BİL-142 Bilgisayar Programlama II (C/C++)

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Genel Bilgiler

Öğretim üyesi : Doç.Dr.M.Ali Akcayol
Ofis : Gazi Ünv., Bilgisayar Mühendisliği Bölümü
E-Posta : akcayol@gazi.edu.tr
Ofis saatleri : Pzt 14:30-15:30

Ders asistanları :

Dersin web sayfası : http://w3.gazi.edu.tr/~akcayol
(\dersler\Bilgisayar Programlama (C/C++))
Derslik :

2

Genel Bilgiler

Değerlendirme

Arasınav : 25%
Lab : 25%
Haftalık ödevler : 10%
Derse katılım : 5%
Final : 35%

Temel ders kitabı

Deitel & Deitel, "C++ How to Program 6/e", Prentice Hall, 2007.

Yardımcı ders kitabı

Walter Savitch, "Absolute C++ (4th Edition)", Addison Wesley, 2009.

3

Genel Bilgiler

Ders konuları

(1) C/C++ programlamaya giriş
(2) Sınıflar ve nesnelere giriş
(3) Kontrol yapıları
(4) Döngü yapıları
(5) Fonksiyonlar
(6) Diziler
(7) İşaretçiler
(8) Sınıflar ve özellikleri
(9) Nesne yönelimli programlamaya giriş
(10) Nesne yönelimli programlama: kalıtım
(11) Nesne yönelimli programlama: çok biçimlilik
(12) Dosyalar ve stream yapıları

4

Programlamanın temelleri

- Problemin veya amacın anlaşılması ve gerçekleştirmek için planlama yapılması gerekir.
- Algoritma tasarımı
  - Algoritma, problemin çözümü için takip edilen adımlar, kurallar kümesi veya süreçtir.
  - Programın kodlanmasından önce problemin çözümü için adımların oluşturulması gerekir.

5

Programlamanın temelleri

- Başarılı bir programlama için aşağıdaki adımlar izlenmelidir:
  - Adım 1: Problemin anlaşılması, programın girişlerinin ve çıkışlarının belirlenmesi.
  - Adım 2: Problemin çözümü için gerekli bileşenlerin belirlenmesi.
  - Adım 3: Programın anahtar özelliklerinin belirlenmesi, akış diyagramın ve pseudo kodun oluşturulması.
  - Adım 4: Programın test edilmesinde her bir parçanın belirlenmesi ve test edilmesi.
  - Adım 5: Sonraki versiyonlardaki gereksinimlerin belirlenmesi ve önceki adımların tüm versiyonlar için tekrarlanması.

6

Terminoloji

Term	Meaning
bug	General catchall word meaning the program is not running correctly.
class	A "job description" for a program component. The job description includes the tasks that the "worker" performs and associated data.
code	This term can refer to the textual-based phrases written in C++ that represent a program (or portion of a program). Code can refer to a single line, such as "a line of code" or the entire program. Also, it can refer to program file contents such as "machine code" or "executable code."
compiler	Actual software (such as Microsoft Visual C++) that reads C++ statements. It checks that the statements are written with the correct syntax. The compiler produces object or machine code.
debugger	A tool in the software development package (such as Microsoft Visual C++) that allows the programmer to run the program one step at a time and to examine program portions. A debugger is used to track down bugs.
executable	The machine language file that the operating system reads. The operating system performs instructions based on the commands in this file. The executable file constitutes the program.

7

Terminoloji

function	A discrete module or unit of code that performs specific tasks.
IDE	Integrated Development Environment. An IDE is a software development program (such as Visual Studio 2005 Express) that provides the developer complete tools for building software. IDEs contain an editor and tools for linking and executing code as well as access to Help files.
linker	Software that combines all the required files together and builds an executable file.
object	In object-oriented programming, an instance of a class.
object code	File produced by a compiler. The source code file is translated into machine language.
RAM	Random Access Memory. Actual computer chips that contains storage areas that are directly accessed by the computer operating system and programs. As a program executes, its data items are stored in RAM.
source code	The text-based file containing C++ statements. The source code is read by the compiler.
syntax	The correct way in which the language's words and symbols are put together so that they have meaning to the C++ language. It may be thought of as the "grammar" and "punctuation" rules for the language.

