Programlamaya Giriş

Veri Türleri ve Değişkenler

Hüseyin Ahmetoğlu

Yorumlar

- Yorumlar, kodlarınızı ve program mantığınızı belgelemek ve açıklamak için kullanılır. Yorumlar programlama beyanları değildir ve derleyici tarafından göz ardı edilir, ancak başkalarının programınızı anlaması için (ve ayrıca üç gün sonra kendiniz için) belgeler ve açıklamalar sağlamak için ÇOK ÖNEMLİDİR.
- C'de iki tür yorum vardır:
  - Çok Satırlı Yorum: / * ile başlar ve * / ile biter ve birkaç satıra yayılabilir.
  - Satır Sonu Açıklaması: // ile başlar ve mevcut satırın sonuna kadar sürer.
- Kodlarınızı açıklamak ve belgelemek için yorumları özgürce kullanmalısınız. Program geliştirme sırasında, bir yığın ifadeyi kalıcı olarak silmek yerine, gerekirse daha sonra geri alabilmeniz için bu ifadeleri yorumlayabilirsiniz.

İfadeler

- **İfade**: Bir programlama ifadesi, bir programdaki en küçük bağımsız birimdir, tıpkı bir cümle gibi. Bir parça programlama eylemi gerçekleştirir. Tıpkı bir cümlenin nokta ile bitmesi gibi, bir programlama ifadesi noktalı virgülle (;) sonlandırılmalıdır. (Neden bir cümle gibi bir nokta ile bitmiyor? Bunun nedeni, noktanın ondalık noktayla çökmesidir - bilgisayar için nokta ile ondalık noktayı ayırt etmek zordur!)

```
// Each of the following lines is a programming statement, which ends with a semi-colon (;)
int number1 = 10;
int number2, number3 = 99;
int product;
product = number1 * number2 * number3;
printf("Hello\n");
```

Bloklar

- **Blok**: Bir blok (veya bir bileşik ifade), köşeli ayraçlar {} ile çevrili bir ifadeler grubudur. Bloğun içindeki tüm ifadeler bir birim olarak ele alınır. Bloklar, fonksiyon, if-else ve döngü gibi yapılarda gövde olarak kullanılır; bunlar birden çok ifade içerebilir, ancak tek bir birim olarak kabul edilir. Karmaşık bir ifadeyi bitirmek için kapanış parantezinden sonra noktalı virgül koymaya gerek yoktur. Boş bloğa (herhangi bir ifade olmadan) izin verilir.

```
if (mark >= 50) {
   printf("PASS\n");
   printf("Well Done!\n");
   printf("Keep it Up!\n");
}

if (number == 88) {
   printf("Got it\n");
} else {
   printf("Try Again\n");
}

i = 1;
while (i < 8) {
   printf("%d\n", i);
   ++i;
}

int main() {
   ...statements...
}
```

Beyaz Boşluklar

- **Beyaz Boşluklar:** Boşluk, sekme ve yeni satır toplu olarak beyaz boşluklar olarak adlandırılır. C, fazladan beyaz boşlukları yok sayar. Yani, birden çok bitişik beyaz boşluk, tek bir beyaz boşluk olarak değerlendirilir.

```
int sum=0;          // Need a white space between int and sum
double average;     // Need a white space between double and average
average=sum/100.0;
```

```
// same as above
int  sum
   =  0 ;

   double  average  ;
average = sum / 100.0;
```

