

Ders 38

```
int main(void)
{
    int (*fpa[])(int) = { &isupper, &islower, &isdigit, &isalnum };
}
```

4 elemanlı fonksiyon pointer dizisi

Typedef ile

```
#define SIZE 100

typedef int (*FPTTEST)(int);

int main(void)
{
    //int (*fpa[])(int) = { &isupper, &islower, &isdigit, &isalnum };
    FPTTEST a[] = {&isupper, &islower, &isdigit, &isalnum};
}
```

```
typedef int (*FPTTEST)(int);

int main(void)
{
    const FPTTEST a[] = {&isupper, &islower, &isdigit, &isalnum};
}
```

Const ile yapmakta iyi

////////////////////////////////////

Basit örnek

```

void f1(void)
{
    printf("f1 cagrildi\n");
}

void f2(void)
{
    printf("f2 cagrildi\n");
}

void f3(void)
{
    printf("f3 cagrildi\n");
}

void f4(void)
{
    printf("f4 cagrildi\n");
}

int main(void)
{
    void (*fpa[])(void) = { &f1, &f2, &f3, &f4 };

    for (size_t i = 0; i < asize(fpa); ++i) {
        fpa[i]();
    }
}

```

Fpa[i]() iki opertörde aynı öncelik seviyesi ve soldan sağa önce fpa daki afrese eriştik sonra fonksiyonadresini () ile çağırdık.

////////////////////////////////////

Mülakat sorusu karakter test sorusu

```

typedef int (*FCTEST)(int);

int main(void)
{
    const FCTEST fa[] = { isupper, islower, isalpha, isdigit,
        isalnum, isxdigit, isspace, ispunct, isprint, isgraph, isblank, iscntrl };

    int ch;

    printf("bir karakter giriniz: ");
    ch = getchar();

    for (size_t i = 0; i < asize(fa); ++i) {
        if (fa[i](ch)) {
            printf("OK!\n");
        }
        else {
            printf("NOT OK!\n");
        }
    }
}

```

Seç Microsoft Visual Studio Debug Console

```

bir karakter giriniz: s
NOT OK!
i OK!
OK!
NOT OK!
si OK!
NOT OK!
NOT OK!
NOT OK!
OK!
OK!
NOT OK!
" NOT OK!

```

İşlerimi hangi fonksiyonun yazdığını yazalım

```

typedef int (*FCTEST)(int);

int main(void)
{
    const FCTEST fa[] = { isupper, islower, isalpha, isdigit, isalnum, isxdigit, isspace, ispunct, isprint, isgraph, isblank,
        const char * const fnames[] = {"isupper", "islower", "isalpha", "isdigit", "isalnum", "isxdigit", "isspace", "ispunct", "
    int ch;

    printf("bir karakter giriniz: ");
    ch = getchar();

    for (size_t i = 0; i < asize(fa); ++i) {
        if (fa[i](ch)) {
            printf("%s testi için OK!\n", fnames[i]);
        }
        else {
            printf("%s testi için NOT OK!\n", fnames[i]);
        }
    }
}

```



```

int main(void)
{
    const FCTEST fa[] = { isupper, islower, isalpha, isdigit, isalnum, isxdigit, isspace, ispunct, isprint, isgr
    const char * const fnames[] = {"isupper", "islower", "isalpha", "isdigit", "isalnum", "isxdigit", "isspace",
    char name_entry[SIZE];
    int ch;

    printf("bir karakter giriniz: ");
    ch = getchar();
    printf("hangi karakter testi yapılacak : ");
    scanf("%s", name_entry);

    size_t i;

    for (i = 0; i < asize(fnames) && strcmp(fnames[i], name_entry); ++i)
        ; // null statement

    if (i == asize(fnames)) {
        printf("ne yazik ki istediginiz testi yapamiyoruz\n");
    }
    else if (fa[i](ch)) {
        printf("%s testi icin %c karakteri OK!\n", fnames[i], ch);
    }
    else {
        printf("%s testi icin %c karakteri NOT OK!\n", fnames[i], ch);
    }
}

```

```

Microsoft Visual Studio Debug Console
: bir karakter giriniz: !
: hangi karakter testi yapılacak : ispunct
: ispunct testi icin ! karakteri OK!
: D:\KURSLAR\NISAN_2022_C\Release\NISAN_2022_C.exe (process 31024) exited with code 0.
: Press any key to close this window . . .

