BÖLÜM 5

VERİ TÜRLERİ VE ARİTMETİK İŞLEÇLER

ANAHTAR KAVRAMLAR

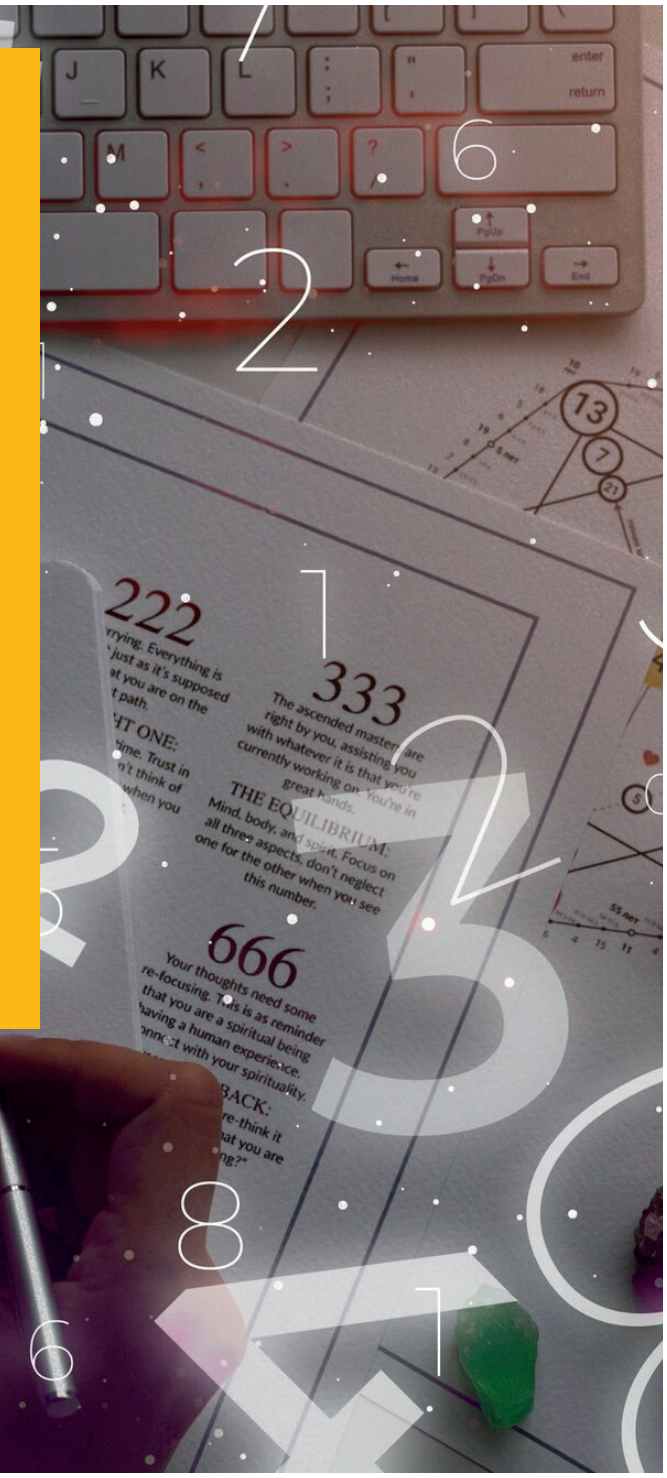
Tam Sayı, Kesirli Sayı, Karakter, Değişken, Atama İşleci, Aritmetik İşleçler, İşleç Önceliği

ÖĞRENME HEDEFLERİ

1. Veri türleri arasındaki farkları bilir
2. Değişken tanımlama ve isimlendirme kurallarını bilir
3. Atama işlecini nasıl kullanacağını bilir
4. Aritmetik işleçleri nasıl kullanacağını bilir
5. Aritmetik işleçlerin öncelik sıralamasını bilir

BEŞİNCİ BÖLÜM

GİRİŞ



Bir veri türü, bir türün alabileceği değer aralıkları ve üzerinde tanımlanmış işlemler ile tanımlanır. Bir bilgisayar programı sakladığı verileri işleyerek sonuçlar üretir. Bazı programlama dilleri bir değişken tanımlanırken veri türünü belirtmeyi şart koşarken bazıları daha esnek olmaları sebebiyle böyle bir şart koşmazlar. Veri türünün önceden bilinmesiyle o değişken için hafızada ne kadar alan gerektiği derleyici tarafından önceden hesap edilebilir. C, C++ ve Java gibi programlama dillerinde değişken oluştururken veri türünün belirtilmesi gerekir. Python ve Ruby gibi diller de böyle bir şart yoktur. Bu tür dillerin avantajı programcının veri türleriyle uğraşmasını ortadan kaldırır. Aynı isme sahip farklı veri türlerinde değişken oluşturmak mümkün hale gelmektedir. Böylece bellek kapasitesi daha verimli bir şekilde yönetilir.

