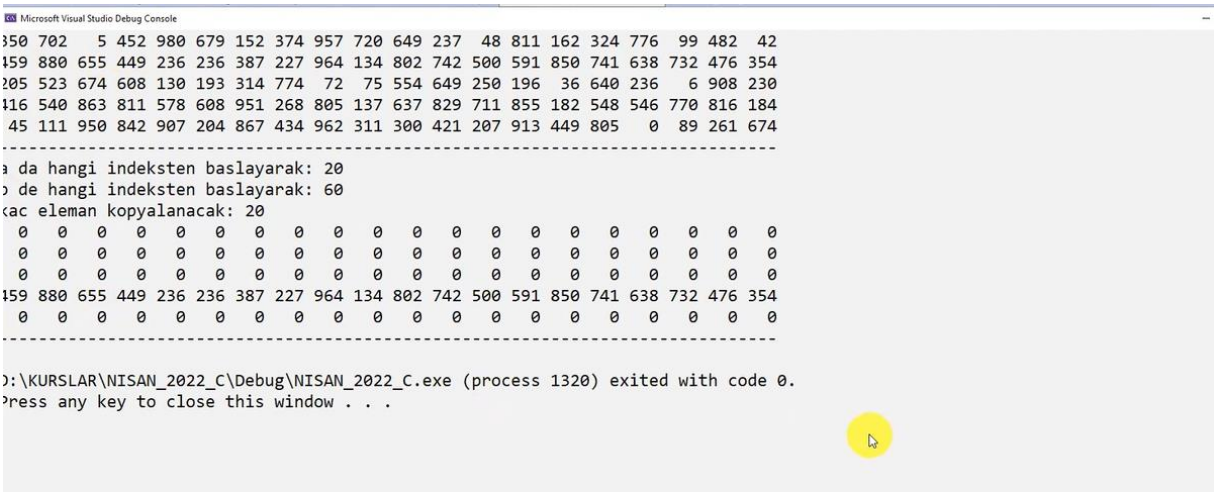
```
void copy_array(int* pdest, const int* psource, int size)
{
    while (size--) {
        *pdest++ = *psource++;
    }
}

//void copy_array(int* pdest, const int* psource, int size)
//{
//  while (size--) {
//      *pdest = *psource;
//      ++pdest;
//      ++psource;
//  }
//}

//void copy_array(int* pdest, const int* psource, int size)
//{
//  for (int i = 0; i < size; ++i) {
//      pdest[i] = psource[i];
//  }
//}

int main()
{
    int a[SIZE];
    int b[SIZE];

    randomize();

    set_array_random(a, SIZE);
    print_array(a, SIZE);

    copy_array(b, a, SIZE);

    print_array(b, SIZE);

}
```

3 farklı yazımda aynı işlemi yapmakta ancak üsteki idiomatik olarak çözüm.

Peki a dan istedğimiz indexten bde istediğimiz index değerinde başlayarak ve belli sayıda bir kopyalamayı nasıl yapardık cevabı aşağıda.

```c
int a[SIZE];
int b[SIZE] = { 0 };

randomize();

set_array_random(a, SIZE);
print_array(a, SIZE);

int idx_a;
int idx_b;
int n;
printf("a da hangi indeksten baslayarak: ");
scanf("%d", &idx_a);
printf("b de hangi indeksten baslayarak: ");
scanf("%d", &idx_b);
printf("kac eleman kopyalanacak: ");
scanf("%d", &n);


copy_array(b + idx_b, a + idx_a, n);
print_array(b, SIZE);
```

Fonksiyon çağrı ifadesini değiştirerek anın belirli adresteki indexinden bnin belirli adresindeki indexine istediğimiz sayıda kopyalama yapabiliriz.



Soru 5 bouble sort algoritması

```
void bsort(int* p, int size)
{
    for (int i = 0; i < size - 1; ++i) {
        for (int k = 0; k < size - 1 - i; ++k) {
            if (p[k] > p[k + 1]) {
                swap(p + k, p + k + 1);
                //swap(&p[k], &p[k + 1]);
            }
        }
    }
}
```

İç içe 2 döngü yazdık ve karşılaştırma yaptık daha sonra swap fonksiyonunu adresler ile çağırıp fonksiyonumuzu tamamladık.