```

Sonuç olarak dışardan girilen bir isimle fonksiyonu çağırmaı başardık.

////////////////////////////////////

Fonksiyonun geri dönüş değeri fonksiyon pointer olan fonksiyonlar

Örnek

```

// int (*)(int)

int(*foo(void))(int)
{
    //code...
    return &isupper;
}

```

Örnek

Parametresiz bar strcmp adresi

```
int (* bar(void))( const char* , const char *)
```

Return &strcmp;

Typedef ile yazalım.

```
//parametresiz olmayan geri dönüş değeri std. strcmp
// işlevinin adresi olan bar isimli fonksiyonu tanımlayınız
typedef int (*FCOMP)(const char*, const char*);

FCOMP bar(void)
{
    return &strcmp;
}
```

Aynı işlevi typedef ile yapınca çok kolay oldu.

////////////////////////////////////

Örnek parametresi ve geri dönüş değeri karakter test fonksiyonu.

```

//int(*func(int (*fp)(int)))(int)
//{
//  //code
//  return fp;
//}

typedef int (*FPTEST)(int);

FPTEST func(FPTEST f)
{
    //code
    return f;
}

```

2 yazımda doğru ama type def daha kolay anlaşılıyor.

Örnek set_func

```

int main(void)
{
    func(); //burada func foo'yu çağırarak
    I fp = set_func(bar); //bu noktadan sonra eger func cagrilirsa func artık foo'yu değil bar'i cagiracak
    //şimdi fp göstericisi eski default fonksiyon olan foo'nun adresini tutuyor
    func(); //burada func bar'i çağırarak
    set_func(fp); //bu noktadan sonra eger func cagrilirsa func artık bar'i' değil foo'i
}

```



.h başlığı

```
#pragma once

typedef void (*FPTR)(void);

void func(void);
FPTR set_func(FPTR);
```

İstediğim bildirimleri burda yaptım.

Eray.c

```
#include <stdio.h>

static void foo(void)
{
    printf("foo cagrildi!!!\n");
}

static FPTR gfp = &foo;

void func(void)
{
    gfp();
}

FPTR set_func(FPTR f)
{
    FPTR ftemp = gfp;
    gfp = f;
    return ftemp;
}
```

Static global yapma nedenimiz bu bildirimleri eray.c dışında başka bir yerin kullanmasını engellemek.

Func fonksiyonu ile global bir *fp değerini çağırıyor.

Sey func ise geri dönüş değeri eskiden tutulan *fp nin değeri ancak içerde farklı bir değer ile global *gfp ye değer atıyor.

Sonra func çağırılınca yeni fonksiyon dönüyor tekrar eskisini çağırarak için set func ın döndürdüğü değeri tekrar fonksiyona set func a argüman olarak veriyoruz.

Main.c

```
#include "eray.h"
#include <stdio.h>

void bar(void)
{
    printf("bar cagrildi!!!\n");
}

int main(void)
{
    func(); //burada func foo'yu çağırarak
    FPTR fp = set_func(bar); //bu noktadan sonra eger func cagrilirsa func artık foo'yu değil bar'
    //şimdi fp göstericisi eski default fonksiyon olan foo'nun adresini tutuyor
    func(); //burada func bar'i çağırarak
    set_func(fp); //bu noktadan sonra eger func cagrilirsa func artık bar'i' değil foo'i
}
```