8

C++ programının temel formatı

- C++ case sensitive programlama dilidir.
  - "toplam", "Toplam" ve "TOPLAM" birbirinden ayrıdır.
- C++ programları fonksiyonlar halinde yazılır.
  - Tüm fonksiyonlar, bir isme, data gönderme ve data döndürme özelliklerine sahiptir.
  - Fonksiyonlar programın işlevlerini gerçekleştirir.
  - C++ programlarının başlangıç noktası **main()** fonksiyonudur.

9

Merhaba dünya! programı

```
//Merhaba dünya! programı.
//08.05.2009

#include <iostream>
using namespace std;

int main()
{
   cout << "Merhaba dünya!  \n";
   return 0;
}
```

10

Açıklamalar

- Açıklama satırları pogram hakkında bilgiler vermek için kullanılır.
- Compiler açıklama satırlarını gözardı eder.
- C++ içinde açıklama yazmanın iki yolu vardır;
  - Tek satırlık açıklamalar

```
// Açıklamalar bu satıra yazılabilir.
// Compiler iki slaş işaretinden satır sonuna
// kadar herşeyi gözardı eder
```

  - Çok satırlık açıklamalar

```
/* Açıklama yazmanın diğer bir yoludur. Compiler
slaş yıldız ile yıldız slaş arasındaki her şeyi
gözardı eder. */
```

11

Önişlemci direktifleri

`#include <iostream>` ön işlemci direktifidir.

- Önişlemci direktifleri compiler'a komutlar gönderir.
- ***`iostream`*** bir kütüphanedir. Klavyeden okuma ve ekrana yazma için gerekli deyimleri bulundurur.
- ***`#include`*** deyimi ile compiler'a **`iostream`** araçlarının kullanılacağı bildirilmiştir.
- C/C++ çok sayıda kütüphaneye ve araca sahiptir.
- C/C++ fonksiyonları kullanılacağı zaman uygun kütüphanenin programa include edilmesi gerekir.

```
#include <cmath>        //matematiksel fonksiyonlar için
                        //kullanılır, sqrt(), cos(),…
#include <iostream>     //ekrana yazmak veya
                        //klavyeden okuma yapar
#include <iomanip>      //çıkış formatını belirler
```

12

using deyimi

- Aşağıdaki deyim C++ kütüphanesindeki standart alan adının kullanıldığını compiler'a belirtir.
- `using namespace std;`
- ***#include*** deyiminden sonra kullanılır.

13

main() fonksiyonu

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

- C++ programının başlangıç noktasıdır.
- C++ programları bir `main()` fonksiyonuna sahiptir.

14

Fonksiyon başlık satırı

- Fonksiyon başlık satırı her fonksiyonda olur. Fonksiyonun adını ve giriş/çıkış parametrelerini tanımlar.
- Genel yazımı,

```
return_type function_name(input parameters)
```

şeklindedir.

- `return_type`, fonksiyonu çağıran yere döndürülecek değerin türünü belirler.
- Fonksiyonlar `{` ve `}` parantezleriyle oluşturulur.

15

C++ deyimleri

- C++ deyimleri çalıştırılacak komutları ifade eder.
- C++ daki çoğu deyim noktalı virgülle sonlandırılır.
- Merhaba dünya! programı iki deyime sahiptir.

```
cout << "Merhaba dünya!";
return 0;
```

- `cout` çıkışı yönlendirir.
- "<<" operatörü datayı konsol ekranına gönderir.
- `return 0;` işletim sistemine sıfır değeri döndürür.

