Algoritmalar ve Programlama I – BLM 103

Hafta 12: Karakterler, Girdi/Çıktı Formatlama, Dosya Okuma ve Yazma

Fenerbahçe Üniversitesi

12. Hafta İçeriği

- Standart C Kütüphanesi
- I/O Akışları
- putchar
- getchar
- Giriş Çıkış Formatlama
- Dosyadan Okuma ve Dosyaya Yazma İşlemleri

Giriş Çıkış Fonksiyonları

- stdio.h kütüphanesinde temel giriş çıkış fonksiyonları bulunmaktadır.

Fonksiyon	*Açıklama*
`putchar`	Bir ascii karakterini ekrana gösterir.
`getchar`	Bir ascii karakterini klavyeden alır.
`printf`	Ekrana formatlanmış değişkeni gösterir,
`scanf`	Formatlanmış girdiyi alır.
`fopen`	Okuma ve yazma için bir dosyayı açar.
`fprintf`	Bir dosyaya formatlanmış bir string yazar.
`fscanf`	Bir dosyadan formatlanmış bir string okur.

Komut Satırından Giriş ve Çıkışlar

Program'a giriş olarak komut satırından argüman verilebilir.

```
#include<stdio.h>

int main(int argc, char* argv[])
{
   int counter;
   printf("Program Name Is: %s",argv[0]);
   if(argc==1)
      printf("\nNo Extra Command Line Argument Passed Other Than Program Name");
   if(argc>=2)
   {
      printf("\nNumber Of Arguments Passed: %d",argc);
      printf("\n----Following Are The Command Line Arguments Passed----");
      for(counter=0;counter<argc;counter++)
         printf("\nargv[%d]: %s",counter,argv[counter]);
   }
   return 0;
}
```

Komut Satırından Giriş ve Çıkışlar

Program'a giriş olarak komut satırından argüman verilebilir.

```
#include<stdio.h>

int main(int argc, char* argv[])
{
 …
return 0;
}
```

```
test.exe arg1 arg2
```

argc (Argument Count): Programa kaç adet argüman verildiğini ifade eder.

argv (Argument Vector): Programa verilen argümanların tutulduğu dizidir.

Karakter Giriş ve Çıkışlar

- `putchar(c)` c karakterinin içeriğini ekrana basar.
- `getchar()` Bir ascii karakteri okur.

-

```
char c = 'h';
```

-

```
...
putchar(c);
putchar('h');
putchar(104);
```

Her üç fonksiyonda ekrana ‘h’ yazmaktadır.

Buffer'lanmış Giriş ve Çıkışlar

- Karakterler bellekte giriş ve çıkış operasyonları esnasında depolanırlar

- Klavyeden girişler
  - Karakterler klavyeden giriş alındıkça bellekte depolanırlar. \n yani enter karakteri girildiğinde bellekteki karakterler hedef saklama alanına aktarılırlar.
  - Bu yapı kullanıcı doğru giriş verdiğinde, girişin kopyalanmasını sağlar.

- Çıkışlar
  - Karakterler sadece \n yani enter karakteri alındığında dışarı çıkartılırlar.

Giriş Tamponlama

```
printf("Giriş karakteri 1:\n");
inChar1 = getchar();

printf("Giriş karakteri 2:\n");
inChar2 = getchar();
```

- İlk çıktı görüldükten sonra, program yeni bir karakter almak üzere bekler.
- Enter karakteri beslenene kadar program ilerlemez.
- Enter'e basıldığında, yenisatır karakteri alınır ve program ikinci çıktıyı verip, yeniden \n karakteri alana kadar bekler.

Çıkış Tamponlama

```
putchar('a');

for (i=0; i<100; i++) sum += i;

putchar('b');
putchar('\n');
```

- İlk çalıştırılan komut ile ekrana bir çıktı verilmez. Çünkü henüz \n karakteri gelmediği için bellekte beklemektedir.

Formatlanmış Giriş ve Çıkışlar

- Printf ve Scanf fonksiyonları, veri türleri arasında dönüşümleri gerçekleştirebilirler.

- Format, ekrana gösterilecek veya alınacak olan değerin biçiminin belirtilmesidir.