Kaynak Kodlarını Biçimlendirme

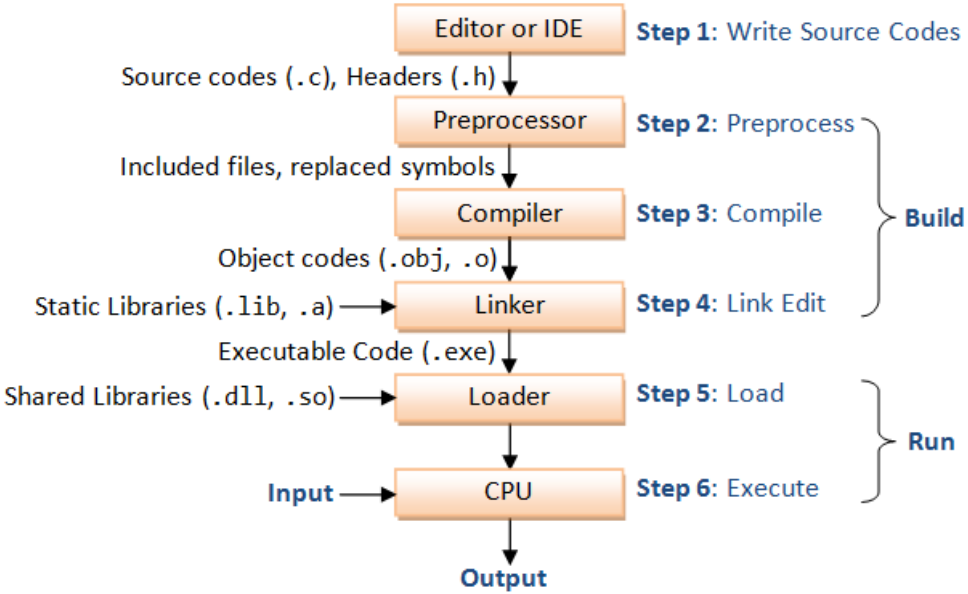
- **Kaynak Kodlarını Biçimlendirme:** Belirtildiği gibi, fazladan beyaz boşluklar yok sayılır ve hesaplama açısından önemi yoktur. Bununla birlikte, uygun girinti (sekmeler ve boşluklarla) ve fazladan boş satırlar, programın okunabilirliğini büyük ölçüde artırır, bu da başkaları için (ve üç gün sonra sizin programlarınızı anlamak için son derece önemlidir).

```
/*
 * Recommended Programming style.
 */
#include <stdio.h>
                              // blank line to separate sections of codes
int main() {  // Place the beginning brace at the end of the current line
   // Indent the body by an extra 3 or 4 spaces for each level

   int mark = 70;
   if (mark >= 50) {          // in level-1 block, indent once
      printf("You Pass!\n");  // in level-2 block, indent twice
   } else {
      printf("You Fail!\n");
   }

   return 0;
}   // ending brace aligned with the start of the statement
```

```
#include <stdio.h>
int main(){printf("Hello, world!\n");return 0;}
```

Ön İşlemci Direktifleri

- C kaynak kodu, nesne koduna derlenmeden önce işlenir.



Ön İşlemci Direktifleri

- Bir # işaretiyle (#include, #define gibi) başlayan bir önişlemci yönergesi, önişlemciye kaynak kodu nesne koduna derlemeden önce belirli bir eylemi (bir başlık dosyası dahil etme veya metin değiştirme gibi) gerçekleştirmesini söyler. . Önişlemci yönergeleri programlama deyimleri değildir ve bu nedenle noktalı virgülle sonlandırılmamalıdır. Örneğin,

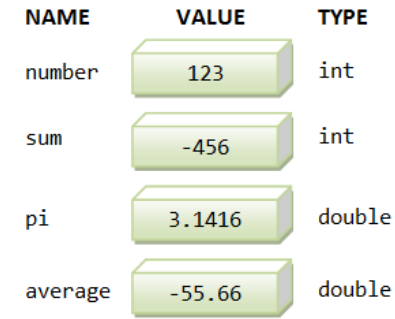
```
#include <stdio.h>      // To include the IO library header
#include <math.h>       // To include the Math library header
#define PI 3.14159265   // To substitute the term PI with 3.14159265 in this file
   // DO NOT terminate preprocessor directive with a semi-colon
```

Değişkenler

- Bilgisayar programları verileri işler. Bir değişken, işlenmek üzere bir veri parçasını depolamak için kullanılır. Depolanan değeri değiştirebileceğiniz için değişken olarak adlandırılır.
- Daha kesin olarak, bir değişken, belirli bir veri türünün değerini depolayan adlandırılmış bir depolama konumudur. Diğer bir deyişle, bir değişkenin bir adı, bir türü vardır ve bir değeri depolar.
- Bir değişkenin bir adı (veya tanımlayıcısı) vardır, örneğin yarıçap, alan, yaş, yükseklik. Ad, değişkene bir değer atamak (örneğin yarıçap = 1,2) ve depolanan değeri (ör. Alan = yarıçap * yarıçap * 3,1416) almak için her değişkeni benzersiz şekilde tanımlamak için gereklidir.

