Algoritmalar ve Programlama I – BLM 103

Hafta 3: C'e Genel Bakış, Değişkenler ve Operatörler

Fenerbahçe Üniversitesi

3. Hafta İçeriği

- C Dili ve Günümüz Kullanım Alanları
- Main Fonksiyonu
- Yorum Satırları
- Veri Tipleri
- Değişkenler
- Giriş ve Çıkış Kavramları
- Operatörler
- Yığınlar

C Dili

- Günümüzde aktif olarak gömülü sistem tasarımlarında kullanılmaktadır.
- Günümüz yüksek seviyeli dillerine (Python, C# …) göre daha donanım'a yakın bir dildir.
- Bu nedenle bir çok mikrokontrolör C veya çok benzeri bir dil ile programlanmaktadır.
- C dilinde donanım üzerinde hakimiyet diğer programlama dillerine göre daha yüksektir.
- Donanımdan soyut bir şekilde tasarım imkanı sağlar
  - Operasyonlar doğrudan komut setine yönelik değildir

İlk C Kodu

```
#include <stdio.h>

int main()
{
    /* İlk C Kodu */
   printf("Hello World!");
   return 0;
}
```

İlk C Kodu

```
#include <stdio.h>

int main()
{
   /* İlk C Kodu */
   printf("Hello World!");
   return 0;
}
```

Çalıştırılacak Kod Parçacıkları

- Her C kodu `main()` isimli bir fonksiyon içerir.
- Bu fonksiyon uygulamanın çalışmaya başladığı yerdir.
- Çalışacak olan kod parçacığı, main fonksiyonunun sınırlarını gösteren süslü parantezin içerisindedir.

İlk C Kodu

```
#include <stdio.h>

int main()
{
   /* İlk C Kodu */
   printf("Hello World!");
   return 0;
}
```

Çalıştırılacak Kod Parçacıkları

- `#include <stdio.h>`
  - Programın derlenmesine stdio.h dosyasının içerisindeki kod parçacıkları da dahil edilir.

- stdio kütüphanesinde giriş ve çıkışlar için kullanılan fonksiyonların tanımlamaları mevcuttur. Örn. Printf

Yorum Satırları

- C kaynak kodunun içerisine yazılmaktadırlar.
- Derleme işlemine girmezler ve çalıştırılmazlar.
- Kod'a uzun bir aradan sonra tekrar bakıldığında kod'un nasıl tasarlandığı, nasıl çalıştığı unutulabilir.
- Kodun çalışma mantığını anlatan bir not yazılması, kod'a bakan başka bir programcının veya uzun bir aradan sonra tekrar kendi koduna bakan bir programcının, koda hakim olma sürecini oldukça hızlandırması bakımından faydalıdır.

Yorum Satırları

```
#include <stdio.h>

int main()
{
   /* İlk C Kodu */
   printf("Hello World!");
   return 0;
}
```

- Bir veya birden çok satır yorum satırı kullanılacaksa

```
/*
Yorum satırı 1
Yorum Satırı  2
*/
```

Şeklinde kullanılabilir.

Yada tek satır yorum satırı kullanılacaksa

```
// Yorum satırı
```

Şeklinde yorum satırı kullanılması mümkündür.

Veri Tipleri

C dilinde en sık kullanılan veri tipleri:

- `char:` 1 byte (8 bit)'lık alanda tek bir karakter saklar
- `int:` Tam sayılar
- `float:` Ondalıklı sayılar, tek hassasiyet
- `double:` Ondalıklı sayılar, çift hassasiyet

Veri Tipleri

VERİ TİPİ	BELLEK (BYTE)	ARALIK	FORMAT GÖSTERİMİ
short int	2	-32,768 / 32,767	%hd
unsigned short int	2	0 / 65,535	%hu
unsigned int	4	0 / 4,294,967,295	%u
int	4	-2,147,483,648 / 2,147,483,647	%d
long int	4	-2,147,483,648 / 2,147,483,647	%ld
unsigned long int	4	0 / 4,294,967,295	%lu
long long int	8	-(2^63) / (2^63)-1	%lld
unsigned long long int	8	0 / 18,446,744,073,709,551,615	%llu
signed char	1	-128 / 127	%c
unsigned char	1	0 / 255	%c
float	4	-3.4E+38 / +3.4E+38	%f
double	8	-1.7E+308 / +1.7E+308	%lf
long double	12	3.4E-4932 / 1.1E+4932	%Lf

Değişkenler

- Tamsayı (integer)
- `123  /* decimal */`
- `-123`
- `0x123 /* hexadecimal */`
- Ondalıklı sayılar (float)
- `6.023`
- `6.023e23  /* 6.023 x 10^23 */`
- `5E12  /* 5.0 x 10^12 */`
- Karakter (char)
- `'c'`
- `'\n'  /* yeni satır */`
- `'\xA' /* ASCII 10 (0xA) */`

Değişken Tanımlama

```
#include <stdio.h>

int main()
{
   int abc = 5;
   char test = 'A';
   float sayi = 3.15;
   double x = 1.111;
   return 0;
}
```

Değişken isimleri harf, rakam ve _ karakterlerinden oluşabilir. Örn. sayi_1 gibi…

Büyük/Küçük harf duyarlılığı vardır.