Bu kodun anlatımını bi daha dinle 1.00.00 saat

////////////////////////////////////

Örnek c++

Fonksiyonu değiştirip tekrar aynı fonksiyona çevirmek

```
int main(void)
{
    terminate_handler fpx = set_terminate(&my_abort);

    ///bu noktada

    set_terminate(fpx),
}
```

////////////////////////////////////

Örnek exit atexit

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

void f1(void){ printf("necati'nin f1 fonksiyonu\n");}
void f2(void){ printf("necati'nin f2 fonksiyonu\n");}
void f3(void){ printf("necati'nin f3 fonksiyonu\n");}
void f4(void){ printf("necati'nin f4 fonksiyonu\n");}

int main(void)
{
    printf("main basladi\n");
    exit(1);
    printf("main devam ediyor\n");
}
```

```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) NISAN_2022_C
Microsoft Visual Studio Debug Console
main basladi
D:\KURSLAR\NISAN_2022_C\Release\NISAN_2022_C.exe (process 26908) exited with code
Press any key to close this window . . .
```

Exit programı sonlandırır.

Ancak daha öncesinde atexit diye bir fonksiyon çağırırsak exit e bu fonksiyonları kaydeder. bu exit programı sonlandıracağı zaman kaydedilen fonksiyoları sondan basa doğru yazdırır.

```

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

void f1(void){ printf("necati'nin f1 fonksiyonu\n");}
void f2(void){ printf("necati'nin f2 fonksiyonu\n");}
void f3(void){ printf("necati'nin f3 fonksiyonu\n");}
void f4(void){ printf("necati'nin f4 fonksiyonu\n");}

int main(void)
{
    printf("main basladi\n");
    atexit(f1);
    atexit(f2);
    atexit(f3);
    atexit(f4);

    exit(1);
    printf("main devam ediyor\n");
}

```

```
#include <stdlib.h>
```

Microsoft Visual Studio Debug Console

```

main basladi
necati'nin f4 fonksiyonu
necati'nin f3 fonksiyonu
necati'nin f2 fonksiyonu
necati'nin f1 fonksiyonu

```

```

D:\KURSLAR\NISAN_2022_C\Release\NISAN_2022_C.exe (process 30224) exited with code 1.
Press any key to close this window . . .

```

Burada kaydedilenler ekrana yazdı.

////////////////////////////////////

Örnek exit benzerini kendimiz yazalım.

```
void f2(void){ printf("necati'nin f2 fonksiyo  
void f3(void){ printf("necati'nin f3 fonksiyo  
void f4(void){ printf("necati'nin f4 fonksiyo
```

```
int main(void)  
{  
    f_register(f1);  
    f_register(f2);  
    f_register(f3);  
    f_register(f4);  
  
    func();  
}
```

Func çağırıldığı zaman baştan sona hepsi çağırılsın .

Eray.h

```
#pragma once  
  
typedef void (*FPTR)(void);  
  
void func(void);  
void f_register(FPTR);
```

Eray.c

```
#define          MAX_NO_OF_FUNCS      20
```

```
static FPTR gfa[MAX_NO_OF_FUNCS];  
static int g_idx = 0;
```

```
void f_register(FPTR f)  
{  
    gfa[g_idx++] = f;  
}
```

```
void func(void)  
{  
    for (int i = 0; i < g_idx; ++i)  
        gfa[i]();  
}
```

Main.c

```
void f2(void){ printf("necati'nin f2 fonksiyonu\n");}  
void f3(void){ printf("necati'nin f3 fonksiyonu\n");}  
void f4(void){ printf("necati'nin f4 fonksiyonu\n");}
```

```
int main(void)  
{  
    f_register(f1);  
    f_register(f2);  
    f_register(f3);  
    f_register(f4);  
  
    func();  
}
```


Örnek

```
int main(void)
{
    //b öyle bir dizi ki boyutu 10 ve elemanları 8 elemanlı double türden diziler
    double b[10][8];
}
```

Bu hatalımı

```
int main(void)
{
    int a[10][20];

