DERS 03 C++ VERİ TİPLERİ

Veri Tipleri

Değişken İsimlendirmedeki Kurallar

Değişkenler

Sabitler

`char`

`string`

Şekillendirme kodları

Mantıksal Değişkenler

NOT

C ve C++ programlama dilleri bazı ortak yazım kuralları olmasına rağmen, birbirlerinden oldukça farklıdırlar.

VERİ TİPLERİ

Veri tipi, bir tür verinin bilgisayarın hafızasında (RAM) nasıl depolanacağını belirler.

C++ 6 temel veri tipi barındırır:

```
char
int
float
double
bool
wchar_t
```

Ayrıca sayısal veri tiplerinin önüne koyulan ve bileşik veriler türetilebilen nitelendiriciler vardır. :

`short, long, signed, unsigned`

Örnek:

```
short int
unsigned char
```

Aşağıdaki tabloda önceki sayfada verilen temel veri tiplerinin alabileceği en büyük ve en küçük değerler görülmektedir.

C++ da tamsayı (int) verisi	Alabilecekleri en küçük ve en büyük değerleri
short	-32767 ... 32767
unsigned short	0 ... 65535
int	**-2 147 483 647 ... 2 147 483 647**
unsigned	0 ... 4 294 967 295
long	-2 147 483 647 ... 2 147 483 647
unsigned long	0 ... 4 294 967 295

C++ da ondalıklı sayılar	En küçük ve en büyük değerleri	Noktadan sonraki rakam sayısı
float	10^{-37} ... 10^{+38}	6
double	**10^{-307} ... 10^{+308}**	**15**
long double	10^{-4931} ... 10^{+4932}	19

Kırmızı renkte verilen veri tipleri temel programlama için bütün ihtiyaçlarınızı karşılayacaktır.

NOTLAR:

1. Ön niteleyicisiz tanımlamalar **`char`**, **`short`**, **`int`**, (**`long int`**) otomatik olarak işaretlendirilir.
2. **`short`** ve **`long`** niteleyicilerinden sonra **`int`** yazmak gerekmez.

DEĞİŞKEN İSİMLENDİRMEDEKİ KURALLAR

Bir değişken isimlendirilirken harfler ve sayılar kullanılabilir. Bu durum sabitleri, fonksiyonları, yapıları ve sınıfları isimlendirme için de aynıdır.

Doğru tanımlanmış bir isim:

- harfle ya da alt çizgi ile (_) başlar,
- sadece harf (**a-z**, **A-Z**), rakam(**0-9**), ya da alt çizgi içerebilriler.
- ancak aşağıda verilmiş C++ komutlarıyla çakışan isimler verilmemelidir:

```
asm, auto, bool, break, case, catch, char, class, const,  const_cast,
continue, default, delete, do, double, dynamic_cast, else, enum, explicit,
export, extern, false, float, for, friend, goto, if, inline, int, long,
mutable, namespace, new, operator, private, protected, public, register,
reinterpret_cast, return, short, signed, sizeof, static, static_cast, struct,
switch, template, this, throw, true, try, typedef, typeid, typename, union,
unsigned, using, virtual, void, volatile, wchar_t, while
```

Bu kurallara göre **doğru** tanımlanmış bazı isimler aşağıda verilmiştir:

```
mass
peynir
pos12
speed_of_light
SpeedOfLight
isPrime
```

Aşağıda verilenler ise **hatalı** tanımlamalardır.

```
2ndBit
speed of light
yağmur
c++
float
```

Sadece İngilizce alfabesindeki harfler isimlerde kullanılmalıdır.
Türkçe harfler ve özel karakterler kullanılamaz.

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

C++ programlama dilinin büyük-küçük harf duyarlılığı vardır.

Buna göre, Büyük harf ve küçük harfle tanımlanmış aynı isimli değişkenler farklı değişkenler kabul edilir.

Örnek olarak, `Ekmek` ve `ekmek` farklı değişkenlerdir.

DEĞİŞKENLER

- Değişken; bir değere verilen sembolik isimdir ve bir değerin bilgisayar hafızasında nereye saklanacağı bilgisini taşır.
- Değişken; doğru tanımlanmış bir isim olmalıdır.
- Bir değişken kullanılmadan önce, tanımlanmış olması gerekir.
- Bir değişken tanımlanırken, nasıl bir değişken olduğu bilgisi bilgisayara verilmelidir; **`int`**, **`char`**, **`double`**, ...
- Bir değişken, bir program bölümünde limitli erişilebilir özelliği vardır. Yani sadece tanımlandığı yerde görülebilir ve kullanılabilir.