"Toplam" değişkeni "toplam" değişkeninden farklıdır.

Değişken ismi sayı ile başlayamaz.

En fazla 32 karakter kullanılabilir. 32 karakterden sonraki karakterler atılır.

Değişken İsimleri Örnekleri

- *Geçerli*
- ```
  i
  deneme
  degisken_adi
   test
  degisken_tanimlamasi_32_karakterden_buyuk
  degisken_tanimlamasi_32_karakterden_buyuk_test
  ```

Aynı Tanımlama

- *Geçersiz*
- ```
  10sayisi
  ikilik'tumleyen
  tamamlandi?
  double
  ```

C Dilinin Bir Tanımı

Girişler ve Çıkışlar

- C dilinde bir çok giriş ve çıkış fonksiyonları bulunmaktadır.
- Bunları kullanmak için `<stdio.h>` kütüphanesinin eklenmesi gerekmektedir.

- `printf("%d\n", abc);`
  - Ekrana abc değişkeninin içerisindeki değeri, tamsayı formatında yazar.
  - Sondaki \n, yeni satır karakterini temsil eder. Ekranda yeni bir satıra geçilir.

- `Scanf_s("%f", &sayi);`
  - Sayı isimli değişkenin içerisine floating point formatında giriş almaktadır.

```
#include <stdio.h>

int main()
{
   int abc = 5;
   char test = 'A';
   float sayi = 3.15;
   double x = 1.111;
   printf ("Tamsayi %d", abc);
   printf ("Karakter %c", test);
   printf ("float %f", sayi);
   printf ("double %lf", x);
   return 0;
}
```

Girişler ve Çıkışlar

- Birden çok değişkeni tek bir prinf fonksiyonu ile bastırmak mümkünüdür.
- `printf("%d %c\n", abc, test);`

- Diğer formatlar:
- `%d` tamsayılar (integer)
- `%x` hexadecimal
- `%c` ASCII karakter
- `%f` Ondalıklı (floating-point) sayılar

Girişler ve Çıkışlar

- Kullanıcıdan klavye üzerinden giriş almak için

- `scanf_s("%d", &abc);`
- Kullanıcıdan tam sayı formatında giriş alındıktan sonra abc değişkeninin içerisine yazmaktadır.

```
#include <stdio.h>

int main()
{
    int abc;
    scanf_s("%d", &abc);
    return 0;
}
```

Girişler ve Çıkışlar

- Birden çok girişi aynı scanf fonksiyonu ile almak mümkündür

- `scanf("%d %d", &abc, &test);`

- Boşluk karakteri ile ayrılmış iki tamsayıyı kullanıcıdan almaktadır.

```
#include <stdio.h>

int main()
{
    int abc , test;
    scanf_s("%d %d", &abc, &test);
    return 0;
}
```

Girişler ve Çıkışlar

- Kaynak kod:

```
printf("%d sayisi bir asal sayidir.\n", 43);
printf("43 + 59 sonucu %d.\n", 43+59);
printf("43 artı 59 sonucu %x (hex).\n", 43+59);
printf("43 + 59 sonucu %c (karakter).\n", 43+59);
```

- Çıkış:

```
43 sayisi bir asal sayidir.
43 + 59 sonucu 102.
43 artı 59 sonucu 66 (hex).
43 + 59 sonucu f (karakter).
```

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a

Global ve Local Kapsamlar

- Değişkenler nereden erişilebilirdir?
- Global: Programın her yerinden erişilebilirdir.
- Local: Sadece belirli bir bölgeden erişilebilir.

- Lokal bir değişken, sadece tanımlandığı bloktan kullanılabilir.
  - Lokal bir alan yaratmak için süslü parantezler kullanılır { }

- Global değişkenler ana fonksiyonun dışında tanımlanırlar.

