Ders 35

Örnek dizide max min adreslerini bulmak

```
void get_array_min_max(const int* pa, size_t size, int** ptr_min, int** ptr_max)
{
    *ptr_min = *ptr_max = pa[0];

    for (size_t i = 1; i < size; ++i) {
        if (pa[i] > **ptr_max)
            *ptr_max = (int*)(pa + i);
        else if (pa[i] < **ptr_min)
            *ptr_min = (int*)(pa + i)
    }
}
```

```
int main(void)
{
    int a[SIZE];

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

    int* pmin;
    int* pmax;

    get_array_min_max(a, SIZE, &pmin, &pmax);

    printf("max = %d ve dizinin %d indisli elemani\n", *pmax, pmax - a);
    printf("min = %d ve dizinin %d indisli elemani\n", *pmin, pmin - a);

    swap(pmin, pmax);

    print_array(a, SIZE);

}
```

Örnek pekiştirme

```c
void func(int** ptr)
{
    *ptr = NULL;
}

int main(void)
{
    int x = 567;
    int* p = &x;

    if (p == NULL)
        printf("dogru\n");
    else
        printf("yanlis\n");

    func(&p); //int *   int **

    if (p == NULL)
        printf("dogru\n");
    else
        printf("yanlis\n");


}
```

Fonksiyon çağrısını parametresini *p ile yapıp p gönderseydik dışardaki pointer nesnesi değişmezdi

Ancak suan pıinter nesneyi değişrmek için **p ile parametre gönderdim ve argümanı &p ile verdim

*p artık pointer değişkenimin değerini değiştirecek.

//

Örnek 2

Dizinin elemanlarını değiştirmek için

```
#include "nutility.h"

void foo(int *, int )
int main(void)
{
    int a[10] = { 1, 3, 6, 78, 5, 1, 4, 7, 7, 8, };

    foo(a, 10);

}
```

A dizi adresi ile int* parametreli dizi oluştururdum.

Dizimiz pointer dizisi olsaydı.

```
#include "nutility.h"

void foo(int** p, int size)
{
    while (size--)
        *p
        ++p
    }
}

void foo(int** p, int size)
{
    for (int i = 0; i < size; ++i) {
        p[i]
    }
}

int main(void)
{
    int* a[10];

    foo(a, 10);
    foo(&a[0], 10);

}
```

Dizi int * olduğu için a ifadesi pointer değişkenlerinin adresi yani int **

Oldu.

Örnek string fonksiyonu

Bu işlemi yapıcak bir fonksiyon yazalım.

```
char* p[] = { ... }

for (size_t i = 0; i < asize(p); ++i) {
    printf("%s ", p[i]);
}
```

```
void print_names(char** pa, size_t size)
{
    for (size_t i = 0; i < size; ++i) {
        printf("%s ", pa[i]);
    }
    printf("\n");
}

int main()
{

    char* p[] = { ... }

    print_names(p, asize(p));
}
```

Ekranayazı yazdık

```
void print_names(char** pa, size_t size)
{
    while (size--) {
        printf("%s ", *pa++);
    }
    printf("\n");
}
```

pa[i] ve*p dizinin gösterdiği stringlerin adresini tutuyor bu yüzden ekrana yazar hatta oynama bile yapabilirim çünkü pointer adresleri ile fonksiyonu çağırdım.

Const anahtra sözcüğü

Örnek hangileri sentax hatası.

```
int main(void)
{
    int x = 10;
    int y = 33;
    int* p1 = &x;
    int* p2 = &y;

    int** ptr = &p1;

    ptr = &p2;
    *ptr = &y;
    **ptr = 888;
}
```

Hiçbiri hatalı değil

```
int main(void)
{
    int x = 10;
    int y = 33;
    int* p1 = &x;
    int* p2 = &y;

    int** const ptr = &p1;

    //ptr = &p2;
    //*ptr = &y;
    //**ptr = 888;
}
```

Birinci hatalı

```c
int main(void)
{
    int x = 10;
    int y = 33;
    int* p1 = &x;
    int* p2 = &y;

    int* const *  ptr = &p1;

    //ptr = &p2;
    //*ptr = &y;
    //**ptr = 888;
}
```

