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

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Konular

- Giriş
- math Kütüphane Fonksiyonları
- Çok Parametreyle Fonksiyon Tanımı
- Fonksiyon Prototipleri
- C++ Standart Kütüphaneleri
- Örnek: Rastgele Sayı Üreteci
- Scope Kuralları
- Unary Scope Resolution Operator
- Call-by-Value ve Call-by-Reference
- Default Parametreler
- Function Overloading
- Özyineleme (Recursion)

Giriş

- Büyük programları geliştirmenin en iyi yolu, programı küçük ve basit bileşenlerle yapılandırmaktır.
- Her bileşenin geliştirilmesi için ayrı ayrı fonksiyon kullanılmalıdır.
- Fonksiyonlar girişler alıp çıkışlar oluşturur.
- C++ kullanıcılara matematiksel hesaplamalar, string işlemleri, karakter işlemleri, giriş/çıkış işlemleri için çok sayıda hazır fonksiyon sunar.
- Fonksiyonlar programlarda reusability artırır.

3

Konular

- Giriş
- math Kütüphane Fonksiyonları
- Çok Parametreyle Fonksiyon Tanımı
- Fonksiyon Prototipleri
- C++ Standart Kütüphaneleri
- Örnek: Rastgele Sayı Üreteci
- Scope Kuralları
- Unary Scope Resolution Operator
- Call-by-Value ve Call-by-Reference
- Default Parametreler
- Function Overloading
- Özyineleme (Recursion)

math Kütüphane Fonksiyonları

- **<cmath>** kütüphanesi çok sayıda hazır fonksiyonu kullanıcılara sunar.

Function	Description	Example
ceil(x)	rounds x to the smallest integer not less than x	ceil(9.2) is 10.0 ceil(-9.8) is -9.0
cos(x)	trigonometric cosine of x (x in radians)	cos(0.0) is 1.0
exp(x)	exponential function e^x	exp(1.0) is 2.71828 exp(2.0) is 7.38906

5

math Kütüphane Fonksiyonları

fabs(x)	absolute value of x	fabs(5.1) is 5.1 fabs(0.0) is 0.0 fabs(-8.76) is 8.76
floor(x)	rounds x to the largest integer not greater than x	floor(9.2) is 9.0 floor(-9.8) is -10.0
fmod(x, y)	remainder of x/y as a floating-point number	fmod(2.6, 1.2) is 0.2
log(x)	natural logarithm of x (base e)	log(2.718282) is 1.0 log(7.389056) is 2.0
log10(x)	logarithm of x (base 10)	log10(10.0) is 1.0 log10(100.0) is 2.0
pow(x, y)	x raised to power y (x^y)	pow(2, 7) is 128 pow(9, .5) is 3
sin(x)	trigonometric sine of x (x in radians)	sin(0.0) is 0
sqrt(x)	square root of x (where x is a nonnegative value)	sqrt(9.0) is 3.0
tan(x)	trigonometric tangent of x (x in radians)	tan(0.0) is 0

6

Konular

- Giriş
- math Kütüphane Fonksiyonları
- Çok Parametreyle Fonksiyon Tanımı
- Fonksiyon Prototipleri
- C++ Standart Kütüphaneleri
- Örnek: Rastgele Sayı Üreteci
- Scope Kuralları
- Unary Scope Resolution Operator
- Call-by-Value ve Call-by-Reference
- Default Parametreler
- Function Overloading
- Özyineleme (Recursion)

Çok Parametreyle Fonksiyon Tanımı

- Fonksiyon için örnek.

```
1  // Fig. 6.3: GradeBook.h
2  // Definition of class GradeBook that finds the maximum of three grades.
3  // Member functions are defined in GradeBook.cpp
4  #include <string> // program uses C++ standard string class
5  using std::string;
6
7  // GradeBook class definition
8  class GradeBook
9  {
10 public:
11    GradeBook( string ); // constructor initializes course name
12    void setCourseName( string ); // function to set the course name
13    string getCourseName(); // function to retrieve the course name
14    void displayMessage(); // display a welcome message
15    void inputGrades(); // input three grades from user
16    void displayGradeReport(); // display a report based on the grades
17    int maximum( int, int, int ); // determine max of 3 values
18 private:
19    string courseName; // course name for this GradeBook
20    int maximumGrade; // maximum of three grades
21 }; // end class GradeBook
22
23
```

8

Çok Parametreyle Fonksiyon Tanımı

```
1  // Fig. 6.4: GradeBook.cpp
2  // Member-function definitions for class GradeBook that
3  // determines the maximum of three grades.
4  #include <iostream>
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  #include "GradeBook.h" // include definition of class GradeBook
10
11 // constructor initializes courseName with string supplied as argument;
12 // initializes maximumGrade to 0
13 GradeBook::GradeBook( string name )
14 {
15    setCourseName( name ); // validate and store courseName
16    maximumGrade = 0; // this value will be replaced by the maximum grade
17 } // end GradeBook constructor
18
19 // function to set the course name; limits name to 25 or fewer characters
20 void GradeBook::setCourseName( string name )
21 {
22    if ( name.length() <= 25 ) // if name has 25 or fewer characters
23       courseName = name; // store the course name in the object
24    else // if name is longer than 25 characters
25    { // set courseName to first 25 characters of parameter name
26       courseName = name.substr( 0, 25 ); // select first 25 characters
27       cout << "Name \"" << name << "\" exceeds maximum length (25).\n"
28          << "Limiting courseName to first 25 characters.\n" << endl;
29    } // end if...else
30 } // end function setCourseName
31
```