1. TEMEL VERİ TÜRLERİ

Temel veri türleri diğer türler cinsinden tanımlanmayan veri türleri olarak tanımlanır. C++ programlama dilinde tanımlı üç adet temel veri tipi vardır:

- Sayısal
- Boolean
- Karakter

Tablo 5.1 C ++ Programla Dilinde Tanımlı Temel Bazı Veri Türleri ve Değer Aralıkları

Veri Türü	Boyut	Değer Aralığı
unsigned short	2 bayt	0 – 65.535
short	2 bayt	-32.768 – 32.767
unsigned long	4 bayt	0 – 4.294.967.295
long	4 bayt	-2.147.483.648 – 2.147.483.647
int	4 bayt	-2.147.483.648 – 2.147.483.647
unsigned int	4 bayt	0 - 4.294.967.295
char	1 bayt	256 karakter
float	4 bayt	1,2e-38 – 3,4e38
double	8 bayt	2,2e-308 – 1,8e308

Tablo 5.1'de C++ programlama dilinde tanımlı bazı veri türleri ve tutabilecekleri değer aralıkları verilmiştir. Tablodaki değerler kullanılan işletim sistemine ve derleyiciye bağlı olarak farklılıklar gösterebilmektedir. Bir değişkenin hafızada ne kadar alana gereksinim duyduğunu öğrenmek için "sizeof()" fonksiyonu kullanılır.

```
cout << sizeof(short)<<endl;
cout << sizeof(int)<<endl;
cout << sizeof(long)<<endl;
cout << sizeof(float)<<endl;
cout << sizeof(double)<<endl;
cout << sizeof(char)<<endl;
cout << sizeof(bool)<<endl;
```

DİKKAT EDELİM

Her bir veri türünün tutabileceği değer aralıkları belirlidir. Değişken tanımlaması yaparken saklanacak veriye uygun veri türleri seçilmelidir.

ARAŞTIRALIM

Tablo 5.1'de olmayan diğer temel veri türleri https:// en.cppreference.com/w/cpp/language/types adresinden incelenebilir.

2. SAYISAL VERİ TÜRLERİ

Bilimsel hesaplamalar genellikle sayısal veriler kullanılarak yapılır. Birçok bilim dalının ihtiyacını karşılamak için farklı sayısal veri türleri tanımlanmıştır. C++ programlama dilinde tanımlı sayısal veri türleri tam sayılar ve kesirli sayılar olmak üzere ikiye ayrılır.

2.1. Tam Sayı Veri Türleri

Tam sayı veri türlerinin kesirli parçaları yoktur. Bir sınıftaki öğrenci sayısı ve otoparktaki araç sayısı bu gruba girer. En genel haliyle "int" anahtar kelimesi kullanarak tam sayı veri türünden değişkenler tanımlanabilir. Kullanıcının ihtiyacına göre daha az alan gerektiren "short" veya daha çok alan gereksinimi olan "long" anahtar kelimeleri ile tam sayı veri türleri tanımlamak mümkündür. Bu anahtar kelimelerin önüne "unsigned" eklenmesiyle sadece pozitif değer aralığında tam sayılar elde edilebilir.

Örnek:

```
int fiyat = 50000;
short mesafe = 40;
long dunya_nufusu=8000000000;
cout<<"Fiyat="<<fiyat<<endl;
cout<<"Mesafe="<<mesafe<<endl;
cout<<"Dünya nüfusu="<<dunya_nufusu<<endl;
```

Çıktı:

```
Fiyat=50000
Mesafe=40
Dünya nüfusu=8000000000
```

2.2. Kesirli Sayı Veri Türleri

Kesirli sayı veri türleri ondalık değerlere sahip verileri temsil etmek için tanımlanmıştır. Bir kişinin ağırlık ve boy bilgisi bu grupta değerlendirilir. En genel haliyle "float" veya "double" anahtar kelimeleri kullanarak tanımlanır. Bir değişken "double" olarak tanımlanmışsa daha geniş değer aralığına sahip olmaktadır ama hafızada daha fazla alan gereksinimi olacaktır. Benzer şekilde önlerine "unsigned" kelimesi eklenerek pozitif değerler aralığında olması sağlanır. Bu türden değerler bilgisayar hafızasında tam olarak saklanamadığı için dikkatli bir şekilde kullanılması gerekmektedir. Özellikle finans uygulamalarında gerçek değer ile saklanan veri türleri tercih edilmelidir.