`%d`	işaretli tam sayılar
`%f`	işaretli ondalık sayılar
`%x`	işaretsiz hexadecimal sayılar
`%b`	işaretsiz ikilik tabandaki sayılar
`%c`	ASCII karakteri
`%s`	ASCII metni

Özel Karakterler

- Bazı özel karakterler tek bir karakter ile temsil edilememektedir.
  - Yeni satır, tab vs.. gibi

- Özel karakterlerin kullanımı:
-

`\n`	yeni satır
`\t`	tab
`\b`	silme (backspace)
`\\`	\ karakteri
`\'`	‘ karakteri
`\"`	" karakteri
`\xnnn`	ASCII kodu *nnn* (hexadecimal)

printf

- Verilen formatlama biçimine göre ekrana gösterir.
-

```
int a = 100;
int b = 65;
char c = 'z';
char banner[10] = "test!";
double pi = 3.14159;

printf("The variable 'a' decimal: %d\n", a);
printf("The variable 'a' hex: %x\n", a);
printf("The variable 'a' binary: %b\n", a);
printf("'a' plus 'b' as character: %c\n", a+b);
printf("A char %c.\t A string %s\n A float %f\n", c, banner, pi);
```

printf

- Printf fonksiyonu çalıştıktan sonra ekrana kaç karakter bastığını döndürür.

```
int main()
{
    int result;
    result = printf("Hello\n");
    printf("%d\n",result);
    return 0;
}
```

Ekrana 6 yazdıracaktır.

printf

```
#include<stdio.h>
main()
{
        int a,b;
        float c,d;

        a = 15;
        b = a / 2;
        printf("%d\n",b);
        printf("%3d\n",b);
        printf("%03d\n",b);

        c = 15.3;
        d = c / 3;
        printf("%3.2f\n",d);
}
```

Çıktılar

```
7
  7
007
5.10
```

printf

- %d (Tamsayı yazdırır)
- %6d (en az 6 karakter genişliğinde tamsayı yazdırır, boşluk varsa boşluk karakteri ile doldurur)
- %06d (en az 6 karakter genişliğinde tamsayı yazdırır, boş yer var ise 0 ile doldurur)
- %f (Ondanlık sayı yazdırır)
- %4f (Genişliği en az 4 olan bir ondalık sayı yazdırır)
- %.4f (Ondalık bölümü 4 karakter olan ondalık sayı yazdırır)
- %3.2f (En az 3 genişliğinde ve 2 basamağı ondalık olan sayı yazdırır)

printf

- The printf(“:%s:\n”, “Hello, world!”);
  - Tüm metni basar
- The printf(“:%15s:\n”, “Hello, world!”);
  - İlk 15 karakteri basar, 15 karakterden küçük ise, boşluk ile doldurur. Boşlukları soldan itibaren doldurur.
- The printf(“:%.10s:\n”, “Hello, world!”);
  - Sadece ilk 10 karakteri bastırır.
- The printf(“:%-10s:\n”, “Hello, world!”);
  - En az 10 karakter bastırır, 10 karakterden küçük ise sağdan itibaren boşluk doldurur.
- The printf(“:%15.10s:\n”, “Hello, world!”);
  - Toplamda 15 karakter bastıracaktır, metnin 10 karakterini bastırır, kalanlar soldan itibaren boşluk ile doldurulur.
- The printf(“:%-15.10s:\n”, “Hello, world!”);
  - Toplamda 15 karakter bastıracaktır, metnin 10 karakterini bastırır, kalanlar sağdan itibaren boşluk ile doldurulur.

printf

```
#include<stdio.h>

main()
{
        printf(":%s:\n", "Hello, world!");
        printf(":%15s:\n", "Hello, world!");
        printf(":%.10s:\n", "Hello, world!");
        printf(":%-10s:\n", "Hello, world!");
        printf(":%-15s:\n", "Hello, world!");
        printf(":%.15s:\n", "Hello, world!");
        printf(":%15.10s:\n", "Hello, world!");
        printf(":%-15.10s:\n", "Hello, world!");
}
```

Çıktılar

```
:Hello, world!:
:  Hello, world!:
:Hello, wor:
:Hello, world!:
:Hello, world!  :
:Hello, world!:
:    Hello, wor:
:Hello, wor    :
```

Unutulmuş Argümanlar

- Bastırılması için gerekli argüman verilmediğinde ne olur?

- `printf("Ekrana bastir %d\n");`

- `%d` ilk elemanın beklendiği adresteki değer ne ise, tam sayıya çevrilerek bastırılacaktır.

  Diğer bir değiş ile, bir şeyler ekrana bastırılacaktır ancak değeri anlamsız olacaktır.

scanf

- Beklenen formata göre girişleri alıp, değişkenlerin içerisine yerleştirmektedir.

```
char isim[100];
int dYili, dGunu, dAyi;
double ortalama;

scanf("%s %d/%d/%d %lf", &isim, &dYili, &dGunu, &dAyi,
&ortalama);
```

scanf Dönüşümü

- Scanf fonksiyonu boşluk karakterlerini gözardı edip, ascii karakterlerini okumaktadır.

Verilen formata göre:

- `%d` Rakamı almaya başlar ve ilk rakam olmayana kadar devam eder.
- `%x` Rakamı almaya başlar ve rakam olmayana kadar devam eder.
- `%s` Metni almaya başlar ve ilk boşluk karakterine kadar devam eder.

scanf Dönen Değeri

- Scanf fonksiyonu kaç adet başarılı dönüşüm yaptığını geri döndürmektedir.
  - Girilen girişin doğru olup olmadığı bu değerden kontrol edilebilir.