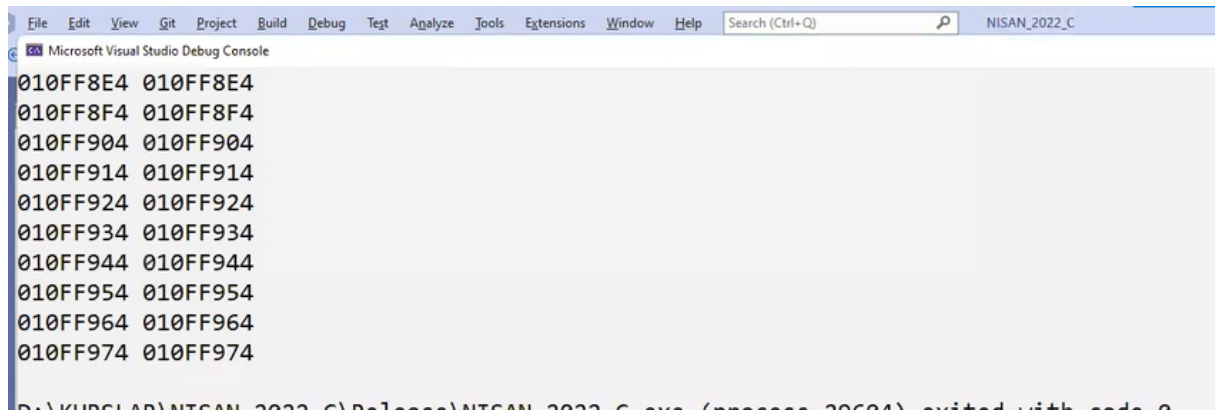
    int* p;
}
```

Evet bu yazım hatalı

Örnek elemanları her biri artınca ekrana neyazar.

```
int main(void)
{
    int a[10][4];

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


```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) NISAN_2022_C
Microsoft Visual Studio Debug Console
010FF8E4 010FF8E4
010FF8F4 010FF8F4
010FF904 010FF904
010FF914 010FF914
010FF924 010FF924
010FF934 010FF934
010FF944 010FF944
010FF954 010FF954
010FF964 010FF964
010FF974 010FF974
D:\KURSLAR\NISAN_2022_C\Debug\NISAN_2022_C.exe (process 30604) exited with code 0
```

4 adet 4 byte hepsi ekrenda 16 16 artarak gider.

Bu örnekte 64 elemanlı diziler var

```
int main(void)
{
    int a[10][64];

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



```
Microsoft Visual Studio Debug Console
00DAF22C 00DAF22C
00DAF32C 00DAF32C
00DAF42C 00DAF42C
00DAF52C 00DAF52C
00DAF62C 00DAF62C
00DAF72C 00DAF72C
00DAF82C 00DAF82C
00DAF92C 00DAF92C
00DAFA2C 00DAFA2C
00DAFB2C 00DAFB2C
```

Bu örnekte 64 *4 256 yani her seferinde 16 lık sayı sisteminde sağdan 3. Digit artıyor.

Örnek double ve 32 eleman olsaydı

```
int main(void)
{
    double a[10][32];

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

Yine 3. Bit artacaktı çünkü 32×8 yine 256 byte yapıyor.