Örnek:

```
float boy=150.35;
cout<<"Boy= "<<boy<<endl;
Çıktı:
Boy= 150.35
```

GÜNLÜK HAYATLA İLİŞKİLENDİRELİM

Günlük hayatta kullandığımız bilgilerin hangileri tam sayı hangileri kesirli sayı veri türündedir.

2.3. Boolean Veri Türü

Bir boolean veri türü sadece "true" veya "false" olmak üzere iki farklı değer alır. Karar alma veya ilişkisel işleçlerde kodun okunabilirliğini artırmak için tercih edilir. Bir boolean değişkeni hafızada 1 baytlık alanda saklanır. Bir boolean değişkeni bool anahtar kelimesi ile tanımlanır. bool aktif=true; Eğer boolean veri türündeki bir değişkenin çıktısının alınması durumunda true için 1 false için 0 değeri yazdırılır.

Örnek:

```
bool aktif = true;
cout <<aktif <<endl;
aktif = false;
cout <<aktif <<endl;
```

Çıktı:

```
1
0
```

2.4. Karakter Veri Türü

Karakter veri türü standart ASCII karakter setindeki 256 karakter veya sembollerden birini temsil eder ve hafızada bir baytlık alanda saklanır. Bir değişkene atama yapılırken tek tırnak içerisine alınması gerekir.

Örnek:

```
char harf = 'a';
cout << harf <<endl;
harf = 'A';
cout <<harf <<endl;
harf = '5';
cout <<harf <<endl;
harf = '?';
cout <<harf <<endl;
```

Çıktı:

```
a
A
5
?
```

Tablo 5.2 Kaçış Karakterlerinin Kullanımı.

\n	Sonraki satıra geç
\t	Klavyedeki tab tuşuna karşılık gelen karakterdir
\'	Tek tırnak
\\	Ters eğik çizgi (backslash)
\0	Boş (null)

Karakter veri türü için <cctype> kütüphanesinde tanımlamış fonksiyonlarının bazıları Tablo 5.3'de verilmiştir.

Tablo 5.3 Karakter Veri Türü İçin Tanımlanmış Bazı Fonksiyonlar.

int isdigit(int c)	Eğer c bir rakam ise 1, değilse 0 döndürür.
int isalpha(int c)	Eğer c bir harf ise 1, değilse 0 döndürür.
int islower(int c)	Eğer c küçük harf ise 1, değilse 0 döndürür.
int isupper(int c)	Eğer c büyük harf ise 1, değilse 0 döndürür.
int tolower(int c)	Eğer c büyük harf ise küçük harfe dönüştürür, değilse değişiklik yapmaz.
int toupper(int c)	Eğer c küçük harf ise büyük harfe dönüştürür, değilse değişiklik yapmaz.

2.5. Numaralandırma (enum) Veri Türü

Numaralandırma veri türü bir sabit listesi oluşturmak için kullanılır. Örneğin araba markalarından oluşan bir enum veri türü aşağıdaki gibi tanımlanabilir.

```
enum araba {Honda, Toyota, Mazda, Subaru};
```

Eğer liste içerisindeki değerlere programcı tarafından bir değer ataması yapılmaz ise listedeki ilk değere 0 sonraki değerler birer artırılarak değer ataması yapılır. Bu durumda yukarıda verilen örnekteki gibi bir tanımlama yapılırsa Honda "0", Toyota "1", Mazda "2" ve Subaru "3" değerlerini alırlar. Programcı isterse listedeki her bir değere aşağıda verilen örnekte olduğu gibi uygun bir atama da yapabilir.

```
enum araba {Honda=10, Toyota=20, Mazda=30,
Subaru=40};
```

Örnek:

```
enum araba {Honda=10, Toyota=20, Mazda=30,
Subaru=40};

araba mycar=Toyota;

cout<<mycar;
```

Çıktı:

```
20
```

Değişkenler

Değişken bilgisayar belleğinde verinin saklandığı ve üzerinde işlemler yapıldığı bir depolama konumudur. Bir değişken adı, sakladığı içeriğe göre veri türü ve üzerinde çağrılabileceği işleçler ile tanımlanır.