9

Çok Parametreyle Fonksiyon Tanımı

```
32 // function to retrieve the course name
33 string GradeBook::getCourseName()
34 {
35    return courseName;
36 } // end function getCourseName
37
38 // display a welcome message to the GradeBook user
39 void GradeBook::displayMessage()
40 {
41    // this statement calls getCourseName to get the
42    // name of the course this GradeBook represents
43    cout << "Welcome to the grade book for\n" << getCourseName() << "!\n"
44       << endl;
45 } // end function displayMessage
46
47 // input three grades from user; determine maximum
48 void GradeBook::inputGrades()
49 {
50    int grade1; // first grade entered by user
51    int grade2; // second grade entered by user
52    int grade3; // third grade entered by user
53
54    cout << "Enter three integer grades: ";
55    cin >> grade1 >> grade2 >> grade3;
56
57    // store maximum in member studentMaximum
58    maximumGrade = maximum( grade1, grade2, grade3 );
59 } // end function inputGrades
60
```

10

Çok Parametreyle Fonksiyon Tanımı

```
61  // returns the maximum of its three integer parameters
62  int GradeBook::maximum( int x, int y, int z )
63  {
64     int maximumValue = x; // assume x is the largest to start
65
66     // determine whether y is greater than maximumValue
67     if ( y > maximumValue )
68        maximumValue = y; // make y the new maximumValue
69
70     // determine whether z is greater than maximumValue
71     if ( z > maximumValue )
72        maximumValue = z; // make z the new maximumValue
73
74     return maximumValue;
75  } // end function maximum
76
77  // display a report based on the grades entered by user
78  void GradeBook::displayGradeReport()
79  {
80     // output maximum of grades entered
81     cout << "Maximum of grades entered: " << maximumGrade << endl;
82  } // end function displayGradeReport
83
84
85
```


Çok Parametreyle Fonksiyon Tanımı

```
1  // Fig. 6.5: fig06_05.cpp
2  // Create GradeBook object, input grades and display grade report.
3  #include "GradeBook.h" // include definition of class GradeBook
4
5  int main()
6  {
7     // create GradeBook object
8     GradeBook myGradeBook( "CS101 C++ Programming" );
9
10    myGradeBook.displayMessage(); // display welcome message
11    myGradeBook.inputGrades(); // read grades from user
12    myGradeBook.displayGradeReport(); // display report based on grades
13    return 0; // indicate successful termination
14 } // end main
15
16
```

```
Welcome to the grade book for
CS101 C++ Programming!

Enter three integer grades: 86 67 75
Maximum of grades entered: 86
```

Konular

- Giriş
- math Kütüphane Fonksiyonları
- Çok Parametreyle Fonksiyon Tanımı
- Fonksiyon Prototipleri
- C++ Standart Kütüphaneleri
- Örnek: Rastgele Sayı Üreteci
- Scope Kuralları
- Unary Scope Resolution Operator
- Call-by-Value ve Call-by-Reference
- Default Parametreler
- Function Overloading
- Özyineleme (Recursion)

Fonksiyon Prototipleri

- Fonksiyon prototipleri derleyiciye fonksiyonun adını, geri döndürdüğü veri türünü, fonksiyonun aldığı değişken sayısını ve herbirinin türünü bildirir.

```
int  maximum( int, int, int );

cout << maximum( 6, 9, 0 );
```

- Fonksiyonun implementation kısmında parametreler aynı sırada ve aynı türde kullanılmalıdır.
- Tür değişimlerinin type casting ile yapılması gerekir.
- Tür değişimlerinde aşağıdaki temel türü yukarıdaki temel türe dönüştürürken veri kaybı olur.
- Yukarıdaki temel türü aşağıdaki türe dönüştürmek için type casting yapılmalıdır.

14

Fonksiyon Prototipleri

Data types	
`long double`	
`double`	
`float`	
`unsigned long int`	(synonymous with `unsigned long`)
`long int`	(synonymous with `long`)
`unsigned int`	(synonymous with `unsigned`)
`int`	
`unsigned short int`	(synonymous with `unsigned short`)
`short int`	(synonymous with `short`)
`unsigned char`	
`char`	
`bool`	

15

Konular

- Giriş
- math Kütüphane Fonksiyonları
- Çok Parametreyle Fonksiyon Tanımı
- Fonksiyon Prototipleri
- C++ Standart Kütüphaneleri
- Örnek: Rastgele Sayı Üreteci
- Scope Kuralları
- Unary Scope Resolution Operator
- Call-by-Value ve Call-by-Reference
- Default Parametreler
- Function Overloading
- Özyineleme (Recursion)

C++ Standart Kütüphaneleri

- C++ çok sayıda standart kütüphane sağlamaktadır.
- Her bir kütüphane kendi içinde bir grup sınıf ve fonksiyona sahiptir.

C++ Standard Library header file	Explanation
`<iostream>`	Contains function prototypes for the C++ standard input and standard output functions, introduced in Chapter 2, and is covered in more detail in Chapter 15, Stream Input/Output. This header file replaces header file `<iostream.h>`.
`<iomanip>`	Contains function prototypes for stream manipulators that format streams of data. This header file is first used in Section 4.9 and is discussed in more detail in Chapter 15, Stream Input/Output. This header file replaces header file `<iomanip.h>`.
`<cmath>`	Contains function prototypes for math library functions (discussed in Section 6.3). This header file replaces header file `<math.h>`.
`<cstdlib>`	Contains function prototypes for conversions of numbers to text, text to numbers, memory allocation, random numbers and various other utility functions. Portions of the header file are covered in Section 6.7; Chapter 11, Operator Overloading; String and Array Objects; Chapter 16, Exception Handling; Chapter 19, Web Programming; Chapter 22, Bits, Characters, C-Strings and `structs`; and Appendix E, C Legacy Code Topics. This header file replaces header file `<stdlib.h>`.
`<ctime>`	Contains function prototypes and types for manipulating the time and date. This header file replaces header file `<time.h>`. This header file is used in Section 6.7.