- Örnek Tanımlamalar

```
int i, j;
long k;
float w, x, y, z;
double hiz, YuzeyKuvveti;
```

- Bir değişkenin ilk değeri atanırken iki farklı şekilde atama yapılabilir:

```
int kek = 122;
```

- veya

```
int kek(122);
```

Örnek Program: *Değişkenlerin tanımlanması ve kullanılması.*

```
#include <iostream>
using namespace std;

int main () {

  short x = 22, y = 11, z;
  z = x - y;
  cout << "z = " << z << endl;

  int p = 3;
  int q = x*y*z - 2*p;
  cout << "q = " << q << endl;

  return 0;

}
```

Çıktı:

```
z = 11
q = 2656
```

Örnek Program: Değişkenlerin Erişilebilirliği

```
#include <iostream>
using namespace std;

int k = 11;          // k genel tanımlanmış bir değişkendir (tüm program
                     // tarafından görülür. Programın kullandığı alt
                     // fonksiyonlarda da görülür.)
int main ()
{

  int k = 22;        //buradaki k sadece main bloğu içinde görülür
  {
    int k = 33;      //buradaki k ise sadece bu blok içinde görülür
    cout << «Ic blok icinde: k = " << k << endl;
  }
  cout << "main() icinde:  k = " << k << endl;
  cout << "Genel k = " << ::k << endl;

}
```

Çıktı:

```
Ic blok icinde: k = 33
main() icinde:  k = 22
Genel k = 11
```

SABİTLER

- Programın güvenliğini artırmak için, değişkenler **`const`** ön ismi kullanılarak tanımlanabilir. Bu sayede program kodlanırken başta tanımlandıktan sonra değerleri değiştirilemez.

```
const double pi = 3.14159265358979;
```

Örneğin burada, **`pi`** ondalıklı sayı tipinde tanımlanmış ve 3.14159265358979 değerine atanmıştır.

(double tipinde tanımlanmış bir onadlıklı sayının noktadan sonra 15 rakam alacağını unutmayın).

`const` niteleyicisi bilgisayar bu değerin değiştirilemeyeceğini anlatır.

- Aynı iş *Sembolik* olarak **`#define`** komuyla da yapılabilir.

```
#define pi 3.14159265358979
```

TAMSAYI VE ONDALIKLI SAYILARIN GÖSTERİLMESİ

- Tamsayı değerleri üç farklı tabanda gösterilebilir: taban-10 (onluk), taban-8 (sekizli) ve taban-16 (hexadecimal)

```
i = 75;      taban-10 (varsayılan)
i = 0113;    75'in 8'li tabandaki gösterimi
i = 0x4B;    75'in 16'lı tabandaki gösterimi
i = 0x4b;    75'in 16'lı tabandaki gösterimi (Küçük b)
```

- Ondalıklı sayılar noktalı veya üslü olarak gösterilebilirler. **E** veya **e** sembolü üs olarak kullanılır.

```
x = 123.456;       Noktalı gösterim.
x = 1234.56e-1;    üslü (anlamı 1234.56 x 10^-1)
c = 1.6E-19;       üslü (anlamı 1.6 x 10^-19)
A = 6.02e23;       üslü (anlamı 6.02x10^23)
```

BAZI ASCII KARAKTERLERİ

Binary	Oct	Dec	Hex	Glyph
010 0000	040	32	20	sp
010 0001	041	33	21	!
010 0010	042	34	22	"
010 0011	043	35	23	#
010 0100	044	36	24	$
010 0101	045	37	25	%
010 0110	046	38	26	&
010 0111	047	39	27	'
010 1000	050	40	28	(
010 1001	051	41	29	)
010 1010	052	42	2A	*
010 1011	053	43	2B	+
010 1100	054	44	2C	,
010 1101	055	45	2D	-
010 1110	056	46	2E	.
010 1111	057	47	2F	/
011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4

Binary	Oct	Dec	Hex	Glyph
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	:
011 1011	073	59	3B	;
011 1100	074	60	3C	<
011 1101	075	61	3D	=
011 1110	076	62	3E	>
011 1111	077	63	3F	?
100 0000	100	64	40	@
100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	105	69	45	E
100 0110	106	70	46	F
100 0111	107	71	47	G
100 1000	110	72	48	H
100 1001	111	73	49	I

