Ders 42

1 Örnek rasgele parola üretmi

Fpbksyon otomatik ömürl, nesne adresi döndüremez.

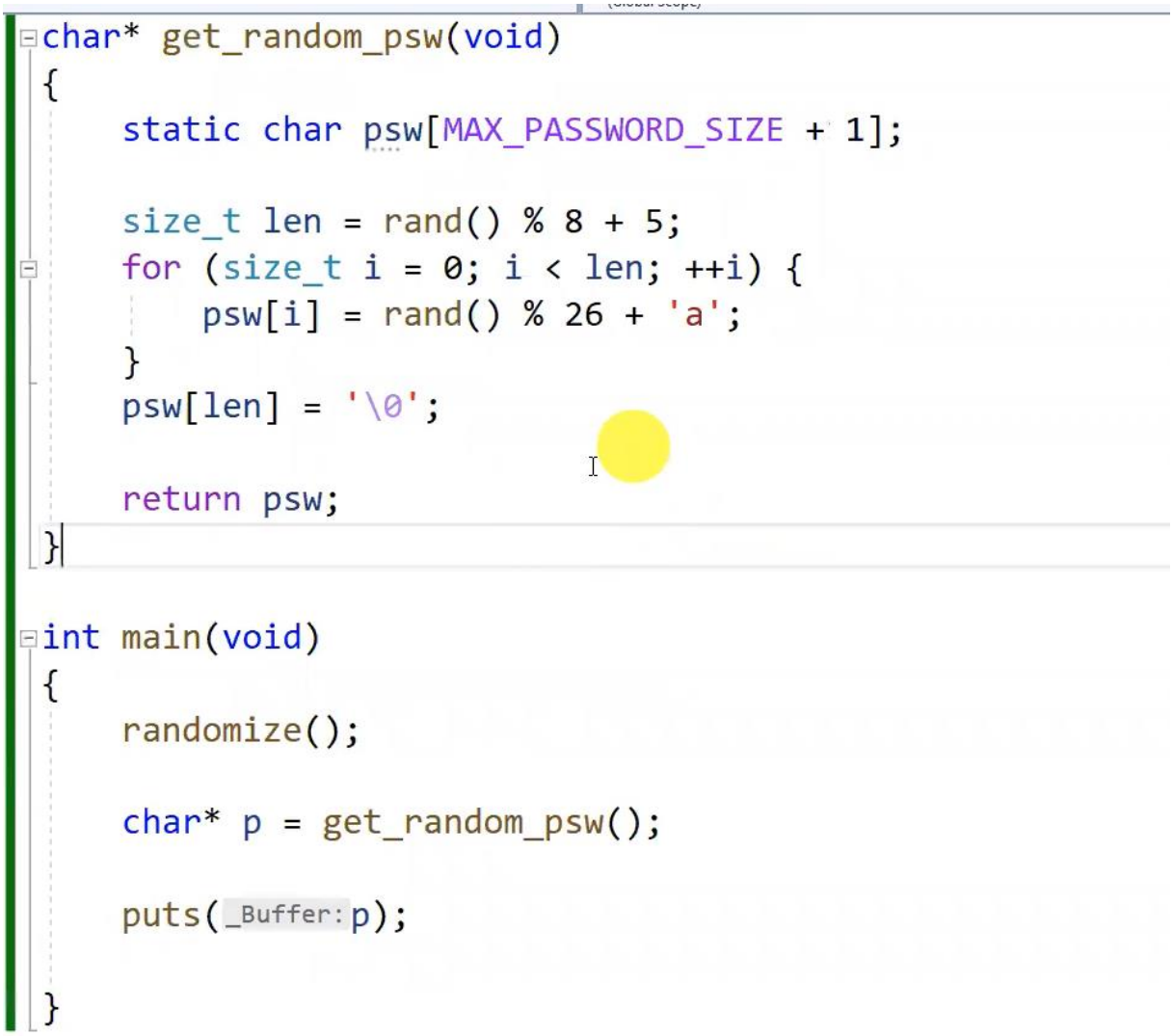
```
char* get_random_psw(void);

// statik ömürlü nesne adresi
//      - global nesne adresi
//      - static yerel nesne adresi
//      - string literali

// dinamik ömürlü nesne adresi
// çağıran kodun kendisine gönderdiği adresi döndü
```

Sadece bu adresleri döndürebilir.