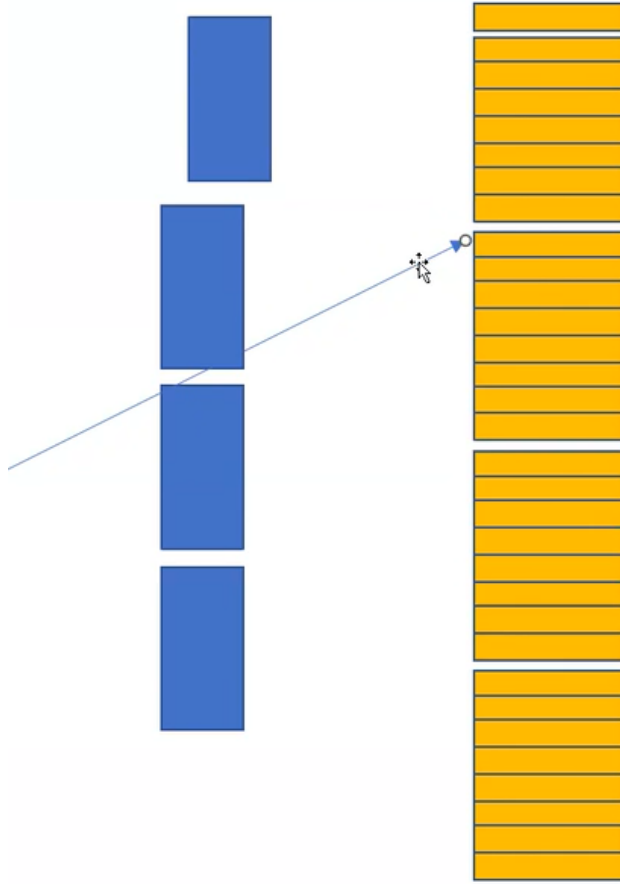
```
int main(void)
{
    int a[10][20];

    int* p1 = &a[0][0];
    int* p2 = a[0]; //decay
    int* p3 = (int*)a;
    int* p4 = &**a;
```

Örnek `int *p` türünden bir değişken tanımlasak ve bunun elemanlarını 1 arttırsak ne olur.

Aşağıdaki gibi şimdi türü `int *` ve diziler sıralı olduğu için ilk eleman ilk adres 2. Eleman 2. Adres diye gider.

Peki ilk dizinin son elemanından sonra 1 arttırsak 2. Dizinin ilk elemanını tutar mı



Evet tutar çünkü ardışık sıralı duruyorlar.

Ama unutma bu şekilde kullanmak istersem `a[10][10]` diye dizi tanımsarsam son elemanı `a+100` olur

Bu yüzden `int (*p)[10]` diye tanımlayada biliriz ama `int *p` de olur.

////////////////////////////////////

Örnek ilk değer verme

```

int main(void)
{
    //int[3]
    int a[5][3] = { {1, 1}, {2, 2, 2}, {3}, {4, 4, 4}, {5, 5} };

    for (int i = 0; i < 5; ++i) {
        for (int k = 0; k < 3; ++k) {
            printf("%d", a[i][k]);
        }
        printf("\n");
    }
}

```

İlk değer vermediklerimiz 0 olacak .

```

int main(void)
{
    //int[3]
    int a[5][3] = { {1, 1,}, {2, 2, 2,}, {3,}, {4, 4, 4,}, {5, 5, }, };

    for (int i = 0; i < 5; ++i) {
        for (int k = 0; k < 3; ++k) {
            printf("%d", a[i][k]);
        }
        printf("\n");
    }
}

```

Tarling comma yani fazla virgül olup yanında sayı olmasaydı da sentax hatası olmazdı.

////////////////////////////////////

Örnek küme parantezi olmadan ilk değer vermek

```

int main(void)
{
    //int[3]
    int a[5][3] = {1, 2, 3, 4, 5, 6, 7, 8};

    for (int i = 0; i < 5; ++i) {
        for (int k = 0; k < 3; ++k) {
            printf("%d", a[i][k]);
        }
        printf("\n");
    }
}

```

Derleyici sırayla ilk değer vericekti dizilere.

Bitti yerden itibaren kalanlar 0 olacaktı. Küme parantezi zorunlu değil.

////////////////////////////////////

Tpik mülakat sorusu

////////////////////////////////////

Hangileri sentax hatasıdır.

```
int main(void)
{
    //int a1[][] = {1, 2, 3, 4, 5, 6, 7, 8};
    //int a2[][3] = {1, 2, 3, 4, 5, 6, 7, 8};
    //int a3[5][ ] = {1, 2, 3, 4, 5, 6, 7, 8};
}
```

Dizinin boyutu ilk[] köşeli parantezdir bunu yazmak zorunda değilim ilk değer verirken ancak 2.[] köşeli parantez türün bir parçasıdır derleyici türünü bilemediği için sentax hatası verir.