17

C++ Standart Kütüphaneleri

`<vector>`, `<list>`, `<deque>`, `<queue>`, `<stack>`, `<map>`, `<set>`, `<bitset>`	These header files contain classes that implement the C++ Standard Library containers. Containers store data during a program's execution. The `<vector>` header is first introduced in Chapter 7, Arrays and Vectors. We discuss all these header files in Chapter 23, Standard Template Library (STL).
`<cctype>`	Contains function prototypes for functions that test characters for certain properties (such as whether the character is a digit or a punctuation), and function prototypes for functions that can be used to convert lowercase letters to uppercase letters and vice versa. This header file replaces header file `<ctype.h>`. These topics are discussed in Chapter 8, Pointers and Pointer-Based Strings, and Chapter 22, Bits, Characters, C-Strings and `structs`.
`<cstring>`	Contains function prototypes for C-style string-processing functions. This header file replaces header file `<string.h>`. This header file is used in Chapter 11, Operator Overloading; String and Array Objects.
`<typeinfo>`	Contains classes for runtime type identification (determining data types at execution time). This header file is discussed in Section 13.8.

18

C++ Standart Kütüphaneleri

`<exception>`, `<stdexcept>`	These header files contain classes that are used for exception handling (discussed in Chapter 16).
`<memory>`	Contains classes and functions used by the C++ Standard Library to allocate memory to the C++ Standard Library containers. This header is used in Chapter 16, Exception Handling.
`<fstream>`	Contains function prototypes for functions that perform input from files on disk and output to files on disk (discussed in Chapter 17, File Processing). This header file replaces header file `<fstream.h>`.
`<string>`	Contains the definition of class `string` from the C++ Standard Library (discussed in Chapter 18).
`<sstream>`	Contains function prototypes for functions that perform input from strings in memory and output to strings in memory (discussed in Chapter 18, Class `string` and String Stream Processing).

19

C++ Standart Kütüphaneleri

`<functional>`	Contains classes and functions used by C++ Standard Library algorithms. This header file is used in Chapter 23.
`<iterator>`	Contains classes for accessing data in the C++ Standard Library containers. This header file is used in Chapter 23, Standard Template Library (STL).
`<algorithm>`	Contains functions for manipulating data in C++ Standard Library containers. This header file is used in Chapter 23.
`<cassert>`	Contains macros for adding diagnostics that aid program debugging. This replaces header file `<assert.h>` from pre-standard C++. This header file is used in Appendix F, Preprocessor.
`<cfloat>`	Contains the floating-point size limits of the system. This header file replaces header file `<float.h>`.
`<climits>`	Contains the integral size limits of the system. This header file replaces header file `<limits.h>`.
`<cstdio>`	Contains function prototypes for the C-style standard input/output library functions and information used by them. This header file replaces header file `<stdio.h>`.
`<locale>`	Contains classes and functions normally used by stream processing to process data in the natural form for different languages (e.g., monetary formats, sorting strings, character presentation, etc.).
`<limits>`	Contains classes for defining the numerical data type limits on each computer platform.
`<utility>`	Contains classes and functions that are used by many C++ Standard Library header files.

20

Konular

- Giriş
- math Kütüphane Fonksiyonları
- Çok Parametreyle Fonksiyon Tanımı
- Fonksiyon Prototipleri
- C++ Standart Kütüphaneleri
- Örnek: Rastgele Sayı Üreteci
- Scope Kuralları
- Unary Scope Resolution Operator
- Call-by-Value ve Call-by-Reference
- Default Parametreler
- Function Overloading
- Özyineleme (Recursion)

Örnek: Rastgele Sayı Üreteci

- **`<cstdlib>`** kütüphanesi içerisindeki **`rand()`** fonksiyonu 0-**`RAND_MAX`** arasında rastgele bir sayı üretir.
- **`RAND_MAX`** değeri GNU C++ için 214748647, Visual Studio için 32767'dir. Her bir kütüphane kendi içinde bir grup sınıf ve fonksiyona sahiptir.
- **`rand()`** fonksiyonun üreteceği sayı için aralıktaki tüm sayılar eşit olasılığa sahiptir.

22

Örnek: Rastgele Sayı Üreteci

```
// Fig. 6.8: fig06_08.cpp
// Shifted and scaled random integers.
#include <iostream>
using std::cout;
using std::endl;

#include <iomanip>
using std::setw;

#include <cstdlib> // contains function prototype for rand
using std::rand;

int main()
{
   // loop 20 times
   for ( int counter = 1; counter <= 20; counter++ )
   {
      // pick random number from 1 to 6 and output it
      cout << setw( 10 ) << ( 1 + rand() % 6 );

      // if counter is divisible by 5, start a new line of output
      if ( counter % 5 == 0 )
         cout << endl;
   } // end for

   return 0; // indicates successful termination
} // end main
```

```
     6     6     5     5     6
     5     1     1     5     3
     6     6     2     4     2
     6     2     3     4     1
```


Örnek: Rastgele Sayı Üreteci

- 6.000.000 kez zar atma sonucunda sayıların sıklıkları.

```
// Fig. 6.9: fig06_09.cpp
// Roll a six-sided die 6,000,000 times.
#include <iostream>
using std::cout;
using std::endl;

#include <iomanip>
using std::setw;

#include <cstdlib> // contains function prototype for rand
using std::rand;

int main()
{
   int frequency1 = 0; // count of 1s rolled
   int frequency2 = 0; // count of 2s rolled
   int frequency3 = 0; // count of 3s rolled
   int frequency4 = 0; // count of 4s rolled
   int frequency5 = 0; // count of 5s rolled
   int frequency6 = 0; // count of 6s rolled

   int face; // stores most recently rolled value
```