Mutlaka dökümandan bakmamız lazım staik ömürlü her zaman aynı nesne adresi dönecek



```
char* get_random_psw(void)
{
    static char psw[MAX_PASSWORD_SIZE + 1];

    size_t len = rand() % 8 + 5;
    for (size_t i = 0; i < len; ++i) {
        psw[i] = rand() % 26 + 'a';
    }
    psw[len] = '\0';

    return psw;
}

int main(void)
{
    randomize();

    char* p = get_random_psw();

    puts(_Buffer: p);

}
```

Burada static ömürlü bir dizi içinde uzunluk bulduk bu uzunluk kadar rasgele sayı aldık her sayıya a karakteri ekleyerek harf karakteri ürettik.

Sonunda null karakter olmasını istediğim için for sonunda null yazıp dizinin adresini verdim ama unutulmamalı bu fonksiyonu her çağırdığımda aynı dizinin adresi dönücek.

Fonksyonu birkaç kez çağıralım

```
char* get_random_psw(void)
{
    static char psw[MAX_PASSWORD_SIZE + 1];

    size_t len = rand() % 8 + 5;
    for (size_t i = 0; i < len; ++i) {
        psw[i] = rand() % 26 + 'a';
    }
    psw[len] = '\0';

    return psw;
}

int main(void)
{
    randomize();

    char* p[5];

    for (int i = 0; i < 5; ++i) {
        p[i] = get_random_psw();
    }

    for (int i = 0; i < 5; ++i) {
        puts(p[i]);
    }

}
```

Burada kodu yazanın amacı rasgele sayıları üretmekti ancak her seferinde aynı dizi üzerinde işlem yapıldığı için en son ekrana aynı değerler yazdı.

Hata fonksiyonda değil doküman edilmişti bu değerleri anında kullansak olabilirdi ama uygun kod yazmadık . bu işlemi kodu çağırmadan önce değeri kopyalayıp bir daha paralo alsam olurdu .

2 boyutlu dizi oluşturup kopyalayabilirim.

```c
int main(void)
{
    randomize();
    char psw[5][40];

    for (int i = 0; i < 5; ++i) {
        strcpy(psw[i], get_random_psw());
    }

```

Gibi tutulabilir fonksiyona göre kod yazılması gerekiyor.

//

Örnek strdup

```c
#define         SIZE        100

int main(void)
{
    char str[SIZE];

    printf("bir yazi girin: ");
    sgets(str);

    char* p = _strdup(str);
    //null pointer
    _strrev(p);
    printf("(%s) (%s)\n", str, p);
    free(p);

}
```

Alınan diziyi kopyalayan hazır kod bunu dinamik bellek kullanarak yapıyor.

Bu yüzden daha sonra free yapıyoruz memory lick olmasın diye

```
char* mystrdup(const char* p)
{
    char* pd = (char *)malloc(strlen(p) + 1);
    if (!pd) {
        return NULL;
    }

    return strcpy(pd, p);
}
```

Strdup u kendımızyazdık ve her çağırana farklı adres döndürücek bunları free yapmak kodu yazanın sorumluluğu

//

Dinamik bellek ile rasgele parola

```
#define         MAX_PASSWORD_SIZE    40

char* get_random_psw(void)
{
    size_t len = rand() % 8 + 5;
    char* pd = malloc(len + 1);
    if (!pd) {
        fprintf(stderr, "cannot allocate memory!\n");
        exit(EXIT_FAILURE);
    }

    for (size_t i = 0; i < len; ++i) {
        pd[i] = rand() % 26 + 'a';
    }
    pd[len] = '\0';

    return pd;
}
```

```c
int main(void)
{
    char* p[10];

    randomize();

    for (int i = 0; i < 10; ++i) {
        p[i] = get_random_psw();
    }

    ///

    for (int i = 0; i < 10; ++i) {
        puts(p[i]);
    }

    for (int i = 0; i < 10; ++i) {
        free(p[i]);
    }


}
```

Dinamik bellek kullanarak kendi kodumuzu yazdık.

Her adresi pointer dizisinde tuttum ve işim bitince free ettim

//

ÖRNEK 2 YAZIYI DİNAMAİK BELLKETE BİRLEŞTİREN KOD

```c
char* strconcat(const char* p1, const char* p2)
{
    char* pd = (char*)malloc(strlen(p1) + strlen(p2) + 1);
    if (!pd) {
        fprintf(stderr, "bellek yetersiz\n");
        exit(EXIT_FAILURE);
    }

    return strcat(strcpy(pd, p1), p2);

}
```

```c
int main(void)
{
    char s1[SIZE];
    char s2[SIZE];

    printf("birinci yaziyi girin: ");
    sgets(s1);
    printf("ikinci yaziyi girin: ");
    sgets(s2);
    char* pd = strconcat(s1, s2);

    printf("(%s) + (%s) = (%s)\n", s1, s2, pd);
```

SORULAR TANIMSIZ OLURMUYDU.

```
        exit(EXIT_FAILURE);
    }

    return strcat(strcat(pd, p1), p2);

}
```

OLURDU ÇÜNKÜ STRCAT NULL DEĞER arıyot. Malloc pd yi null ali vermiyor.

```
        exit(EXIT_FAILURE);
    }

    *pd = '\0';
    return strcat(strcat(pd, p1), p2);

}
```

Bu kodda tanımsız olmazdı strcat null bulup yazardı.