Değişken Bidirimi

Bir değişken saklayacağı veri türü ve değişken adıyla birlikte tanımlanır. Örneğin kişinin yaş bilgisini tutacak değişkenin bildirimi aşağıdaki gibi olacaktır.

```
int yas;
```

Bir değişkenin bildirimi yapıldıktan sonra kullanmak için bir değer ataması yapılması gerekir. Örneğin kişinin yaşının 40 olduğu durumda değer atama işlemi için aşağıdaki ifade yazılmalıdır.

```
yas = 40;
```

C++ programlama dili bir değişkenin hem bildiriminin hem değer atamasının aynı satırda yapılabilmesini mümkün kılmaktadır. Dolayısıyla yukarıda iki ayrı satırda gerçekleştirilen bildirim ve atama işlemini tek satırda yapmak için yazılması gereken ifade aşağıdaki gibi olmalıdır.

```
int yas = 40;
```

Eğer daha sonra yas değişkenine tekrar ihtiyaç duyulursa veri türü bilgisi olmadan kullanılması gerekir. Örneğin yas değerini 10 artırmak istiyorsak aşağıdaki gibi aritmetik ifade yazılmalıdır.

```
yas = yas + 10;
```

Gerektiği durumlarda birden fazla aynı türe ait değişkenin aynı satırda bildirimi gerçekleştirilir. Örneğin tam sayı olarak tanımlanacak yas ve maas değişkenleri için aşağıdaki ifade uygun olacaktır.

```
int yas, maas;
```

Birden fazla değişkenin bildirimi yapılırken aynı zamanda da değer ataması yapılabilir.

```
int yas=40, maas=1000;
```

Eğer değişkenler farklı türden iseler ayrı satırlarda bildirim yapılması gerekmektedir.

```
int yas, maas;
float kilo, boy;
```

Değişkenlerin İsimlendirilmesi

Bir değişkene uygun ve anlamlı bir isim vermek önemlidir. Çünkü programcının yazacağı kodu sadece kendisi değil aynı zamanda aynı projede çalışan yazılım ekibindeki diğer programcılar da kullanabilecektir. Ayrıca bir yazılımın yaşam döngüsü içerisinde bulunan test aşamasında genellikle kodu yazan kişiden farklı biri test sürecini yürütebilmektedir. Belki ilerleyen zamanlarda yazılıma bakım yapılması gerekebilecek ve büyük ihtimalle yine kodu yazan tarafından değil başka programcılar tarafından bakım süreci yürütülecektir. Örneğin bir kişinin yaş bilgisini saklayacak bir değişkene yas ismini vermek uygundur. Ama x veya y gibi bir isim vermek uygun olmayacaktır.

Bir değişkene uygun bir isim verilebilmesi için bazı kurallara uymak gerekmektedir. Bu kurallar aşağıda belirtilmiştir.

• Değişken isimleri rakam ile başlayamaz.

• Değişkenlerin isimleri için "!, ?, {,]" gibi özel karakterler kullanılamaz.

• C++ programlama dili büyük ve küçük harf duyarlıdır. Örneğin var, VAR ve Var üç farklı değişken olarak algılanır.

• Değişken isimleri İngiliz alfabesinde bulunan büyük (A-Z) veya küçük (a-z) karakterler ya da rakamlar (0-9) kullanılarak oluşturulmalıdır.

• Değişken isminin uzunluğu en fazla 255 karakter olabilir ama ilk 32 karakter anlamlıdır. Yani ilk 32 karakteri aynı olan 32. karakterden sonraki herhangi bir karakteri farklı olan değişkenler aynı değişken olarak kabul edilir.

• Değişken ismi birden fazladan kelimeden oluşuyor ise kelimeler arasında boşluk bırakarak değişken ismi oluşturulamaz. (Örneğin "en buyuk değer"). Bu durumda ya kelimeler boşluk olmadan birleştirilebilir ("enbuyukdeger") ya da kelimeler arasında alt çizgi kullanılabilir (en_buyuk_deger). Genellikle kod okunabilirliği artırmak için deve (camel) gösterimi gibi bir standart tercih edilmektedir ("enBuyukDeger"). Deve gösteriminde değişken isminin ilk kelimesinin ilk harfi küçük sonraki kelimelerin ilk harfleri büyük kullanılması tercih edilir.

• C++ programlama dilinde anahtar kelime olarak tanımlanan kelimeler değişken ismi olarak kullanılamaz (Örneğin: for, while, case gibi kelimeler bir değişkene isim olarak tanımlanamazlar). C++dilindeki tanımlı anahtar kelimeler Tablo 5.5'de verilmiştir.