16

Örnek

```
/* Bu program ekrana hava durumuyla ilgili metinler
   yazar.   */

#include <iostream>
using namespace std;

int main()
{
   //Hava bilgisi
   cout << "\n Bugün hava bulutlu \n";
   cout << "\n Hava yağmurlu olabilir \n";
   cout << "\n Hava güneşli bugün \n";
   return 0;
}
```

17

Boşluk karakterleri ve C++ ile esnek yazım

- Boşluk karakterleri (whitespaces) programın okunabilirliğini artırır.
- Enter, tab ve space ile oluşturulur.
- Compiler gözardı eder.
- Merhaba dünya! programı aşağıdaki gibi yazılırsa yine çalışır.

```
//Tek satırla merhaba dünya programı

#include <iostream>
using namespace std;

int main(){ cout << "Merhaba dünya! \n"; return 0;}
```

18

Syntax

- C++ programları syntax kurallarına uygun yazılmalıdır.

```
//Merhaba dünya! programı
//Compile yapılamaz

#INCLUDE <iostream>          // #include büyük harfle
                             // yazılamaz
using namespace standard;    // C++ için "std"
                             // doğrudur, standard
                             // yanlıştır
int Main ()                  // main küçük harf
                             // olmalıdır
{
   Cout << "Merhaba dünya!";      // cout küçük harf
                                  // olmalıdır
   return 0                  // ; yazılmamış
}
```

Hello.cpp(4) : fatal error C1021: invalid preprocessor command 'INCLUDE'

19

Syntax

- Önişlemci yazım hatası düzeltilirse aşağıdaki hataları verir.

Hello.cpp(5) : error C2871: 'standard' : does not exist or is not a namespace

Hello.cpp(9) : error C2065: 'Cout' : undeclared identifier

Hello.cpp(9) : error C2297: '<<' : illegal, right operand has type 'char [13]'

Hello.cpp(11) : error C2143: syntax error : missing ';' before '}'

- Programdaki az sayıdaki hata, çok sayıda compiler hatası üretebilir.

20

İyi program yazım şekli

- C++ ile yazılan programın kolay okunabilir olması gerekir.
- Tanımlayıcı açıklamaların yapılması gerekir.
- Anlamlı ve uygun uzunlukta değişken isimlendirme yapılması gerekir.
- Programdaki blokların hizalandırılması gerekir.
- Başlangıçta okunabilirlik için harcanan zaman, compiler hatalarının düzeltilmesi veya programın update edilmesi sırasında çok zaman kazandırır.

21

C++ anahtar kelimeleri

- C++ çok sayıda ayrılmış kelimeye sahiptir.
- C dili 31 ve C++ dili 63 anahtar kelimeye sahiptir.
- **Anahtar kelimeler** özel anlama sahiptir ve **değişken adı olarak kullanılamaz**.

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	return	reinterpret_cast	short	signed
sizeof	static	static_cast	struct	switch	template
this	throw	true	try	typedef	typeid
typename	union	unsigned	using	virtual	void
volatile	while	wchar_t			

22

Data türleri

- Bir veri türü programda kullanılacak değeri belirler.
- Her değer için bir tür belirlenmelidir.

Data Type	Name	The Data It Contains	Example
`char`	char or character	A single character	a
`int`	integer	A whole number (no decimal point)	43
`float`	float or floating point	A number with six to seven digits of precision	14.937453
`double`	double	A number with thirteen to fourteen digits of precision	3.14159265294753
`void`	void	Empty or nothing; specifies function as returning no values	(is presented later)
`bool`[a]	boolean	Stores values of true or false	true
`wchar_t`[a]	wide character	Holds wide characters (16 bits)	Japanese character

[a]*Note*: Not defined in the C language.

23

Data türleri

- Bir veri türü, veri saklama alanını ifade eder.
- Bir değişken, veri saklama alanının adını gösterir.
- Her veri türü, saklama alanının boyutunu belirler.