Örnek: Rastgele Sayı Üreteci

- 6.000.000 kez zar atma sonucunda sayıların sıklıkları.

```
24    // summarize results of 6,000,000 rolls of a die
25    for ( int roll = 1; roll <= 6000000; roll++ )
26    {
27       face = 1 + rand() % 6; // random number from 1 to 6
28
29       // determine roll value 1-6 and increment appropriate counter
30       switch ( face )
31       {
32          case 1:
33             ++frequency1; // increment the 1s counter
34             break;
35          case 2:
36             ++frequency2; // increment the 2s counter
37             break;
38          case 3:
39             ++frequency3; // increment the 3s counter
40             break;
41          case 4:
42             ++frequency4; // increment the 4s counter
43             break;
44          case 5:
45             ++frequency5; // increment the 5s counter
46             break;
47          case 6:
48             ++frequency6; // increment the 6s counter
49             break;
50          default: // invalid value
51             cout << "Program should never get here!";
52       } // end switch
53    } // end for
```


Örnek: Rastgele Sayı Üreteci

- 6.000.000 kez zar atma sonucunda sayıların sıklıkları.

```
54
55    cout << "Face" << setw( 13 ) << "Frequency" << endl; // output headers
56    cout << "   1" << setw( 13 ) << frequency1
57       << "\n   2" << setw( 13 ) << frequency2
58       << "\n   3" << setw( 13 ) << frequency3
59       << "\n   4" << setw( 13 ) << frequency4
60       << "\n   5" << setw( 13 ) << frequency5
61       << "\n   6" << setw( 13 ) << frequency6 << endl;
62    return 0; // indicates successful termination
63 } // end main
```

- Program tekrar çalıştırıldığında aynı sonucu oluşturacaktır.

Face	Frequency
1	999702
2	1000823
3	999378
4	998898
5	1000777
6	1000422

Face	Frequency
1	999702
2	1000823
3	999378
4	998898
5	1000777
6	1000422

Örnek: Rastgele Sayı Üreteci

- **`srand()`** fonksiyonu bir işaretsiz integer değer alır ve **`rand()`** fonksiyonunun farklı sonuçlar üretmesini sağlar.
- **`srand()`** fonksiyonuna sürekli yeni değer girilmesi genellikle bilgisayarın zaman fonksiyonu ile yapılır.

```
srand( time( 0 ) );
```

- Random üretilen sayı shift edilebilir ve ölçeklenebilir.

```
number = shiftingValue + rand() % scalingFactor;
```

```
face = 1 + rand() % 6;
```

27

Örnek: Rastgele Sayı Üreteci

```
// Fig. 6.10: fig06_10.cpp
// Randomizing die-rolling program.
#include <iostream>
using std::cout;
using std::cin;
using std::endl;

#include <iomanip>
using std::setw;

#include <cstdlib> // contains prototypes for functions srand and rand
using std::rand;
using std::srand;

int main()
{
   unsigned seed; // stores the seed entered by the user

   for (int j = 1; j <= 2; j++)
   {
      cout << "\nEnter seed: ";
      cin >> seed;
      srand( seed ); // seed random number generator

      // loop 10 times
      for ( int counter = 1; counter <= 10; counter++ )
      {
         // pick random number from 1 to 6 and output it
         cout << setw( 10 ) << ( 1 + rand() % 6 );

         // if counter is divisible by 5, start a new line of output
         if ( counter % 5 == 0 )
            cout << endl;
      } // end for
   }
   system("PAUSE");
   return 0; // indicates successful termination
} // end main
```

```
c:\Documents and Settings\M...ALI\My Documents\Visual Studio 2

Enter seed: 10
         6         4         3         3         6
         3         3         1         3         4

Enter seed: 20
         2         4         2         4         5
         4         3         5         1         4
Press any key to continue . . .
```

28

Konular

- Giriş
- math Kütüphane Fonksiyonları
- Çok Parametreyle Fonksiyon Tanımı
- Fonksiyon Prototipleri
- C++ Standart Kütüphaneleri
- Örnek: Rastgele Sayı Üreteci
- Scope Kuralları
- Unary Scope Resolution Operator
- Call-by-Value ve Call-by-Reference
- Default Parametreler
- Function Overloading
- Özyineleme (Recursion)

Scope Kuralları

- Her değişken tanımlandığı blok (scope) içerisinde geçerlidir.
- C++ ile function scope, file scope, block scope, function-prototype scope, class scope ve namespace scope oluşturulabilir.
- Aynı değişken adı farklı seviyelerdeki scope'larda kullanılabilir.
- Bir değişken, default olarak kendisine en yakın scope kullanılarak belirlenir.
- Bir üst seviyedeki değişken adına :: operatörü ile ulaşılabilir.