Birincide hata var

2.cide 3 eleman 3 eleman ayırıcay boyutu 3 olucak hata yok

3.de son [] boş hatalı.

////////////////////////////////////

Örnek

Asize makrosu kaçıtır.

```

int main(void)
{
    int a[3][3] = {1, 2, 3, 4, 5, 6, 7, 8};

    printf("%zu\n", asize(a));
}

```

Kendisi / ilk eleman boyutu olduğu için 3 cevabı çıkar

```

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

    printf(_Format: "%zu\n", asize(a));
}

```

Bu örnekte boyut belirtilmemiş

cevap 4 olur çünkü 10 eleman var derleyici fazla eleman için derleyici boyutu 4 dür diye düşünür.

////////////////////////////////////

Soru deziknetion ini azilion köşeli parantezle ilk değer vermek.

```

int main(void)
{
    int a[10][5] = { [7] = {2, 8, 9, 4}, [5] = { [3] = 3 } };
}

```

Burada 7 indisli eleman dizi diziye ilk değer verdik.

Sonra 5 indisli elemana da dizi olduğu için onunda 3 indisli elemanına 3 değerini verdik hata yok.

Boyutu silsem geçerlimi

```
int main(void)
{
    int a[][5] = { [7] = {2, 8, 9, 4}, [5] = { [3] = 3 } };
}
```

Bence geçerli en büyük elemanı 7. İndis olur boyutu 8 olur.

////////////////////////////////////

```
int a[5][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9},
               {10, 11, 12}, {13, 14, 15}
};

for (int i = 0; i < 5; ++i) {
    for (int k = 0; k < 3; ++k) {
        printf("%d ", a[i][k]);
    }
    printf("\n");
}
```

Bütün elemanları ekrana yazdırdık.

```

        printf("%d ", a[i][k]);
    }
    printf("\n");
}

printf("\n\n");

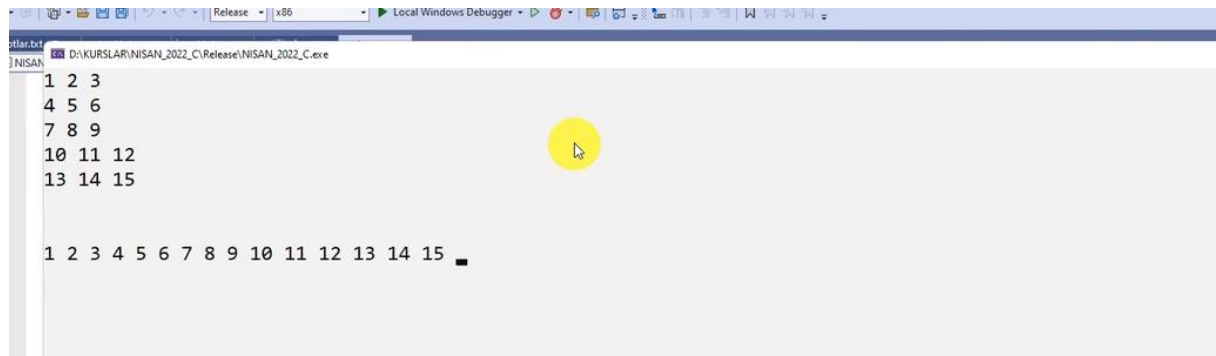
int* p = &a[0][0];
//int* p = a[0];
//int* p = (int*)a;
//int* p = &**a;

int n = 15;
while (n--) {
    printf("%d ", *p++);
    _getch();
}

```

Buradada elemanları yazdırdık.

İlk elemanı pointer ile atadık ve dizi bellekte sıralı adreslerde olduğu için tek bir diz imiş gibi sıra ile yazdırdık.



```

1 2 3
4 5 6
7 8 9
10 11 12
13 14 15

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

```