- Örnek:

```
scanf("%s %d/%d/%d %lf", isim, &ay, &gun, &yil, &ortalama);
```

- *Giris Metni* *Dönen Değer*
  `Test 02/16/69 3.02` 5
- `Test 02 16 69 3.02` 2

/ girişi olmadığı için, beklenen ile uyuşmayıp çıkar.

Dosya Giriş ve Çıkışları

- Program kapandığında bellekte olan tüm değişkenler silinmektedir. Bazı veriler daha sonra okunmak üzere sabit diskte tutulmak istenebilir.
- Bu durumda bir dosyaya yazma ve okuma işlemlerinin yapılması gerekir.
- Bu işlem için dosya işaretçisi kullanılır:

```
FILE *infile;
```

- `FILE` veri tipi <stdio.h> kütüphanesinde tanımlıdır.

fopen

- Fopen fonksiyonu bir dosyayı okuma veya yazma amacıyla açılmasını sağlar.
- `FILE *fopen(char* dosyaAdi, char* mod);`

- İlk Argüman: `dosyaAdi`

- İkinci Argüman: `mod`
  - Kullanımı:
    `"r"` -- Okuma
    `"w"` – Yazma, dosyanın başından itibaren başlar.
    `"a"` – Yazma, dosyanın sonuna ekleme yapar (append)

fprintf ve fscanf

- Dosya açıldıktan sonra, dosyaya yazma ve okuma yapmak için `fscanf()` ve `fprintf()` kullanılabilir.

- Bu fonksiyonlar scanf ve printf gibi çalışmaktadırlar. Farklı olarak yazacakları veya okuyacakları yerin işaretçisini alırlar.

- `fprintf(outfile, "Yazilacak metin %d\n", x);`

- ```
  fscanf(infile, "%s %d/%d/%d %lf",
         isim, &ay, &gun, &yil, &ortalama);
  ```
  ```

Örnek 1: Dosyaya yazma

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int num;
FILE* fptr;
int sonuc = fopen_s(&fptr, "program.txt", "w");
if (sonuc != 0)
{
printf("Hata!");
exit(1);
}
printf("Sayi giriniz: ");
scanf_s("%d", &num);
fprintf(fptr, "%d", num);
fclose(fptr);
return 0;
}
```

Örnek 2: Dosyadan okuma

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int num;
FILE* fptr;
int sonuc;
if ((sonuc = fopen_s(&fptr, "dosya.txt", "r")) != 0 ) {
printf("Okuma hatasi");
exit(1);
}
fscanf_s(fptr, "%d", &num);
printf("Deger = %d", num);
fclose(fptr);

return 0;
}
```

Dosya İşlem Fonksiyonları

Fonksiyon	Açıklama
fopen()	Yeni bir dosya yazmak veya okumak üzere açar
fclose()	Doyayı kapatır
getc()	Dosyadan bir karakter okur
putc()	Dosyaya bir karakter yazar
fscanf()	Dosyadan belirli bir miktar veri okur
fprintf()	Dosyaya belirli bir miktar veri yazar
getw()	Dosyadan tamsayı okur
putw()	Dosyaya tamsayı yazar
fseek()	Dosyada istenen noktaya götürür
ftell()	Dosyadaki şu andaki bulunulan noktayı döndürür
rewind()	Başlangıç noktasına götürür

Örnek 3: Okuma ve Yazma

```
#include<stdio.h>

int main()
{
FILE* fp, *fp2;
char ch;
int rtrn = fopen_s(&fp, "dosya.txt", "w");
printf("Karakter giriniz...");
while ((ch = getchar()) != EOF) { // CTRL-Z EOF karakteri yaratmaktadır.
putc(ch, fp);
}
fclose(fp);
int rtrn2 = fopen_s(&fp2, "dosya.txt", "r");

while ((ch = getc(fp2)) != EOF) printf("%c", ch);

fclose(fp);
return 0;
}
```

Örnek 4: Satır Satır Okuma

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

int main(void)
{
FILE* stream;
char buffer[255];
int sonuc = fopen_s(&stream,"dosya.txt", "r");
if (sonuc != 0)
return 0;
while (fgets(buffer, 255, (FILE*)stream)) {
printf("%s\n", buffer);
}
fclose(stream);
return 1;
}
```

Örnek 4: Satır Satır Yazma

```
#include <stdio.h>

int main()
{
FILE* fp;
int i;
fopen_s(&fp, "dosya.txt", "w");

for (i = 0; i < 10; i++) {
fprintf(fp, "Satir %d\n", i + 1);
}

fclose(fp);
return 0;
}
```