30

Scope Kuralları

```
// Fig. 6.12: fig06_12.cpp
// A scoping example.
#include <iostream>
using std::cout;
using std::endl;

void useLocal( void ); // function prototype
void useStaticLocal( void ); // function prototype
void useGlobal( void ); // function prototype

int x = 1; // global variable

int main()
{
   int x = 5; // local variable to main

   cout << "local x in main's outer scope is " << x << endl;

   { // start new scope
      int x = 7; // hides x in outer scope

      cout << "local x in main's inner scope is " << x << endl;
   } // end new scope

   cout << "local x in main's outer scope is " << x << endl;

   useLocal(); // useLocal has local x
   useStaticLocal(); // useStaticLocal has static local x
   useGlobal(); // useGlobal uses global x
   useLocal(); // useLocal reinitializes its local x
   useStaticLocal(); // static local x retains its prior value
   useGlobal(); // global x also retains its value

   cout << "\nlocal x in main is " << x << endl;
   system("PAUSE");
   return 0; // indicates successful termination
} // end main
```

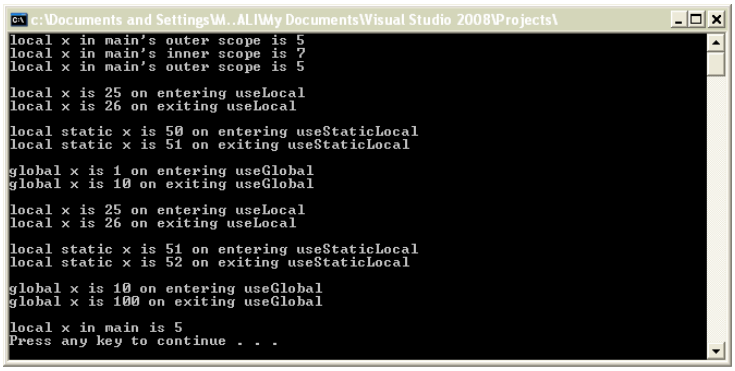
31

Scope Kuralları

```
// useLocal reinitializes local variable x during each call
void useLocal( void )
{
   int x = 25; // initialized each time useLocal is called

   cout << "\nlocal x is " << x << " on entering useLocal" << endl;
   x++;
   cout << "local x is " << x << " on exiting useLocal" << endl;
} // end function useLocal

// useStaticLocal initializes static local variable x only the
// first time the function is called; value of x is saved
// between calls to this function
void useStaticLocal( void )
{
   static int x = 50; // initialized first time useStaticLocal is called

   cout << "\nlocal static x is " << x << " on entering useStaticLocal"
      << endl;
   x++;
   cout << "local static x is " << x << " on exiting useStaticLocal"
      << endl;
} // end function useStaticLocal

// useGlobal modifies global variable x during each call
void useGlobal( void )
{
   cout << "\nglobal x is " << x << " on entering useGlobal" << endl;
   x *= 10;
   cout << "global x is " << x << " on exiting useGlobal" << endl;
} // end function useGlobal
```

32

Scope Kuralları



33

Konular

- Giriş
- math Kütüphane Fonksiyonları
- Çok Parametreyle Fonksiyon Tanımı
- Fonksiyon Prototipleri
- C++ Standart Kütüphaneleri
- Örnek: Rastgele Sayı Üreteci
- Scope Kuralları
- Unary Scope Resolution Operator
- Call-by-Value ve Call-by-Reference
- Default Parametreler
- Function Overloading
- Özyineleme (Recursion)

Unary Scope Resolution Operator

- Unary scope resolution operator (::) aynı isimdeki global değişkene ulaşımı sağlar.

```
1  // Fig. 6.23: fig06_23.cpp
2  // Using the unary scope resolution operator.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  int number = 7; // global variable named number
8
9  int main()
10 {
11    double number = 10.5; // local variable named number
12
13    // display values of local and global variables
14    cout << "Local double value of number = " << number
15       << "\nGlobal int value of number = " << ::number << endl;
16    return 0; // indicates successful termination
17 } // end main
```

```
c:\Documents and Settings\M...ALI\My Documents\Visu
Local double value of number = 10.5
Global int value of number = 7
Press any key to continue . . .
```

35

Konular

- Giriş
- math Kütüphane Fonksiyonları
- Çok Parametreyle Fonksiyon Tanımı
- Fonksiyon Prototipleri
- C++ Standart Kütüphaneleri
- Örnek: Rastgele Sayı Üreteci
- Scope Kuralları
- Unary Scope Resolution Operator
- Call-by-Value ve Call-by-Reference
- Default Parametreler
- Function Overloading
- Özyineleme (Recursion)

Call-by-Value ve Call-by-Reference

- Bir fonksiyon aldığı parametrelerin değişen değerlerini aynı değişkenle geri gönderebilir.
- **Call-by-reference**, fonksiyona gönderilen parametrelerin fonksiyondan çıkarken son değerlerinin geri gönderilmesini sağlar.
- **Call-by-value**, fonksiyona gönderilen parametrelerin değerlerini geri göndermez.
- Call-by-reference şeklinde parametre gönderildiğinde değişkenin eski değeri değişir.
- Call-by-value, default çalışma şeklidir.
- `const` ile yapılan tanımlamalarda değişkenin değeri değişmez.

37

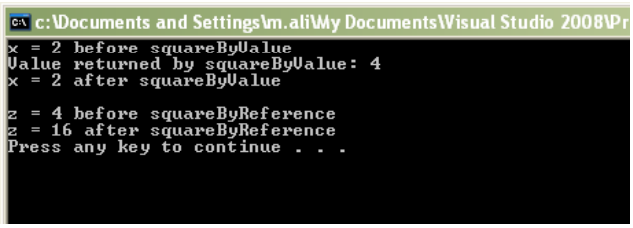
Call-by-Value ve Call-by-Reference

```
1  // Fig. 6.19: fig06_19.cpp
2  // Comparing pass-by-value and pass-by-reference with references.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  int squareByValue( int ); // function prototype (value pass)
8
9  void squareByReference( int & ); // function prototype (reference pass)
10
11 int main()
12 {
13     int x = 2; // value to square using squareByValue
14     int z = 4; // value to square using squareByReference
15
16     // demonstrate squareByValue
17     cout << "x = " << x << " before squareByValue\n";
18     cout << "Value returned by squareByValue: "
19          << squareByValue( x ) << endl;
20     cout << "x = " << x << " after squareByValue\n" << endl;
21
22     // demonstrate squareByReference
23     cout << "z = " << z << " before squareByReference" << endl;
24
25     squareByReference( z );
26     cout << "z = " << z << " after squareByReference" << endl;
27     system("PAUSE");
28     return 0; // indicates successful termination
29 } // end main
30
```