Bu koddaki hata nedir.

```
int main(void)
{
    char s1[SIZE];
    char s2[SIZE];

    printf("birinci yaziyi girin: ");
    sgets(s1);
    printf("ikinci yaziyi girin: ");
    sgets(s2);
    char *pd = strconcat(strconcat(s1, s2), "hasan");

    printf("(%s) + (%s) = (%s)\n", s1, s2, pd);
    //

    free(pd);

}
```

Memory lezk var

İlk strconcat döndürdüğü değeri 2. Ye verdi ama ilkinin adresini saklamadık bu yüzden free ile geri veremiyeceğim

Problem 2 tane bellek ürettim ama free yapamadım

Kd düzgün çalışır ama ilk kodu free yapamayız artık

//

```
int n;
//......

int *pd = (int*)malloc(n * sizeof(int));
=======================================================

int **pd = (int**)malloc(n * sizeof(int*));
```

Eğer elemanları nesne değeri olan bir bellek bloğo tanımlayacaksam bunu int * türünden yapmalıyım

Ancak lelemanlar nesne adresi olacaksa yanı elemanlarım int * ve sayısı belli değilse bu durumda bellek bloğunu int* türüne göre açarım bunu int** değişkende saklarım di reference olunca adresleri döndürür.

//

2boyutlu bir diziyi dinamik bellekte tanımlama yöntemleri 2 farklı yöntem var

2i yöntem her bir dizi için farklı malloc oluşturduk

```c
int main(void)
{
    size_t row, col;

    printf("matrisin satir ve sutun sayisini girin: ");
    scanf("%zu%zu", &row, &col);
    int** pd = (int **)malloc(row * sizeof(int*));
    if (!pd) {
        fprintf(stderr, "cannot allocate memory!\n");
        exit(EXIT_FAILURE);
    }

    for (size_t i = 0; i < row; ++i) {
        pd[i] = (int*)malloc(col * sizeof(int));
        if (!pd[i]) {
            fprintf(stderr, "cannot allocate memory!\n");
            exit(EXIT_FAILURE);
        }
    }
```

```c
    randomize();

    for (size_t i = 0; i < row; ++i) {
        for (size_t k = 0; k < col; ++k) {
            pd[i][k] = rand() % 10;
        }
    }


    //...
    for (size_t i = 0; i < row; ++i) {
        for (size_t k = 0; k < col; ++k) {
            printf("%d", pd[i][k]);
        }
        printf("\n");
    }


    for (size_t i = 0; i < row; ++i) {
        free(pd[i]);
```

Sonundada free(pd) olucak burda her dizi için bellek ayırdık bu bellekleri yine farklı bir bellekte adreslerini sakladık

Burda çok boyutlu dizilerdeki gibi bütün elemanlar ardışık değil.

1.yönhepsini yanyana sıralı olarak tek bir dizi gibi oluşturup 1 adet malloc çağıralım.

```
printf("matrisin satir ve sutun sayisini girin: ");
scanf("%zu%zu", &row, &col);

int* pd = (int*)malloc(row * col * sizeof(int));
//code

for (size_t i = 0; i < row; ++i) {
    for (size_t k = 0; k < col; ++k) {
        pd[i * col + k] = rand() % 10;
    }
}

for (size_t i = 0; i < row; ++i) {
    for (size_t k = 0; k < col; ++k) {
        pd[i * col + k] = rand() % 10;
    }
}
```

```
    for (size_t i = 0; i < row; ++i) {
        for (size_t k = 0; k < col; ++k) {
            printf("%d", pd[i * col + k]);
        }
        printf("\n");
    }

    int n = row * col;
    int* p = pd;

    free(pd);
}
```

Burada tek bir bellek oluşturdu bu yüzden p[i][k] gibi yazamıyoruz yazarsak hata olırız bunun yerine 2 boyutlu gibi davranıyoruz.

Ama bu dizidede elemanlar sıralı bir adresten sonra bir diğer dizi adresi ardışık.