Tablo 5.4'de değişken isimlendirme kurallarına göre doğru ve yanlış örnekler verilmiştir.

Tablo 5.4 Değişken İsimleri için Doğru ve Yanlış Örnekler.

Doğru	Yanlış
adSoyad	ad soyad
abc9	9abc
buyuk	büyük
en_buyuk	[enbuyuk]

Tablo 5.5 C++ Programlama Dilinde Tanımlı Anahtar Kelimeler.

asm	auto	break	case	char
const	continue	default	do	double
else	enum	extern	float	for
goto	if	inline	int	long
register	return	short	signed	sizeof
static	struct	switch	typedef	union
unsigned	void	volatile	while	
and	and_eq	bitand	bitor	bool
catch	class	compl	const_cast	delete
dynamic_cast	explicit	export	false	friend
mutable	namespace	new	not	not_eq
operator	or	or_eq	private	protected
public	reinterpret_ cast	static_cast	template	this
throw	true	try	typeid	typename
using	virtual	wchar_t	xor	xor_eq
alignas	alignof	char16_t	char32_t	constexpr
decltype	noexcept	nullptr	static_assert	thread_local

3. İŞLEÇLER

İşleçler bir programın sonuç üretebilmesi için gerekli matematiksel veya mantıksal hesaplamaların yapılmasını sağlar. İşleçlerin bir sonuç üretebilmesi için bir veya daha fazla işlenen kullanılarak ifadelerin oluşturulması gerekir. Bir işleç, eğer sonuç üretebilmesi için tek bir işlenen yeterli ise birli (unary), iki işlenene gereksinim duyuyor ise ikili (binary) ve üç işlenen gerekiyor ise üçlü (ternary) olarak ifade edilir. Örneğin artırma ve eksiltme işleci birli bir işleçtir. Bir değişkene ait değerin 1 artırılması veya eksiltmesi amacıyla kullanır. Yani işlecin bir sonuç üretmesi için bir değişken yeterlidir. Diğer taraftan toplama çıkarma gibi aritmetik işleçler iki işlenen kullanarak sonuç üretirler. Şartlı işleç (? :) ise üç işlenene gereksinim duyar. İşleçleri kullanım amaçlarına göre sınıflandırılması aşağıdaki gibidir:

- Atama işleçleri
- Aritmetik işleçler
- İlişkisel işleçler
- Mantıksal işleçler
- Bit Tabanlı İşleçler

Bu bölümde atama ve aritmetik işleçler üzerinde durulacaktır. Diğer işleç türleri sonraki bölümlerde açıklanacaktır.

Atama İşleçleri

Bir değişkene bir değer atamak için kullanılır. Atama işleci "=" sembolü kullanılarak gerçekleştirilir. Atama işlecinin sağındaki değer işlecin solundaki değişkene atanır. İşlecin sağındaki değer sabit bir sayı olabileceği gibi aritmetik, mantıksal veya ilişkisel bir ifade de olabilir. Örneğin 50 yaşında olan bir kişinin yaşını tanımlı yas değişkenine atanması isteniyorsa aşağıdaki gibi bir ifade yazılmalıdır:

```
yas = 50;
```

Aritmetik İşleçler

Matematiksel ifadelerin hesaplanabilmesi için bir veya daha fazla işleç ve işlenen kullanarak bir programın sonuç üretebilmesi sağlanır. C++ programlama dilinde toplama (+), çıkarma (-), çarpma (*), bölme (/) gibi sıklıkla kullanılan aritmetik işleçler tanımlıdır. Bunlara ek olarak kalan (%), artırma (++) ve azaltma (--) gibi işleçler de mevcuttur.

Toplama ve Çıkarma İşleçleri

Toplama işleci iki işlenen değerini toplamak için kullanılır. Örneğin sayi1 ve sayi2 değişkenlerinin toplamının elde edilmesi ve sonucun sonuc değişkenine atanması isteniyorsa yazılması gereken aritmetik ifade aşağıdaki gibi olmalıdır:

```
sonuc = sayi1 + sayi2;
```

Benzer şekilde iki değişkenin farkının hesaplanması gerektiğinde yazılması gereken ifade aşağıdaki gibi olmalıdır:

```
sonuc = sayi1 - sayi2;
```

Çarpma ve Bölme İşleçleri

İki değişkene ait değerlerin çarpılması isteniyorsa yazılması gereken ifade aşağıdaki gibi olmalıdır:

```
sonuc = sayi1 * sayi2;
```

Bir değişkene ait değerin diğer bir değişken değerine bölünmesi durumunda elde edilen bölüm değerini başka bir değişkene atamak için aşağıdaki ifade yazılmalıdır:

```
sonuc = sayi1 / sayi2;
```

DİKKAT EDELİM

Bir tam sayı olarak tanımlanmış değişkenin yine bir tam sayı değişkenine bölümü sonucunda bir tam sayı değeri elde edilir. Kesirli değer elde edilmek isteniyorsa tür dönüşümü yapılması gerekir.