38

Call-by-Value ve Call-by-Reference

```
31 // squareByValue multiplies number by itself, stores the
32 // result in number and returns the new value of number
33 int squareByValue( int number )
34 {
35     return number *= number; // caller's argument not modified
36 } // end function squareByValue
37
38 // squareByReference multiplies numberRef by itself and stores the result
39 // in the variable to which numberRef refers in function main
40 void squareByReference( int &numberRef )
41 {
42     numberRef *= numberRef; // caller's argument modified
43 } // end function squareByReference
```



39

Call-by-Value ve Call-by-Reference

- Bir değişken başka bir değişkenle referans gösterilebilir.
- Bir değişkenin referansı & işareti ile gösterilir ve değişkenin adresini saklar.
- Referans göstermek için kullanılan değişkenin başlangıç değerinin atanması gereklidir.
- Aşağıdaki örnekte **`&cRef`** değişkeni count değişkenini referans gösterir.
- **`&cRef`** değişkenin değeri değiştiğinde count değişkeninin de değeri değişir.

```
int count = 1; // declare integer variable count
int &cRef = count; // create cRef as an alias for count
cRef++; // increment count (using its alias cRef)
```

40

Call-by-Value ve Call-by-Reference

```
1  // Fig. 6.20: fig06_20.cpp
2  // References must be initialized.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  int main()
8  {
9      int x = 3;
10     int &y = x; // y refers to (is an alias for) x
11
12     cout << "x = " << x << endl << "y = " << y << endl;
13     y = 7; // actually modifies x
14     cout << "x = " << x << endl << "y = " << y << endl;
15
16     system("PAUSE");
17     return 0; // indicates successful termination
18 } // end main
```

41

Call-by-Value ve Call-by-Reference

- Başlangıç değeri atanmayan referans değişkeni için hata oluşur.

```
1  // Fig. 6.20: fig06_20.cpp
2  // References must be initialized.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  int main()
8  {
9      int x = 3;
10     int &y; // Error: y must be initialized
11
12     cout << "x = " << x << endl << "y = " << y << endl;
13     y = 7; // actually modifies x
14     cout << "x = " << x << endl << "y = " << y << endl;
15
16     system("PAUSE");
17     return 0; // indicates successful termination
18 } // end main
```

```
deneme.cpp(10) : error C2530: 'y' : references must be initialized
```

42

Konular

- Giriş
- math Kütüphane Fonksiyonları
- Çok Parametreyle Fonksiyon Tanımı
- Fonksiyon Prototipleri
- C++ Standart Kütüphaneleri
- Örnek: Rastgele Sayı Üreteci
- Scope Kuralları
- Unary Scope Resolution Operator
- Call-by-Value ve Call-by-Reference
- Default Parametreler
- Function Overloading
- Özyineleme (Recursion)

Default Parametreler

- Fonksiyonlarda tanımlanan parametrelere başlangıç değeri atanabilir.
- Başlangıç değerleri fonksiyonun prototip tanımlaması yapılırken verilir.
- Birden fazla parametre kullanılırsa, fonksiyon çağırıldığında default değer yerine verilen değerler soldaki parametreden başlanarak aktarılır.

44

Default Parametreler

```
// Fig. 6.22: fig06_22.cpp
// Using default arguments.
#include <iostream>
using std::cout;
using std::endl;

// function prototype that specifies default arguments
int boxVolume( int length = 1, int width = 1, int height = 1 );

int main()
{
   // no arguments--use default values for all dimensions
   cout << "The default box volume is: " << boxVolume();

   // specify length; default width and height
   cout << "\n\nThe volume of a box with length 10,\n"
      << "width 1 and height 1 is: " << boxVolume( 10 );

   // specify length and width; default height
   cout << "\n\nThe volume of a box with length 10,\n"
      << "width 5 and height 1 is: " << boxVolume( 10, 5 );

   // specify all arguments
   cout << "\n\nThe volume of a box with length 10,\n"
      << "width 5 and height 2 is: " << boxVolume( 10, 5, 2 )
      << endl;
   system("PAUSE");
   return 0; // indicates successful termination
} // end main

// function boxVolume calculates the volume of a box
int boxVolume( int length, int width, int height )
{
   return length * width * height;
} // end function boxVolume
```

```
c:\Documents and Settings\M..ALI\My Documents\
The default box volume is: 1

The volume of a box with length 10,
width 1 and height 1 is: 10

The volume of a box with length 10,
width 5 and height 1 is: 50

The volume of a box with length 10,
width 5 and height 2 is: 100
Press any key to continue . . .
```

45

Konular

- Giriş
- math Kütüphane Fonksiyonları
- Çok Parametreyle Fonksiyon Tanımı
- Fonksiyon Prototipleri
- C++ Standart Kütüphaneleri
- Örnek: Rastgele Sayı Üreteci
- Scope Kuralları
- Unary Scope Resolution Operator
- Call-by-Value ve Call-by-Reference
- Default Parametreler
- Function Overloading
- Özyineleme (Recursion)

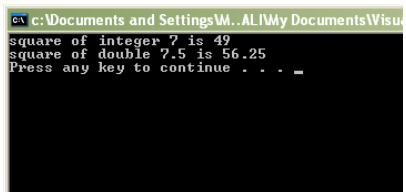
Function Overloading

- C++ aynı isimde farklı parametreler alan ve farklı sonuçlar geri döndüren fonksiyonlar tanımlamaya olanak verir.
- **Function overloading** farklı parametrelere sahip olan aynı isimde birden fazla fonksiyon tanımlamayı ifade eder.