Bits	Possible Bit Combinations		Unique Values
1	0	0	The number of unique values can be calculated by 2^n, where n = number of bits.
	1	1	$2^1 = 2$ combinations
2	00	0	
	01	1	$2^2 = 4$ combinations
	10	2	
	11	3	
3	000	0	
	001	1	
	010	2	$2^3 = 8$ combinations
	011	3	
	100	4	
	101	5	
	110	6	
	111	7	
8	00000000	0	
	00000001	1	$2^8 = 256$ combinations
	⋮	⋮	
	11111111	255	

Bytes	Bits	Possible Combinations	Signed Range	Unsigned Range
1	8	256	-128 to 127	0 to 255
2	16	65,536	-32,768 to 32,767	0 to 65,535
4	32	4,294,967,296	-2,147,483,648 to 2,147,483,647	0 to 4,294,967,295

Data türleri

Keyword	Typical Bytes of Memory	Precision Range
char	1	–128 to 127
unsigned char	1	0 to 255
signed char	1	–128 to 127
int	4	–2,147,483,648 to 2,147,483,647
short int	2	–32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
long int	4	–2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295
float	4	6 digits, i.e., 0.xxxxxx
		3.4 E ± 38 (7 digits in Visual C++)
double	8	10 digits, i.e., 0.xxxxxxxxxx
		1.7 E ± 308 (15 digits in Visual C++)
long double	10	10 digits, i.e., 0.xxxxxxxxxx
		1.2 E ± 4932 (19 digits in Visual C++)

25

Data türleri

- Veri türü tanımlama formatı

```
data_type variable_name;
```

şeklindedir.

- Örnek veri türü tanımlamaları:

```
float balance;
float deposit;
float withdraw;
int transaction_count;
int check_number;
```

26

Data türleri

- Değişkenlere değer atanması aşağıdaki gibi yapılır;

```
float ave;
int num = 5;
ave = 35.29;
double money, speed;
money = speed = 0.0;
```

27

C++ ile isimlendirme kuralları

- Değişkenlere isim belirlenmesinde uyulacak kurallar:
  - İsimler A-Z, a-z, 0-9 veya _ karakterlerinden oluşur.
  - İlk karakter harf veya _ olmalıdır.
  - İsimlerde ~ ! @ # $ % ^ & * ( ) – "+ = \ | ', sembolleri ve boşluk karakteri kulanılamaz.
  - Anahtar kelimeler değişken ismi olarak kullanılamaz.
  - İsimler 1024 karakterden daha uzun olamaz.

28

C++ ile isimlendirme kuralları

- Örnek değişken isimleri:

Variable Name	Valid or Invalid	If Invalid, Why?
`balance`	Valid	(Not applicable)
`transaction amount`	Invalid	Contains a space
`convert_2_#s`	Invalid	No symbols like #
`MyMoney`	Valid	(Not applicable)
`case`	Invalid	case is a C++ keyword
`Case`	Valid	(Not applicable)
`4_temperature`	Invalid	Cannot start with number
`auto`	Invalid	Cannot be a keyword
`My_auto`	Valid	(Not applicable)

29

Değişken tanımlama

- Değişkenler fonksiyon içinde, dışında veya başlık satırında tanımlanabilir.
- Bir değişken kullanılmadan önce tanımlanmalıdır.
- Bir değişkene tanımlandığı yerden ulaşılır (scope).

30

C++ ta operatörler

- C++ ta operatörler belirli bir işlemi ifade eder.

```
F_temp = 9.0/5.0 * C_temp + 32.0;
```

- Operatörler eşitliğin sağ tarafında kullanılırlar.

31

Klavyeden veri alma

- Klavyeden veri almak için **`cin`** deyimiyle (**>>**) operatörü birlikte kullanılır.
- Program çalışırken **`cin`** deyimini gördüğünde kullanıcıdan giriş yapmasını bekler.
- Kullanıcı giriş yapıp Enter tuşuna basınca girilen veri ilgili değişkene aktarılır.