Binary	Oct	Dec	Hex	Glyph
100 1010	112	74	4A	J
100 1011	113	75	4B	K
100 1100	114	76	4C	L
100 1101	115	77	4D	M
100 1110	116	78	4E	N
100 1111	117	79	4F	O
101 0000	120	80	50	P
101 0001	121	81	51	Q
101 0010	122	82	52	R
101 0011	123	83	53	S
101 0100	124	84	54	T
101 0101	125	85	55	U
101 0110	126	86	56	V
101 0111	127	87	57	W
101 1000	130	88	58	X
101 1001	131	89	59	Y
101 1010	132	90	5A	Z
101 1011	133	91	5B	[
101 1100	134	92	5C	\
101 1101	135	93	5D	]
101 1110	136	94	5E	^

Binary	Oct	Dec	Hex	Glyph
101 1111	137	95	5F	_
110 0000	140	96	60	`
110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	c
110 0100	144	100	64	d
110 0101	145	101	65	e
110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l
110 1101	155	109	6D	m
110 1110	156	110	6E	n
110 1111	157	111	6F	o
111 0000	160	112	70	p
111 0001	161	113	71	q
111 0010	162	114	72	r
111 0011	163	115	73	s

Binary	Oct	Dec	Hex	Glyph
111 0100	164	116	74	t
111 0101	165	117	75	u
111 0110	166	118	76	v
111 0111	167	119	77	w
111 1000	170	120	78	x
111 1001	171	121	79	y
111 1010	172	122	7A	z
111 1011	173	123	7B	{
111 1100	174	124	7C	|
111 1101	175	125	7D	}
111 1110	176	126	7E	~

char

- C++ karakter tipindeki değişkenleri sayısal kodlarla gösterir.
- **char** tipinde tanımlanmış bir değişkene, sadece tek bir karakter atanabilir.
- Karakter tipinde tanımlanmış bir değişkene değer atarken, tek tırnak kullanılır.

```
'A'        'b'        ' '              ';'
```

- Karakterler tamsayı kodlarıyla gösterildiğinden, C++ bunların `char` verisinden `int` verisine çevrilmesine ya da tam tersine izin verir.

- Örneğin, aşağıdaki tanımlamada soru işareti için kullanılan tamsayı kodu listelenmiştir :

```
int  SoruIsareti= '?';
cout << " ? isaretinin kodu = " << SoruIsareti << endl;
```

string

- C++ dilinin standart kütüphanelerinden biri de karakterlerin gruplanabilmesini sağlayan **string** veri tipidir.
- Ancak bu veri tipini kullanmak için öncelikle kütüphane koda dahil edilmelidir:

```
#include <string>
```

- **string** veri tipleri aynı diğer veri tipleri gibi tanımlanır ve kullanılırlar.
- Ancak bu tip değişkenlere değer atamak için çift tırnak kullanılmalıdır.
- Örneğin:

```
string a,b;
a="Sample 1";
b="Sample 2";
cout<<"a="<<a<<endl;
cout<<"b="<<b<<endl;
```

ŞEKİLLENDİRME KODLARI

Karakter tipi değişkenlerle kullanılan bazı şekillendirme kodları ters kesme (\) ile kullanılırlar:

Şekil Kodu	Tanımı	Örnek
\a	alarm (sesli uyarı)	cout << "Error !\a";
\b	backspace	cout << "Ondokuz \bmayis";
\r	İmleci sütun 1'e gönderir	cout << "ondokuzmayis\rO";
\n	Yeni satır	cout << "Ondokuz\nmayis";
\t	Yatay tab	cout << x << '\t' << y;
\'	tek tırnak	cout << "Ondokuzmayis\' ta ";
\\	ters kesme	cout << "Ondokuzmayis\Samsun";

MANTIKSAL DEĞİŞKENLER

- C++ bool komutuyla mantıksal değişkenleri tanımlar. Bu değişkenler sadece `true` (doğru) veya `false` (yanlış) değerlerini alabilir.

- Değer atamaları karşılaştırma operatörleri kullanılarak da verilebilir.
- Örneğin `x > 100` ve `y < = 0`. Bunu seçim yapılarını konuşurken daha detaylı inceleyeceğiz.

```
int main(){
bool a,b;
a=true;
b=1>2;
cout<<"a="<<a<<endl;
cout<<"b="<<b<<endl;
```