Örnek:

```
int a = 5;
cout << a << endl;
cout << --a << endl;
cout << a << endl;
```

Çıktı:

```
Fiyat / Kilo = 33
Fiyat / Kilo = 33.3333
```

Kalan işleci

Bazı durumlarda bir bölme işlemi sonucu ortaya çıkan kalan değerinin elde edilmesi istenebilir. Kalan işleci bir bölme işlemi sonunda ortaya çıkan değeri elde etmek için kullanılır. Bu işleç "%" sembolü ile kullanılır. Örneğin sayi1 değişkenin sayi2 değişkenine bölünmesi durumunda elde edilecek kalan değerini sonuc değişkenine atanması gerekiyorsa aşağıdaki ifade yazılmalıdır:

```
sonuc = sayi1 % sayi2;
```

Eğer sayi1 değişkenin değeri 12 ve sayi2 değişkenin değeri 5 olması durumunda kalan 2 olacağı için sonuc değişkenine 2 değeri atanır.

Artırma ve Azaltma İşleçleri

Özellikle döngüler oluşturulurken kullanılan ifadelerde bir döngünün kaç defa çalışmasını kontrol eden sayaç değişkenin problemin çözümüne bağlı olarak ya 1 artırılması ya da 1 azaltılması söz konusudur.

Dolaysıyla toplama veya çıkarma işleçlerini kullanarak istenen sonuçlar elde edilebilir. Bu durumda yazılması gereken ifade aşağıdaki gibi olmalıdır.

```
sayac = sayac +1
veya
sayac = sayac – 1
```

Bu ifadeler yazılımcılar tarafından sıklıkla kullanıldıkları için C++ programlama diline 1 artırma (++) ve 1 çıkarma (--) işleçleri eklenmiştir. Böylece bir değişkenin değeri 1 artırılmak isteniyorsa sayac++ veya 1 azaltılması isteniyorsa sayac-- ifadeleri tercih edilebilir.

Her iki işleç iki farklı şekilde kullanılmaktadır. Eğer işleç işlenenden önce kullanılırsa (++sayac veya --sayac şeklinde) önce artır veya azalt, işlenenden sonra kullanılırsa (sayac++ veya sayac-- şeklinde) sonra artır veya azalt olarak tanımlanır. Her iki kullanıma bağlı olarak aritmetik ifadelerin sonuçları farklılık gösterilmektedir. Bu yüzden her iki işleç dikkatli bir şekilde kullanılmalıdır. Kullanıma bağlı olarak farklı sonuç üretebilecek durumlar aşağıdaki örnek kod üzerinden anlatılacaktır.

Örnek:

```
int a = 5;
cout << a << endl;
cout << --a << endl;
cout << a << endl;
```

Çıktı:

```
5
4
4
```

İlk olarak işleç uygulanmadan önce, değişkenin ilk atanan değeri 5 yazdırılır. İşlecin uygulanmasıyla değişkenin değeri önce 1 azaltılır sonra ekrana yazdırılır. Bu yüzden ekrana 4 değeri yazdırılır. İşlecin kullanılmasında sonra değişken değeri tekrar kontrol edildiğinde 4 olduğu gözlemlenir.

Örnek:

```
int a = 5;
cout << a << endl;
cout << a-- << endl;
cout << a << endl;
```

Çıktı

```
5
5
4
```

Yakarıdaki kod bloğunda sonra azalt işleci kullanılmıştır. İlk olarak değişkenin ilk değeri olarak 5 yazdırılır. Sonrasında değişkenin değeri herhangi azaltma yapılmadan yazdırılır. Bu durumda ekrana 5 değeri yazdırılır. İşlecin kullanılmasında sonra değişken değeri tekrar kontrol edildiğinde 4 olduğu gözlemlenir.

Artırma ve azaltma işleçleri diğer aritmetik işleçlerle birlikte matematiksel ifadeler içerisinde beraber kullanılabilir.