Değişkenler

- Bir değişken, belirli türden bir değeri depolayabilir. Çoğu programlama dilinde bir değişkenin bir türle ilişkilendirildiğini ve yalnızca belirli bir türün değerini depolayabileceğini not etmek önemlidir. Örneğin, bir int değişkeni 123 gibi bir tamsayı değeri depolayabilir, ancak 12.34 gibi gerçek bir sayı veya "Merhaba" gibi metin depolayamaz.
- Tür kavramı, 0'lar ve 1'lerden oluşan verilerin yorumlanmasını basitleştirmek için erken programlama dillerinde tanıtıldı. Tür, verilerin boyutunu ve düzenini, değerlerinin aralığını ve uygulanabilecek işlem kümesini belirler.



A variable has a ***name***, stores a ***value*** of the declared ***type***

Tanımlayıcılar

- Bir değişkeni (veya bir fonksiyon veya sınıf gibi başka bir varlığı) adlandırmak için bir tanımlayıcı gereklidir. C, tanımlayıcılara aşağıdaki kuralları uygular:
  - Bir tanımlayıcı, belirli bir uzunluğa kadar (derleyiciye bağlı, tipik olarak 255 karakter), büyük ve küçük harfler (a-z, A-Z), rakamlar (0-9) ve alt çizgi "_" içeren bir karakter dizisidir.
  - Beyaz boşluk (boşluk, sekme, yeni satır) ve diğer özel karakterlere (+, -, *, /, @, &, virgül vb.) İzin verilmez.
  - Bir tanımlayıcı bir harf veya alt çizgiyle başlamalıdır. Bir rakamla başlayamaz. Alt çizgi ile başlayan tanımlayıcılar genellikle sistem kullanımı için ayrılmıştır.
  - Bir tanımlayıcı, ayrılmış bir anahtar kelime veya ayrılmış bir hazır bilgi (ör. İnt, double, if, else, for) olamaz.
  - Tanımlayıcılar büyük / küçük harfe duyarlıdır. book, Book ve BOOK farklı değişkenlerdir.

Tanımlayıcılar

- Değişken adı, bir isim veya birkaç kelimeden oluşan bir isim tümcedir. İlk kelime küçük harfle yazılırken, geri kalan kelimeler arasında boşluk bırakılmadan ilk harf büyük yazılır. Örneğin, thefontSize, roomNumber, xMax, yMin, xTopLeft ve thisIsAVeryLongVariableName.
- **Öneriler**
- Kendini tanımlayan ve değişkenin anlamını yakından yansıtan bir ad seçmek önemlidir,
- A, b, c, d, i, j, k, i1, j99 gibi anlamsız isimler kullanmayın.
- Koordinatlar için x, y, z, dizin için i gibi yaygın isimler olmadıkça, yazması daha kolay ancak çoğu zaman anlamsız olan tek alfabe adlarından kaçının.

Değişken Bildirimi

Syntax	Example

// Declare a variable of a specified type
type identifier;
// Declare multiple variables of the same type, separated by commas
type identifier-1, *identifier-2*, ..., *identifier-n*;
// Declare a variable and assign an initial value
type identifier = *value*;
// Declare multiple variables with initial values
type identifier-1 = *value-1*, ..., *identifier-n* = *value-n*;
	int option;
	
	double sum, difference, product, quotient;
	
	int magicNumber = 88;
	
	double sum = 0.0, product = 1.0;

```
int mark1;            // Declare an int variable called mark1
mark1 = 76;           // Use mark1
int mark2;            // Declare int variable mark2
mark2 = mark1 + 10;   // Use mark2 and mark1
double average;       // Declare double variable average
average = (mark1 + mark2) / 2.0;   // Use average, mark1 and mark2
int mark1;            // Error: Declare twice
mark2 = "Hello";      // Error: Assign value of a different type
```