47

Function Overloading

```
1  // Fig. 6.24: fig06_24.cpp
2  // Overloaded functions.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  // function square for int values
8  int square( int x )
9  {
10    cout << "square of integer " << x << " is ";
11    return x * x;
12 } // end function square with int argument
13
14 // function square for double values
15 double square( double y )
16 {
17    cout << "square of double " << y << " is ";
18    return y * y;
19 } // end function square with double argument
20
21 int main()
22 {
23    cout << square( 7 ); // calls int version
24    cout << endl;
25    cout << square( 7.5 ); // calls double version
26    cout << endl;
27    system("PAUSE");
28    return 0; // indicates successful termination
29 } // end main
30
```



48

Function Overloading

- Farklı nümerik veri türleri için (float toplama, integer toplama, double toplama) aynı isimle kullanılan matematiksel işlem yapan fonksiyonlar kullanılabilir.
- Overload yapılan fonksiyonları aynı parametre listesi ve farklı dönen değerle tanımlamak compile hatası oluşturur.

```
error C2556: 'double square(int)' : overloaded function differs only by return type from 'int square(int)'
```

49

Konular

- Giriş
- math Kütüphane Fonksiyonları
- Çok Parametreyle Fonksiyon Tanımı
- Fonksiyon Prototipleri
- C++ Standart Kütüphaneleri
- Örnek: Rastgele Sayı Üreteci
- Scope Kuralları
- Unary Scope Resolution Operator
- Call-by-Value ve Call-by-Reference
- Default Parametreler
- Function Overloading
- Özyineleme (Recursion)

Özyineleme (Recursion)

- Bazı problemlerin çözümünde fonksiyonların kendi kendisini çağırması daha uygundur.
- Bir **recursive function** doğrudan yada dolaylı olarak kendisini çağıran fonksiyondur.
- Bir problem birbirine benzeyen basit parçalar halinde ifade edilirse her parça ayrı ayrı çözülebilir.
- Her parçanın çözümünde aynı işlemlerin tekrar edilmesi için aynı fonksiyon tekrar tekrar çağırılabilir.
- Recursive çağırmada tüm paçalar için fonksiyon kendi kendisini çağırır.

51

Özyineleme (Recursion)

- Faktöriyel işlemi recursive fonksiyon çağırma kullanılarak çözülebilir.

$$n! = n(n-1)(n-2). \ldots .1$$

- Iterative bir şekilde çözüm aşağıdaki gibi yapılabilir:

```
factorial = 1;

for ( int counter = number; counter >= 1; counter-- )
   factorial *= counter;
```

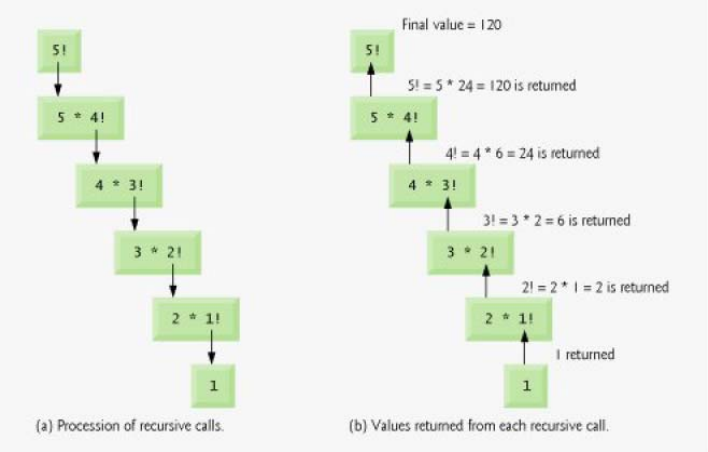
- Recursive şekilde problem aşağıdaki gibi ifade edilir:

$$n! = n(n-1)!$$

52

Özyineleme (Recursion)

- Recursive çalışmaya için örnek aşağıda görülmektedir.



53

Özyineleme (Recursion)

```
// Fig. 6.29: fig06_29.cpp
// Testing the recursive factorial function.
#include <iostream>
using std::cout;
using std::endl;

#include <iomanip>
using std::setw;

unsigned long factorial( unsigned long ); // function prototype

int main()
{
   // calculate the factorials of 0 through 10
   for ( int counter = 0; counter <= 10; counter++ )
      cout << setw( 2 ) << counter << "! = " << factorial( counter )
         << endl;

   system("PAUSE");
   return 0; // indicates successful termination
} // end main

// recursive definition of function factorial
unsigned long factorial( unsigned long number )
{
   if ( number <= 1 ) // test for base case
      return 1; // base cases: 0! = 1 and 1! = 1
   else // recursion step
      return number * factorial( number - 1 );
} // end function factorial
```

```
c:\Documents and Settings\M..ALI\My Documents\Visual St
 0! = 1
 1! = 1
 2! = 2
 3! = 6
 4! = 24
 5! = 120
 6! = 720
 7! = 5040
 8! = 40320
 9! = 362880
10! = 3628800
Press any key to continue . . .
```

54

Özyineleme (Recursion)