32

`cin` örnek

- Bir değişkene veri alma.

```
cout << "\n Ortalama hızı giriniz =";

cin >> hiz;
```

- `cin` kullanarak birden fazla değişkene veri alma.

```
cout  <<"\n Haftada araç kullandığınız gün sayısını,"
      <<"\n ortalama hızınızı, ve her gün kaç saat araç
            kullandığınızı giriniz"
      <<"\n örnek giriş 4   68.0   7.5";

cin   >> gunSayisi >> ortalamaHiz >> gunlukKullanimSuresi;
```

33

Atama operatörü

- Atama operatörü (=) sağ taraftaki değeri sol taraftaki değişkene aktarır.

```
miktar = 1534.34;
islemSirasi = 8;
x = y;
```

- Birden fazla değişkene bir ifadeyle değer atanabilir.

```
a = b = c = 0;
```

34

İşlem öncelikleri

- İşlem öncelikleri deyimlerin çalışma şeklini (operatörlerin işlem sırasını) gösterir.

Priority	Operator Type	Operator	Associativity
Highest	Primary	`() [] . ->`	Left to right
	Arithmetic	`* / %`	Left to right
	Arithmetic	`+ -`	Left to right
Lowest	Assignment	`=`	Right to left

35

İşlem öncelikleri

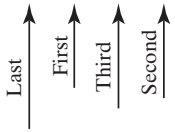
```
F_temp = 9.0/5.0 * C_temp + 32.0;
```

- Yukarıda dört operatör var (=, /, *, +).
- Çarpma (*) ve bölme (/) en yüksek önceliğe sahiptir. Sonra toplama (+) ve en son atama (=) işlemi yapılır.
- Atama hariç işlemlerin tamamı soldan sağa önceliklendirilerek yapılır.
  - Önce 9.0/5.0 bölme işlemi yapılır.
  - Hesaplanan değer `C_temp` değişkeniyle çarpılır.
  - Sonra toplama işlemi yapılır.
  - En son atama operatörüyle hesaplanan değer `F_temp` değişkenine aktarılır.

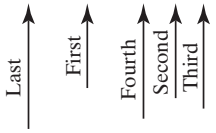
36

İşlem öncelikleri

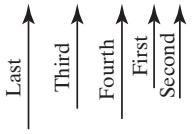
a = b * c + d * f; — 4 operators: = * + *



m = (s + t) + r/p*q; — 5 operators: = () + / *



m = s + t + r/p*q; — 5 operators: = ++ / *



37

Veri türleri ve saklanan değerler

- Veri türü hafızaya saklanacak değeri belirler.

```
double x = 15; // x değeri 15.0000000000000 olur.
```

- Bir integer değişkene değer atandığında tam kısmı saklanır.
- Ondalıklı kısmı yuvarlanmaz truncate (atılır) yapılır.

```
int miktar = 435.83;    // miktar değeri 435 olur.
```

- Programcı truncate yapılacak bir değer atadığında compiler uyarı mesajı üretir.

38

Veri türleri ve saklanan değerler

- Aşağıdaki iki değişkenin değeri truncate yapılır.

```
float pi = 3.141592653589793;     // pi değişkeninin değeri
                                  // 3.141593 olur.

short int toplam = 56332;  // short int limit 32.767 olur.
```

39

C++ değişkenleri initialize etmez

- C++ değişkenlerin başlangıç değerini atamaz.
- Değişkenlerin başlangıç değeri 0 olarak alınamaz.
- Bazı C++ geliştirme ortamları en büyük negatif sayıyı başlangıç değeri alır.
- Değer atanmadan kullanılan değişkenler anlamsız sonuçlar üretir.