3. yöntem hem ardışık hemde matris rotasyonunda kullanıcam

```
{
    size_t row, col;

    printf("matrisin satir ve sutun sayisini girin: ");
    scanf("%zu%zu", &row, &col);

    int* p = (int*)malloc(row * col * sizeof(int));
    //kontrol
    int** pp = (int **)malloc(row * sizeof(int*));
    //kontrol

    for (size_t i = 0; i < row; ++i) {
        pp[i] = p + (i * col);
    }
```

```
    }

    for (size_t i = 0; i < row; ++i) {
        for (size_t k = 0; k < col; ++k) {
            pp[i][k] = rand() % 10;
        }

    }

    ///

    for (size_t i = 0; i < row; ++i) {
        for (size_t k = 0; k < col; ++k) {
            printf("%d", pp[i][k]);
        }
        printf("\n");
    }

    free(pp);
    free(p);
}
```

Bu yöntem ile sadece 2 ayrı malloc çağırarak sadece 2 bellek kullandım ama bu sefer 2 boyutlu dizi aritmetiğindende yararlanmayı denedim.

Burada free sırasının hiçbir önemide yok.

//

Calloc fonsiyonu

```
int main(void)
{
    int n;

    printf("kac elemanli int dizi: ");
    scanf("%d", &n);

    int* pd = (int*)malloc(n * sizeof(int));
    /

    print_array(pd, n);

    free(pd);
}
```

Malloc belleği çöp değeriyle veriyor. 0 lamak için memset ile yazılmalı.

```
int main(void)
{
    int n;

    printf("kac elemanli int dizi: ");
    scanf("%d", &n);

    int* pd = (int*)malloc(n * sizeof(int));
    //control

    //memset(pd, 0, n * sizeof(int));

    print_array(pd, n);

    free(pd);
}
```

Calloc bu 0 lama işlerini kendisi yapıyor.

```
    int n;

    printf("kac elemanli int dizi: ");
    scanf("%d", &n);

    int* pd = (int*)calloc(n, sizeof(int));
    //control

    print_array(pd, n);

    free(pd);
}
```

//

Realloc fonksiyonu

Realloc ataması

```
pd = (int *)realloc(pd, (n + x) * sizeof(int));
```

Bu şekilde yapılıyor ancak riskli eğerki realloc başarısız olursa null pointer döndürücek ve eski bloğu free etmeyecek bu durumda eski bloğun adresinide kaybetmiş oluruz.

Eğerki realloc başarısız olduğunda program sonlanıcaksa yazılabilir ama sonlanmıyacaksa

Başka pointere yazılmalı.

Örnek realloc

```c
    int n;

    printf("kac tam sayi:  ");
    scanf("%d", &n);

    int* pd = (int*)malloc(n * sizeof(int));
    if (!pd) {
        fprintf(stderr, "bellek yetersiz\n");
        return 1;
    }

    randomize();
    set_array_random(pd, n);
    print_array(pd, n);

    int n_add;
    printf("kac tam sayi eklensin:  ");
    scanf("%d", &n_add);
    pd = (int*)realloc(pd, (n + n_add) * sizeof(int));
    if (!pd) {
        fprintf(stderr, "bellek yetersiz\n");
        return 1;
    }

    set_array_random(pd + n, n_add);
    print_array(pd, n + n_add);
    //
```

```
Microsoft Visual Studio Debug Console
kac tam sayi:  40
658 345 493 932 229 922 659 522 163 746 242 462 444 532 130 248 125 191 114 434
221 581  79 167 578  60 414 897 541 356 110 398 451  82 954 843  37 980 405 257
--------------------------------------------------------------------------------
kac tam sayi eklensin:  60
658 345 493 932 229 922 659 522 163 746 242 462 444 532 130 248 125 191 114 434
221 581  79 167 578  60 414 897 541 356 110 398 451  82 954 843  37 980 405 257
 27 246 190 447 429 101 447 736 202 411 280 399  60 826 121 536  67 331 408 953
833 905 328 665 894 394 195 455 478 326 200 430 143 443 592 424  63 224 982   0
870 633 342 965 535 594 672 980 997 448 668 972 652 164 391 768  77 977 855 880
--------------------------------------------------------------------------------

D:\KURSLAR\NISAN_2022_C\Debug\NISAN_2022_C.exe (process 27544) exited with code 0.
Press any key to close this window . . .
```

Malloc ile dizi oluşturduk bu diziyi daha sonra arttırmak için bu sefer realloc kullandık iki adresi de aynı pointere yazdım çünkü realloc başarısız olursa program biticek

Sonra bu değerleri set edip ekranda yazdırdım.

Örnek ralloc ve malloc adresi ekrana yazdırma

Aynı yerdede büyültebilir farklıda olabilir 2 kere denedik biri ayn biri farklı oldu.