Değişken Bildirimi

- C'de, kullanılmadan önce bir değişkenin adını bildirmeniz gerekir.
- C, "güçlü tip" bir dildir. Bir değişken bir türü alır. Bir değişkenin türü bildirildikten sonra, yalnızca bu belirli türe ait bir değeri depolayabilir. Örneğin, bir int değişkeni yalnızca 123 gibi bir tamsayı tutabilir ve -2.17 gibi kayan noktalı sayı veya "Merhaba" gibi metin dizesi saklanamaz.
- Tür kavramı, 0'lar ve 1'lerden oluşan verilerin yorumlanmasını basitleştirmek için erken programlama dillerinde tanıtıldı. Bir veri parçasının türünü bilmek, yorumlanmasını ve işlenmesini büyük ölçüde basitleştirir.
- Her değişken yalnızca bir kez bildirilebilir.
- C'de, kullanılmadan önce bildirildiği sürece, programın herhangi bir yerinde bir değişken tanımlayabilirsiniz. (C99'dan önceki C'de, tüm değişkenler işlevlerin başında bildirilmelidir.)
- Bir değişkeni ilk kullanımdan hemen önce bildirmeniz önerilir.
- Bir değişkenin türü program içinde değiştirilemez.

Değişken Bildirimi

- Bir değişken bildirildiğinde, siz bir başlangıç değeri atayana kadar değersizdir ama bu boş olduğu anlamına gelmez. Bir değişkeni başlatmadan önce kullanırsanız, C'nin herhangi bir uyarı / hata vermediğini unutmamak önemlidir - bu kesinlikle bazı beklenmedik sonuçlara yol açar.

```c
#include <stdio.h>

int main() {
   int number;              // Declared but not initialized
   printf("%d\n", number);  // Used before initialized
                            // No warning/error, BUT unexpected result
   return 0;
}
```

Sabitler

- Sabitler, değiştirilemez değişkenlerdir ve const anahtar sözcüğü ile bildirilir. Program yürütülürken değerleri değiştirilemez. Ayrıca const, bildirim sırasında başlatılmalıdır.

```
const double PI = 3.1415926;  // Need to initialize
```

- Sabit Adlandırma Kuralı: Alt çizgi ile birleştirilmiş büyük harfli kelimeler kullanın. Örneğin, MIN_VALUE, MAX_SIZE.

Atama

- Bir atama bildirimi:

  (RHS) bir değişkene (LHS) birebir değeri atar; veya
  (RHS) bir ifadeyi değerlendirir ve ortaya çıkan değeri bir değişkene (LHS) atar.

  RHS bir değer olacaktır; ve LHS bir değişken (veya bellek adresi) olacaktır.

Syntax	Example

`// Assign the literal value (of the RHS) to the variable (of the LHS)`
`variable = literal-value;`
`// Evaluate the expression (RHS) and assign the result to the variable (LHS)`
`variable = expression;`
	`number = 88;`
	
	`sum = sum + number;`

- Atama ifadesi şu şekilde yorumlanmalıdır: Sağ taraftaki (RHS) ifade ilk olarak bir sonuç değeri (rvalue veya right-value olarak adlandırılır) üretmek için değerlendirilir. Daha sonra r değeri, sol taraftaki (LHS) değişkene (veya bir r değerini tutabilen bir konum olan lvalue) atanır. Elde edilen değeri LHS'ye atamadan önce ilk olarak RHS'yi değerlendirmeniz gerektiğini unutmayın.
- "=" Sembolü, atama operatörü olarak bilinir. Programlamada "=" nin anlamı matematikten farklıdır. Eşitlik yerine atamayı ifade eder. RHS gerçek bir değerdir; veya bir değer olarak değerlendirilen bir ifade; LHS ise değişken olmalıdır. x = x + 1'in programlamada geçerli olduğunu (ve sıklıkla kullanıldığını) unutmayın. x + 1'i değerlendirir ve sonuçtaki değeri x değişkenine atar. x = x + 1 Matematikte bu imkansızdır. Matematikte x + y = 1'e izin verilirken, programlamada geçersizdir (çünkü bir atama ifadesinin LHS'si bir değişken olmalıdır). Bazı programlama dilleri, eşitlikle karışıklığı önlemek için atama operatörü olarak ": =", "←", "->" veya "→" sembolünü kullanır.

```
number = 8;            // Assign literal value of 8 to the variable number
number = number + 1;   // Evaluate the expression of number + 1,
                       //  and assign the resultant value back to the variable number
```

Temel Türler

- **Integers**: C, şu tam sayı türlerini destekler: artan bir boyut düzeninde char, short, int, long, ve long long, (C11'de). Gerçek boyut, uygulamaya bağlıdır. Tam sayılar (char hariç) işaretli sayılardır (sıfır, pozitif ve negatif sayıları tutabilir). İşaretsiz tamsayılar (sıfır ve pozitif sayılar tutabilen) bildirmek için unsigned [char | short | int | long | long | long] anahtar sözcüğünü kullanabilirsiniz. Char | short | int | long | long | ile signed|unsigned birleştirilmiş toplam 10 tam sayı türü vardır.