İkinci hatalı

```c
int main(void)
{
    int x = 10;
    int y = 33;
    int* p1 = &x;
    int* p2 = &y;

    int const**  ptr = &p1;

    ptr = &p2;
    *ptr = &y;
    **ptr = 888;
}
```

Üçüncü hatalı

Const neyin önündeyse onlar hatalıdır.

```c
int main(void)
{
    int x = 10;
    int y = 33;
    int* p1 = &x;
    int* p2 = &y;

    const int **  const ptr = &p1;

    ptr = &p2;
    *ptr = &y;
    **ptr = 888;
}
```

(local variable) const int **const ptr
Search Online

1 ve 3 hatalı

Hepsi const olsaydı hepsi hatalı olurdu.

Örnek const nerede olmalı

```c
void print_names(char*const * pa, size_t size)
{
    while (size--) {
        printf("%s ", *pa++);
    }
    printf("\n");
}


int main()
{

    char* p[] = { ... }

    print_names(p, asize(p));
}
```

Amacımız elemanları değişmemek için yazdığımız için araya yazdık ama ** dan sonrada const olabilirdi.

Örnek elelmanları alfabetik sıralama sıralayan fonksiyon

```
void sort_names(char** pa, size_t size)
{
    for (size_t i = 0; i < size - 1; ++i) {
        for (size_t k = 0; k < size - 1 - i; ++k) {
            if (strcmp(pa[k], pa[k + 1]) > 0) {
                char* ptemp = pa[k];
                pa[k] = pa[k + 1];
                pa[k + 1] = ptemp;
            }
        }
    }
}
```

```
  sort_names(p, asize(p))
  print_names(p, asize(p));
```

İçerdeki pointerları char** sayesindeadresten kendine ulşarak fonksiyon içinde sıraladık.

Bildirimlerde const kullanmak

.h yerine .cye koy işe yaramıcaksa

```
//volkan.h

void func(int x);

//volkan.c
void func(const int x)
{

}
```

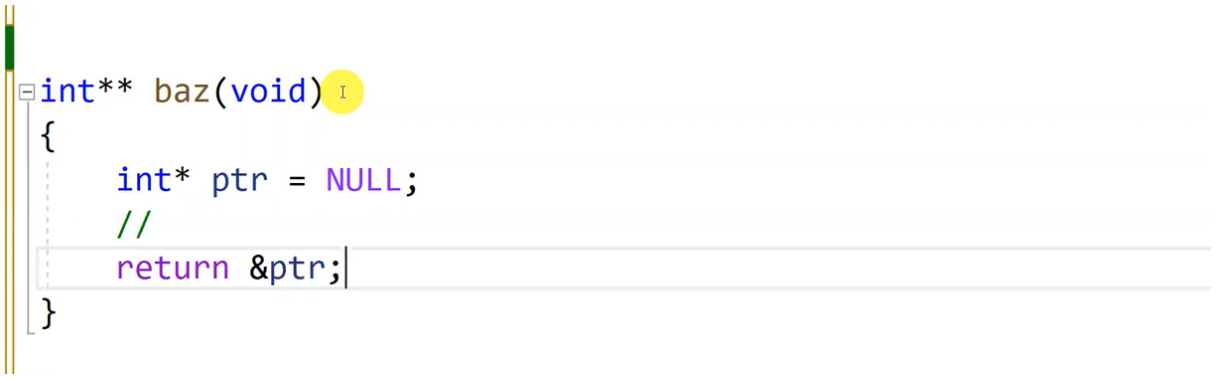
Parametre fonksiyon değişteremz ise koymana gerek yok const *int p olabilir

Ancak int* const p saçma adresi zaten fonksiyon değişmez.

//

//

Çok önemli örnek tanımsız davranış



Bu kod tanımsız davranıştır

Çünkü otomatik ömürlü değişken adresi döndiriyor.

Sınav sorusu bu kod ne yapar

```c
void foo(int** p1, int** p2)
{
    int x = **p1;
    **p1 = **p2;
    **p2 = x;
}

int main(void)
{
    int x = 34;
    int y = 67;
    int* p1 = &x;
    int* p2 = &y;

    foo(&p1, &p2);


}
```

Bu kod x ve ye yi swap yapar.