Örnek:

```
int a=5, b=10;
cout<<(a++)*(--b)<<endl;
```

Çıktı:

```
45
```

DİKKAT EDELİM

Hatalı sonuçlar üretmemek veya kod okunabilirliğini artırmak için aritmetik ifadelerde uygun yerlerde parantezler kullanılmalıdır.

Bileşik Atama İşleçleri

Programlarda genellikle aynı değişkenin değeri sabit veya başka değişkenler kullanılarak güncellenmesi gerekir. Örneğin tam sayı tanımlı var değişkenin değeri 5 olsun ve değişkenin değerini 3 artırmak için yazılması gereken ifade "var = var + 3;" olacaktır. Görüldüğü gibi ifade içerisinde iki defa değişken adının tekrar edilmesi söz konusudur. Bu ve buna benzer ifadeleri kısaltmak için bileşik atama işleçleri tanımlanmıştır. Aşağıda verilen tabloda görüleceği üzere toplama işlecine ek olarak çıkarma, çarpma, bölme ve kalan bileşik atama işleçleri de tanımlıdır.

Tablo 5.6 Bileşik Atama İşleçleri ve Kullanımları.

İşleç	Örnek	Eşdeğeri
+=	var +=3;	var = var + 3;
-=	var -=3;	var = var - 3;
*=	var *=3;	var = var * 3;
/=	var /=3;	var = var / 3;
%=	var %=3;	var = var % 3;

Aritmetik İşleçlerin Öncelik Sıraları

Bir matematiksel ifade içerisinde birden fazla aritmetik işleç kullanılarak daha karmaşık ifadelerin hesaplanması sağlanabilir. Böyle ifadelerin doğru şekilde sonuç üretebilmesi için aritmetik işleçlerin öncelik sırası bilinmelidir. Değilse hatalı sonuç almak mümkündür. Örneğin, a + b * 3 - (c / d) ifadesinde toplama, çarpma, çıkarma ve bölme işleçleri aynı ifade içerisinde kullanılmıştır. İlk hangi işlecin hesaplanması gerektiği, aşağıda verilen kurallara göre belirlenmektedir.

Aritmetik İşleçlerin Öncelik Kuralları

• Eğer ifadede parantez varsa öncelikli olarak parantez içerisindeki işleçler hesaplanır. Parantezin içerisinde de başka parantezler var ise en içteki parantezden başlayarak hesaplamalar yapılır.

• Daha sonra çarpma, bölme ve kalan işleçleri hesaplanır. Eğer bir ifadede birden fazla çarpma, bölme veya kalan işleci var ise hesaplama soldan başlanarak sağa doğru devam edilir. Çarpma, bölme ve kalan işleçleri aynı seviye önceliği sahiptir.

•En son olarak toplama ve çıkarma işleçleri uygulanır. Eğer ifade içerisinde birden fazla toplama veya çıkarma işleci var ise, ifadenin hesaplanmasına soldan başlanacaktır. Toplama ve çıkarma işleçleri aynı seviye önceliğe sahiptir.

Örnek: y = 1* 2 + (3 + 4) – 5 ifadesinin hesaplanmasında sırasıyla şu işleçler uygulanacaktır. Öncelikle parantez içi hesap edilerek ifade y = 1* 2 + 7 – 5 halini alacaktır. Daha sonra kalan işleçler arasında çarpma işleci öncelikli olduğu için çarpma işlemi uygulanacaktır. İfade y = 2 + 7 – 5 halini alacaktır. Kalan toplama ve çıkarma işleçleri aynı seviyede oldukları için soldan başlanarak hesaplama devam edilecektir. Kalan ifade y = 9 – 5 olacaktır. En son olarak çıkarma işlemi uygulanarak sonuç y = 4 olarak bulunacaktır.

DİKKAT EDELİM

Doğru sonuçlar elde edebilmek için işleçlerin öncelik kuralları bilinmelidir.