Temel Türler

- **Characters**: Karakterler (ör. 'A', 'Z', '0', '9') ASCII'de tam sayı olarak kodlanır ve karakter türünde tutulur. Örneğin, karakter '0' 48 (onlu) veya 30H (onaltılık); 'A' karakteri 65 (onlu) veya 41H (onaltılık); 'a' karakteri 97 (ondalık) veya 61H (onaltılık).
- Char türünün ASCII kodundaki karakter veya 8 bitlik bir tamsayı olarak yorumlanabileceğini unutmayın. İşaretli int veya long aksine, uygulamaya bağlı olarak char işaretli veya işaretsiz olabilir. İşaretli veya işaretsiz karakterleri açıkça belirtmek için İşaretli karakter veya işaretsiz karakter kullanabilirsiniz.

Kayan noktalı sayıları:

- 3 kayan nokta türü vardır: float, double ve long double. float ve double, IEEE 754 standardında belirtildiği gibi temsil edilir. Bir float, yaklaşık olarak ± 1.40239846 × 10 ^ -45 ve ± 3.40282347 × 10 ^ 38 arasında bir sayıyı temsil edebilir. Bir double, yaklaşık olarak ± 4,9406564584124654 4 × 10 ^ -324 ve ± 1,7976931348623157 0 × 10 ^ 308 arasında bir sayıyı temsil edebilir. Unutmayın ki tüm gerçek sayılar float ve double ile temsil edilemez çünkü sonsuz gerçek sayılar vardır. Değerlerin çoğu yaklaşıktır.

Integers

TİP	TANIM	BYTE	MİN	MAX
int (or signed int)	Signed integer (of at least 16 bits)	4 (2)	-2147483648	2147483647
unsigned int	Unsigned integer (of at least 16 bits)	4 (2)	0	4294967295
char	Character (can be either signed or unsigned depends on implementation)	1		
signed char	Character or signed tiny integer (guarantee to be signed)	1	-128	127
unsigned char	Character or unsigned tiny integer (guarantee to be unsigned)	1	0	255
short (or short int) (or signed short) (or signed short int)	Short signed integer (of at least 16 bits)	2	-32768	32767
unsigned short (or unsigned shot int)	Unsigned short integer (of at least 16 bits)	2	0	65535

Integers devamı…

TİP	TANIM	BYTE	MİN	MAX
long (or long int) (or signed long) (or signed long int)	Long signed integer (of at least 32 bits)	4 (8)	-2147483648	2147483647
unsigned long (or unsigned long int)	Unsigned long integer (of at least 32 bits)	4 (8)	0	same as above
long long (or long long int) (or signed long long) (or signed long long int)	Very long signed integer (of at least 64 bits)	8	-2^{63}	$2^{63}-1$
unsigned long long (or unsigned long long int)	Unsigned very long integer (of at least 64 bits)	8	0	$2^{64}-1$

Kayan noktalı sayıları:

TİP	TANIM	BYTE	MİN	MAX
`float`	Floating-point number, ≈7 digits (IEEE 754 single-precision floating point format)	4	3.4e38	3.4e-38
`double`	Double precision floating-point number, ≈15 digits (IEEE 754 double-precision floating point format)	8	1.7e308	1.7e-308
`long double`	Long double precision floating-point number, ≈19 digits (IEEE 754 quadruple-precision floating point format)	12 (8)		
`wchar_t`	Wide (double-byte) character	2 (4)		

KAYNAKLAR

- Goel, A., & Mittal, A. (2016). Computer Fundamentals and Programming in C (RMK). Pearson Education India. Retrieved from https://www.oreilly.com/library/view/computer-fundamentals-and/9789332579200
- yet another insignificant Programming Notes. (2021, April 08). Retrieved from https://www3.ntu.edu.sg/home/ehchua/programming/index.html#Cpp