40

C++ değişkenleri initialize etmez

- C++ başlangıç değeri atanmadan kullanılan değişkenler için uyarı mesajı verir.

```
yollar.cpp(18) : warning C4700: local variable
'gunSayisi' used without having been initialized

yollar.cpp(18) : warning C4700: local variable
'ortalamaHiz' used without having been initialized

yollar.cpp(21) : warning C4700: local variable
'gunlukKullanim' used without having been initialized
```

- Aşağıdaki gibi bir sonuç oluşabilir.

```
toplamDeger = 858993460
ortalamaNot = -9.25596e+061
```

41

lvalue ve rvalue

- **lvalue** atama operatörünün sol tarafına, **rvalue** sağ tarafına denir.
- Sağ taraf hesaplanan bir değer sol taraf değişken olabilir.

Yanlış atamalar

```
double x, sqrootX;
5.2 = x;            // can't assign from right to left

sqrt(x) = sqrootX; // can't call sqrt on left of =
                   //sign
```

Doğru atamalar

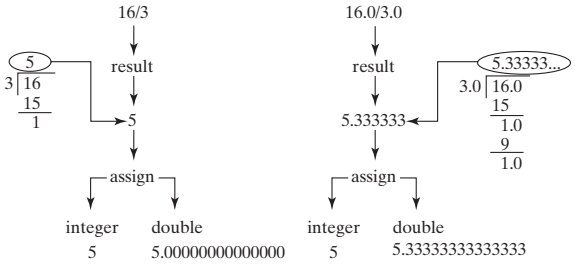
```
double x, sqrootX;
x = 5.2;           //assign number from right to left

sqrootX = sqrt(x);  //calculate on right, assign to left
```

42

Veri türleri ve aritmetik işlem sonuçları

Operand Data Types	Result Data Type	Example	Result
Both integer	int	5*4	20
		16/3	5
		7 + 8	15
		8 – 2	6
		17 % 5	2
int float	double	5*4.0	20.00000000000000
int double			
Both float/double	double	5.0*4.0	20.00000000000000



Integer ve floating-point bölme işlemlerinin sonuçları

43

Cebirsel eşitlikler ve C++ ifadeleri

Algebra	C++: The Right Way	C++: The Wrong Way
$v = (a + b)(c - d)$	`double v,a,b,c,d;`	`double v,a,b,c,d;`
	`v = (a + b) * (c - d);`	`v = (a + b)(c - d); //Note 1`
$SA = \text{pi} \cdot \text{radius}^2$	`double SA, pi, rad;`	`double SA, pi, rad;`
	`SA = pi * rad * rad;`	`SA = (pi)(rad)(rad); //Note 1`
	`//OR See Note 2.`	`//OR`
	`SA = pi * pow(rad, 2);`	`SA = pi * rad **2; // Note 3`
		`SA = pi * rad ^2; // See Note 3`
$a = \frac{c + b}{x - y}$	`double a, b, c, x, y;`	`double a, b, c, x, y;`
	`a = (c+b)/(x-y);`	`a = c + b/x - y;  //Note 4`
$m = \frac{\sqrt{x \cdot 3y}}{w}$	`double x, y, w, m;`	`double x, y, w, m;`
	`m = sqrt(x * 3*y)/w; //Note 2`	`m = sqrt(x.3y)/w;  //Note 5`
$f(x) = \frac{2}{3}\sin(x - 0.3)$	`double x, fofx;`	`double x, f(x);`
	`fofx=2.0/3.0*sin(x-0.3);`	`f(x)=2/3sin(x-0.3);  //Note 6`

Note 1: To perform multiplication, the * operator must be used. The *()()* does not mean multiplication in C++.
Note 2: The *cmath* library needs to be included to use the square root and power functions.
Note 3: The rad**2 or rad ^2 are not valid ways to do exponentiation in C/C++.
Note 4: Division has higher priority, b/x would be done first.
Note 5: The dot operator (.) is not multiplication in C++.
Note 6: Cannot name variables *f(x)*.