//

Void pointer

```c
//türü void olan bir nesne olamaz
//sizeof operastörünün operandı void türü olamaz
// void is an incomplete type
// elemanları void türden
// void[]
```

Örnek

```
int main(void)
{
    int x = 34;
    double dval = 4.5;

    void* vp = &x;

    vp = &dval;

    *vp

}
```

Void pointer ı direference edemeyiz eğer *vp yaparsak bu sentax hatası olur.

Örnek kod hatalımı

```
int main(void)
{
    void* vp1;
    void* vp2 = &vp1;
}
```

Hayır her adresi tututabilir.

Void** adreside bir adres

Örnek void * dönüşümü

```c
int main(void)
{
    int x = 10;
    void* vptr = &x;


}
```

C ve c++ da bu kod legal

```c
int main(void)
{
    int x = 10;
    void* vptr = &x;
    int* iptr = vptr;  //Cde legal ve dogru


}
```

C de legal ancak c++ da sentax hatası.

C++ için (int*) tür dönüştürme operatörü kullanılmalıydı.

```c
void* foo(void);

int main(void)
{
    int* ptr;

    ptr = (int*)foo();


}
```

C dede tür kullan niyetimiz belli olsun kural değil ama tavsiyedir.

Mülakat sorusu ///////////////

Hangisi const

```
#include <stdlib.h>

typedef int* IPTR;

int main(void)
{
    int ival = 34;
    int x = 45;

    const IPTR p = &ival;
    *p = 44;
}
```

Burada const p yi niteliyor.

Yani p = &y sentax hatası

Ancak *p = 44 legal bir kod unutma.

Örnek

```
int main(void)
{
    int x = 10;
    void* vptr = &x;
    //
    int* ip = &x;

    ///

    if (vptr == ip)
}
```

Legal bir karşılaştırma

Türden bağımsız fonksiyon yazmka

Adresteki türleri byte byte takas yapıyoruz.

Örnek gswap (void*vp1 , void *vp2 , size_t size)

```
void gswap(void* vp1, void* vp2, size_t sz)
{
    char* p1 = (char*)vp1;
    char* p2 = (char*)vp2;

    while (sz--) {
        char temp = *p1;
        *p1++ = *p2;
        *p2++ = temp;
    }
}
```

```
int main(void)
{
    int x = 4565, y = 1356;
    int a1[] = { 1, 5, 7, 9 };
    int a2[] = { -1, -5, -7, -9 };

    gswap(&x, &y, sizeof(int));
    printf("x = %d  y = %d\n", x, y);
    double dx = 4.436, dy = 75.873245;
    gswap(&dx, &dy, sizeof(double));
    printf("dx = %f  dy = %f\n", dx, dy)

    gswap(a1, a2, sizeof(a1));

    print_array(a1, 4);
    print_array(a2, 4);


}
```

```
Microsoft Visual Studio Debug Console
x = 1356  y = 4565
dx = 75.873245  dy = 4.436000
 -1  -5  -7  -9
------------------------------
  1   5   7   9
------------------------------

D:\KURSLAR\NISAN_2022_C\Release
Press any key to close this win
```

Soru 2 dizinin ilk 3 elemanı ile son 3 elemanını takas edebilirmiyim.

```
int main(void)
{
    int a[SIZE];
    int b[SIZE];
    randomize();

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

    gswap(a, b + SIZE - 3, 3 * sizeof(int));
    print_array(a, SIZE);
    print_array(b, SIZE);


}
```

Adresi başlatacağımız yerler farklı olur ve sizeof ile 3 int boyutu gönderiririz

Standart kütüphane

```
tüm adres türlerinden void* türüne (örtülü) dönüşüm var
C dilinde void * türünden diğer pointer türlerine örtülü dönüşüm var

<string.h>

void* memset(void *vpbuf, int c, size_t);
void *memcpy(void* vpdest, const void *vpsource, size_t);
void *memmove(void* vpdest, const void *vpsource, size_t);
void* memchr(const void *vp, int c, size_t);
int memcmp(const void* vp1, const void* vp2, size_t);
```