Global ve Lokal Örneği

```
#include <stdio.h>
int globalDegisken = 0;

int main()
{
  int lokalDegisken = 1;   /* main fonksiyonu için lokal */
  printf("Global %d Local %d\n", globalDegisken, lokalDegisken);
  {
    int lokalDegisken = 2;   /* bu blok için lokal */
    globalDegisken = 4;
    printf("Global %d Local %d\n", globalDegisken, lokalDegisken);
  }
  printf("Global %d Local %d\n", globalDegisken, lokalDegisken);
  return 1;
}
```

Çıktı

```
Global 0 Local 1
Global 4 Local 2
Global 4 Local 1
```

Operatörler

- Operatörler hakkında bilinmesi gereken üç özellik

(1) Fonksiyon

- Operatörlerin ne iş yaptığının bilinmesi

(2) Öncelikler

- Çarpmanın toplamaya olan önceliği gibi
  "a * b + c * d" -> "(a * b) + (c * d)" şeklinde yazılabilir.

(3) Birleşim

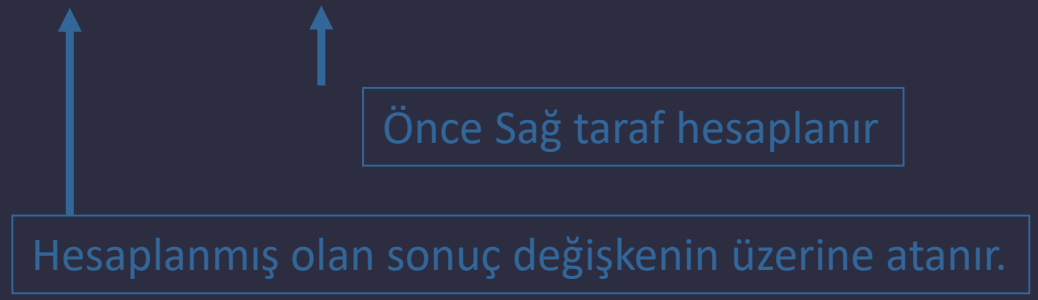
- "a - b - c" -> "(a - b) - c"
  Soldan sağa doğru işlemler gerçekleştirilmektedir.

Atama Operatörü

- Değişkenin içindeki değeri değiştirir.

-

```
x = x + 4;
```



Atama Operatörü

- Atamalar sağdan sola doğru olmaktadır.

- ```
  y = x = 3;
  ```

- Öncelikle x değişkenine 3 atanır, artından y değişkenine 3 atanmaktadır.
  ```

Aritmetik Operasyonlar

Sembol	Operasyon	Kullanım
*	çarpma	`x * y`
/	bölme	`x / y`
%	mod	`x % y`
+	toplama	`x + y`
–	çıkarma	`x - y`

Aritmetik Operasyonlar

Öncelikler	Operator	Açıklama
1	++ --	Sayının değerini 1 arttırma veya azaltma
	! ~	Mantık Değil ve Bitlerin Değilini Alma
	(type)	Bir değişken türünü, diğer bir değişkene dönüştürmek için kullanılır
	&	Bir değişkenin bellekteki adresini gösterir
	sizeof	Bir değişkenin bellekte kapladığı büyüklüğü döndürür
3	* / %	Çarpma, bölme ve mod
4	+ -	Toplama ve Çıkarma
5	<< >>	Sola veya sağa kaydırma
6	< <=	Küçüktür veya küçük eşittir
	> >=	Büyüktür veya büyük eşiittir
7	== !=	Eşittir veya eşit değildir
8	&	Bitlerin Ve (And) işlemi
9	^	Bitlerin XOR işlemi
10	|	Bitlerin Veya (OR) işlemi
11	&&	Mantık Ve (And)
12	||	Mantık (OR)
14	=	Atama İşlemi
	+= -=	Atamanın değerinin üzerine, ekleme veya çıkartma

Aritmetik Operasyonlar

- Farklı türlerde işlemler gerçekleştiriliyorsa (örn int ve float), işlem geniş aralığa sahip olan türe göre yapılır (Yani X eğer başlangıçta int ise, float'a dönüştürülerek, sonuç float hesaplanır).
- `x + 4.3`

- Tamsayı bölmesi – küsürat atılır.
- `x / 3`

  Örneğin x tamsayı ve 5 değerine sahi olsun, sonuç 1 hesaplanır (1.666666... değil)

- Mod işlemi.
- `x % 3`
- x 5'e eşit olsun, sonuç 2'dir.

Bit İşlemleri

Sembol	Operasyon	Kullanım
~	Bitleri Değil	`~x`
<<	sola kaydır	`x << y`
>>	sağa kaydır	`x >> y`
&	bitleri and'le	`x & y`
^	bitleri xor'la	`x ^ y`
|	bitleri or'la	`x | y`

Mantık Operasyonları

Sembol	Operasyon	Kullanım
!	Mantık Değil	`!x`
&&	Mantık Ve	`x && y`
||	Mantık Veya	`x || y`

- Değer 0 değilse Doğru, 0 ise yanlış olarak çalışır.

Kontrol Operatörleri

Sembol	Operasyon	Kullanım
• >	büyüktür	`x > y`
• >=	büyük eşittir	`x >= y`
• <	küçüktür	`x < y`
• <=	küçük eşittir	`x <= y`
• ==	eşittir	`x == y`
• !=	eşit değildir	`x != y`

Sonuç 1 ise Doğru, 0 ise Yanlıştır.

Özel Operatörler: ++ ve --

- Değişkenin değerini bir artırım veya azaltım yapan operatördür.

Sembol	Operasyon	Kullanım
++	önce arttırım	++x
--	önce azaltım	--x
++	sonra arttırım	x++
--	sonra azaltım	x--

- Önce: Önce değişkenin değerini arttırır ve sonra kullanılır.
- Sonra: Önce değişken kullanılır, sonra artırım gerçekleştirilir.

++ ve -- Kullanımı

- `x = 4;`
- `y = x++;`
- Sonuç: x = 5, y = 4
  (Çünkü x, atamadan sonra arttırıldı)

- `x = 4;`
- `y = ++x;`
- Sonuç: x = 5, y = 5
  (Çünkü x, atamadan önce arttırıldı)

Öncelikler

- Örn. a=1, b=2, c=3, d=4.