BÖLÜM ÖZETİ

Bu bölümde temel veri türleri, değişkenler, atama ve aritmetik işleçlerden bahsedilmiştir. Bir bilgisayar programı hafızada sakladığı çeşitli verileri işleyerek birtakım sonuçlar üretir. Birçok farklı alanda programlara ihtiyacımız olduğu düşünülünce çok farklı veri türlerine gereksinim duyulmaktadır. C++ programlama dilinde tanımlı üç farklı temel veri türü vardır. Bu türler sayısal, boolean ve karakter veri türleridir. Sayısal veri türleri kendi içinde tam ve kesirli sayılar olmak üzere ikiye ayrılmaktadır. Bir bilgisayar programı programcının tanımladığı değişkenleri işleyerek istenilen sonuca ulaşır. Değişkenlere uygun isim vermek yazılan kodun okunabilirliğini artırır. Çünkü yazılım yaşam döngüsünde bulunan süreçlerin gereği olarak bir programcının yazmış olduğu kodu başka programcılarında kullanması veya test etmesi gerekmektedir. Ayrıca C++ programlama dilinde bir değişkene isim vermek için bazı kurallar vardır. Değişkenlere bu kurallara uygun isimler vermek derleme hatalarının önüne geçecektir. Tanımlı değişkenler aritmetik, ilişkisel veya mantıksal işleçler içerisinde kullanılarak daha karmaşık ifadelerin yazılması mümkün hale gelmektedir. İşleçler konusu oldukça geniş bir kavramdır. Bu bölümde atama ve aritmetik işleçlerden bahsedildi. Atama işleci bir değişkene bir değer atamak için kullanılır. Buna ek olarak toplama, çıkarma, bölme, çarpma ve kalan gibi aritmetik işleçler anlatılmıştır. Bir ifade içerisinde birden fazla işleç kullanarak daha karmaşık ifadelerin elde edilmesi mümkündür. Bu durumda hangi işlecin önce hesaplanması gerektiği işlem öncelik kurallarına göre belirlenir. Bu kuralları bilmek program içerisinde yapılabilecek hataları en aza indirgeyecektir.

GÖZDEN GEÇİRELİM

1. **Aşağıda verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

```
int a = 2 * 5 - 3 + 4 % 2;
cout << a <<endl;
```

a. 3
b. 4
c. 5
d. 6
e. 7

2. **Aşağıda verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

```
int a = 4, b = 4;
cout << (++a) * (b--) <<endl;
```

a. 12
b. 15
c. 16
d. 20
e. 24

3. **Aşağıda verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

```
int a= 4, b = 4;
++a;
b++;
int c = a * b;
cout << c <<endl;
```

a. 16
b. 20
c. 24
d. 25
e. 30

4. **Aşağıdakilerden hangisi C++ programlama dilinde tanımlı bir tam sayı veri türüdür?**

a. int
b. float
c. double
d. char
e. string

5. **Aşağıdakilerden hangisi C++ programlama dilinde tanımlı bir kesirli sayı veri türüdür?**

a. short
b. float
c. int
d. long
e. char

6. **Aşağıdakilerden hangisi C++ programlama dilinde atama işleci olarak kullanılır?**

a. =
b. *
c. %
d. /
e. ==

7. **Aşağıda verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

```
int a = 10;
a += 2;
cout << a <<endl;
```

a. 5
b. 8
c. 10
d. 12
e. 20

8. **Aşağıda verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

```
int sayi1=13, sayi2=3;
int sonuc = sayi1 % sayi2;
cout<<sonuc<<endl;
```

a. 0
b. 1
c. 2
d. 3
e. 4

9. **Aşağıdakilerden hangisi C++ programlama dilinde tanımlanabilecek bir değişken ismi değildir?**

a. boy
b. kilo
c. ad
d. soyad
e. enKüçük

10. **Aşağıda verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

```
enum renkler {siyah, beyaz, gri};
renkler araba = beyaz;
cout<<araba<<endl;
```

a. 0
b. 1
c. 2
d. 5
e. 10

Yanıt Anahtarı: 1-E, 2-D, 3-D, 4-A, 5-B, 6-A, 7-D, 8-B, 9-E, 10-B

Kaynakça

Deitel, P., Deitel, H. (2017). C++ How to Program. 10. Baskı. Pearson. Essex, İngiltere.

Dikici, M. (2017). C++ Programlama Dili: Yapısal Programlama, Nesnelerle Programlama ve Jenerik Programlama. Seçkin Yayıncılık. İstanbul, Türkiye.

Çobanoğlu, B. (2021). C/C++ Programlama Eğitim Kitabı. Kodlab. İstanbul, Türkiye.

Demir, S., Eryiğit, R., Ar, Y., Tuğrul, B., Gök, O., Kul, S., . . . Yağanoğlu, M. (2019). Programlama Temelleri. Erzurum: Atatürk Üniversitesi Açıköğretim Fakültesi.

Kaleli, C., Bilge, A., Ergin, S., & Şora Günal, E. (2016). Bilgisayar ve Programlamaya Giriş. Eskişehir: Anadolu Üniversitesi.

Faydalı Kaynaklar ve İnternet Kaynakları

https://en.cppreference.com/w/

https://www.w3schools.com/cpp/default.asp