44

Artırma ve azaltma operatörleri

- ++ ve -- operatörleri hızlı bir şekilde değişkenin değerini 1 artırır veya azaltır.

```
++i;   veya      i++;
```

aşağıdakine eşittir

```
i = i + 1;
```

45

Artırma ve azaltma operatörleri

- Toplama işlemi olarak postfix ve prefix arasında fark yoktur.

```
++i;  //prefix operator ++ değişkenden önce gelir
i++;  //postfix operator ++ değişkenden sonra gelir
```

- Prefix operatör önce artırır / azaltır sonra atama yapar.
- Postfix operatör önce atama yapar sonra artırır / azaltır.

46

Artırma ve azaltma operatörleri

- ***i*** değişkeninin başlangıç değeri 5'tir.

Operator	Job	Format	Equivalent To	Start i = 5, then ...
Prefix increment	Add *1* to *i* then assign *i* into *m*.	`m = ++i;`	`i = i + 1;` `m = i;`	`i = 6` `m = 6`
Postfix increment	Assign *i* into *m* and then add *1* to *i*.	`m = i++;`	`m = i;` `i = i + 1;`	`m = 5` `i = 6`
Prefix decrement	Subtract *1* from *i* and then assign *i* into *m*.	`m = --i;`	`i = i - 1;` `m = i;`	`i = 4` `m = 4`
Postfix decrement	Assign *i* into *m* and then subtract *1* from *i*.	`m = i--;`	`m = i;` `i = i - 1;`	`m = 5` `i = 4`

47

Accumulation operatörleri

- Accumulation operatörleri `(+=, -=, *=, /=)` atama işlemlerini kısa bir şekilde yazmak için kullanılır.

```
toplam = toplam + x;
toplam += x;

fark = fark - x;
fark -= x;
```

48

`#define` önişlemci komutu

- **`#define`** sembolik sabit tanımlamak için kullanılır.
- **`#define`** önişlemci komutu sonuna `;` konulmaz.

```
#define symbolic_name character_sequence
```

Örnek: **`#define PI 3.14159265`**

`#define` önişlemci komutu

- **`#define`** önişlemci komutu dosyanın başına yazılır.
- **`#define`** önişlemci komutu ile yapılan tanımlamalarda genellikle tümü büyük harf kullanılır.

```
#define MAKSIMUM 100
#define DOSYAADI  "F:\data\input.dat"
#define ORAN 8.50
```

`#define` önişlemci komutu

- PI için `#define` kullanımı

```
#include <iostream>
using namespace std;
#define PI 3.14159265

int main()
{
   double circleArea, radius = 5.0;
   circleArea = PI * radius * radius;
   cout << "\n Circle area =" << circleArea;
   return 0;
}
```

51

`const`

- `const` değişken tanımlamalarında kullanılır.
- `const` ile tanımlanan değer programın sonuna kadar değişmez.

```
const data_type variable_name = initial_value;
```

Örnek: `const double x = 7.1111;`

52

const

- PI için **const** kullanımı

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

int main()
{
   const double PI = 3.14159265;
   double circleArea, radius = 5.0;
   circleArea = PI * radius * radius;
   cout << "\n Circle area" << circleArea;
   return 0;
}
```

53

Klavye girişi ve ekran çıkışı format karakterleri

- Klavye girişi ve ekran çıkışında yaygın kullanılan karakterler.

\a	beep
\n	newline
\f	formfeed
\t	tab
\b	backspace
\\	backslash

54

setw()

- **<iomanip>** kütüphanesinde bulunur.
- Hemen ardından gelen değişkene ayrılacak alanın boyutunu belirler.
- Aşağıda **x** değişkeninin yazdırılması için 5 karakterlik alan ayrılır.

```
cout << "\n x değeri = " << setw(5) << x;
```