```
x = a * b + c * d / 2;   /* x = 8 */
```

- Aynı ifade:

```
x = (a * b) + ((c * d) / 2);
```

- Kullanılacak ifadeler uzadıkça, parantez kullanımı kodun okunabilirliğinin arttırılması için faydalı olmaktadır.

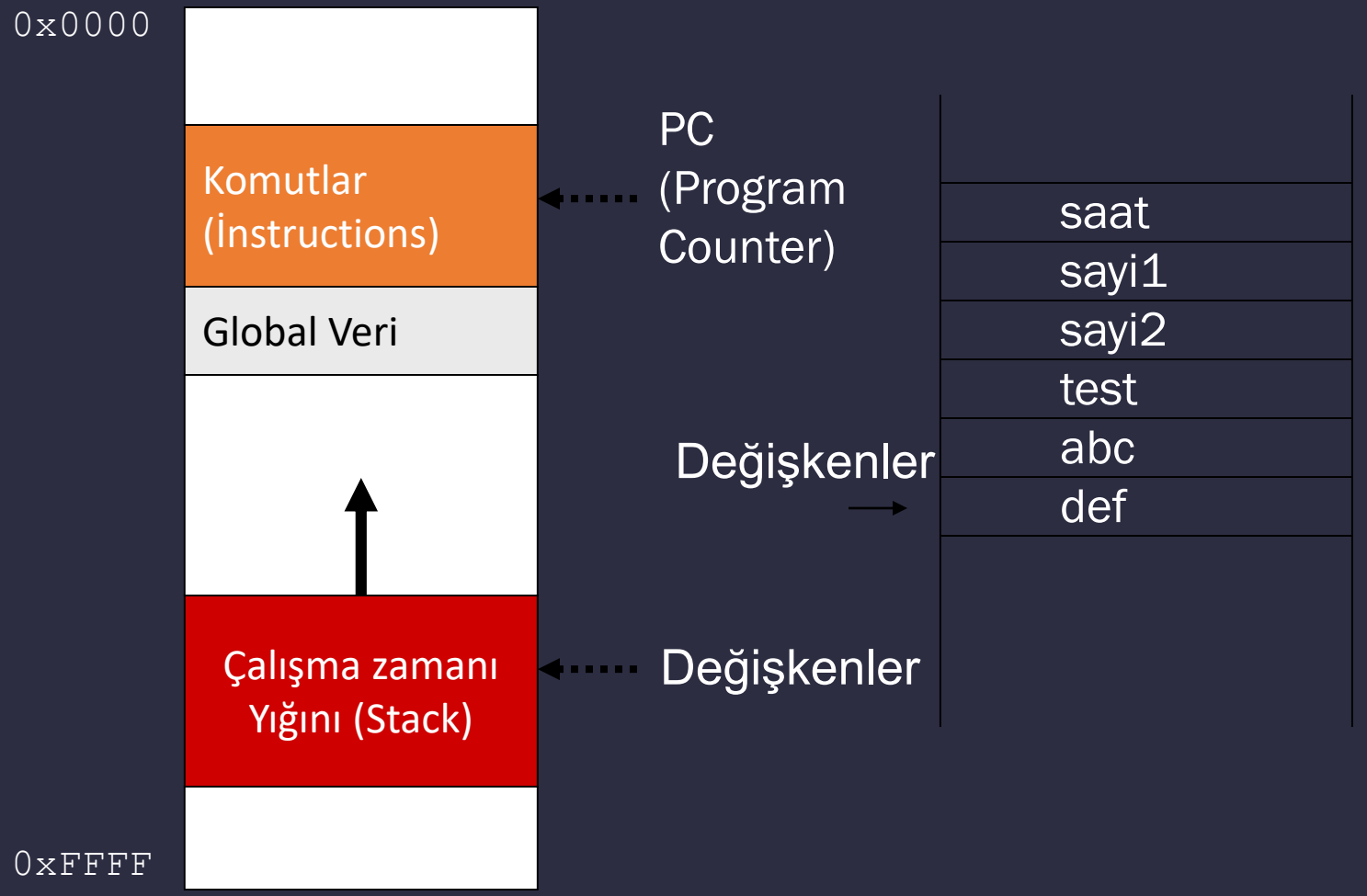
Sembol Tablosu

- Derleyici değişkenleri bellekte tutmaktadır. Tutulan değişkene ait bilgiler:
- Değişken ismi
- Tipi (int, float)
- Bellekteki yeri
- Geçerlilik alanı

İsim	Tür	Offset	Yer
saat	int	0	main
sayi1	int	-3	main
sayi2	int	-4	main
test	int	-1	main
abc	int	-5	main
def	int	-2	main

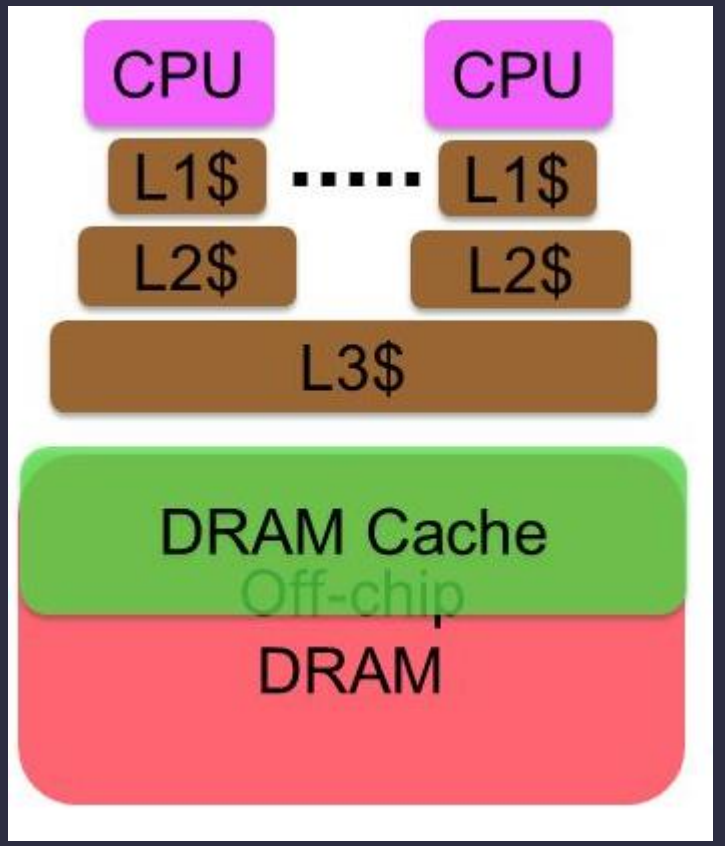
Değişkenler için Bellekte Yer Ayrılması

- Global Veriler
  - Tüm global tanımlanmış verilerin tutulduğu yerdir.
- Çalışma Zamanı Yığını (Stack)
  - Lokal olarak tanımlanmış ve kullanıldıktan sonra kaldırılacak olan verilerin tutulduğu alandır.



Değişkenler ve Bellek Lokasyonları

- Bir değişken DRAM üzerinde bir adreste tutuluyor.
- Ancak derleyiciler çok kullanılan bir değişkeni sürekli DRAM üzerinden alıp kullanmak yerine Cache denen, CPU içindeki küçük belleklerde/saklayıcılarda tutar.
- Bu saklayıcılar, CPU'nun içerisinde olduğu ve hızlı çalıştıkları için, DRAM'den alınacak bir veriye göre çok daha düşük bir zamanda kullanılabilirler.



DRAM – CPU Bağlantısı