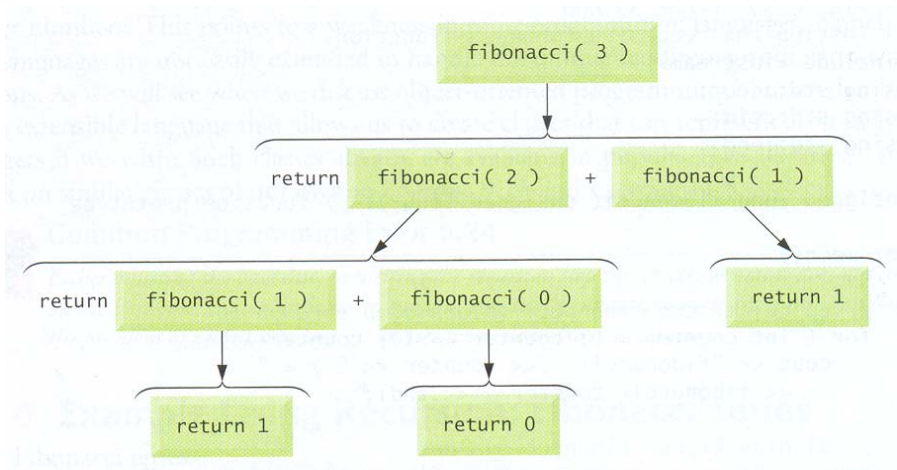
```
1  // Fig. 6.30: fig06_30.cpp
2  // Testing the recursive fibonacci function.
3  #include <iostream>
4  using std::cout;
5  using std::cin;
6  using std::endl;
7
8  unsigned long fibonacci( unsigned long ); // function prototype
9
10 int main()
11 {
12    // calculate the fibonacci values of 0 through 10
13    for ( int counter = 0; counter <= 10; counter++ )
14       cout << "fibonacci( " << counter << " ) = "
15          << fibonacci( counter ) << endl;
16
17    // display higher fibonacci values
18    cout << "fibonacci( 20 ) = " << fibonacci( 20 ) << endl;
19    cout << "fibonacci( 30 ) = " << fibonacci( 30 ) << endl;
20    cout << "fibonacci( 35 ) = " << fibonacci( 35 ) << endl;
21    system("PAUSE");
22    return 0; // indicates successful termination
23 } // end main
24
25 // recursive method fibonacci
26 unsigned long fibonacci( unsigned long number )
27 {
28    if ( ( number == 0 ) || ( number == 1 ) ) // base cases
29       return number;
30    else // recursion step
31       return fibonacci( number - 1 ) + fibonacci( number - 2 );
32 } // end function fibonacci
33
```

fibonacci (0) = 0
fibonacci (1) = 1
fibonacci (n) = fibonacci (n-1) + fibonacci (n-2)

```
c:\Documents and Settings\M...ALI\My Documen
fibonacci( 0 ) = 0
fibonacci( 1 ) = 1
fibonacci( 2 ) = 1
fibonacci( 3 ) = 2
fibonacci( 4 ) = 3
fibonacci( 5 ) = 5
fibonacci( 6 ) = 8
fibonacci( 7 ) = 13
fibonacci( 8 ) = 21
fibonacci( 9 ) = 34
fibonacci( 10 ) = 55
fibonacci( 20 ) = 6765
fibonacci( 30 ) = 832040
fibonacci( 35 ) = 9227465
Press any key to continue . . .
```

55

Özyineleme (Recursion)



56

Özyineleme (Recursion)

```
1  // Fig. 6.32: fig06_32.cpp
2  // Testing the iterative factorial method.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include <iomanip>
8  using std::setw;
9
10 unsigned long factorial( unsigned long ); // function prototype
11
12 int main()
13 {
14    // calculate the factorials of 0 through 10
15    for ( int counter = 0; counter <= 10; counter++ )
16       cout << setw( 2 ) << counter << "! = " << factorial( counter )
17          << endl;
18
19    system("PAUSE");
20    return 0;
21 } // end main
22
23 // iterative method factorial
24 unsigned long factorial( unsigned long number )
25 {
26    unsigned long result = 1;
27
28    // iterative declaration of method factorial
29    for ( unsigned long i = number; i >= 1; i-- )
30       result *= i;
31
32    return result;
33 } // end function factorial
34
```

Iterative faktöriyel hesabı

```
c:\Documents and Settings\M...ALI\My Docume
 0! = 1
 1! = 1
 2! = 2
 3! = 6
 4! = 24
 5! = 120
 6! = 720
 7! = 5040
 8! = 40320
 9! = 362880
10! = 3628800
Press any key to continue . . .
```

57

Ödev

- $f(x) = ax^3 + bx^2 + cx + d$ şeklinde bir fonksiyonun a, b, c ve d katsayıları dışarıdan giriliyor.
- Bu eşitliğin f(x) = 0 olduğu bir x değerini **bisection**, **secant** ve **newton** metodlarıyla **f(x) = ±0,001** aralığında bulan bir program yazınız.
- Program ilk çalıştığında aşağıdaki ekran gelecektir.

```
a katsayısını giriniz = 1
b katsayısını giriniz = -3
c katsayısını giriniz = 0
d katsayısını giriniz = 1

1- Bisection
2- Secant
3- Newton
4- Yeni katsayı girişi

Metodu seçiniz (1, 2, 3, 4, Çıkış için 0 giriniz) : 1
```

- Hesaplama sırasında her adımda, adım sırası, x ve y değerleri ekrana yazılacaktır.
- Menüden 4 seçilirse yeni bir fonksiyon için katsayı girişi yapılacaktır.
- 1-3 arası seçimlerde girilen fonksiyon için seçilen metod uygulanacaktır.
- Her metod seçiminden sonra ekran temizlenerek menüye dönülecektir.

58

Ödev

- Aşağıdaki örnek çıktı y = x^3 - $3x^2$ + 1 fonksiyonu ve Bisection metodu için verilmiştir.

```
Bisection
*************
i              xi               yi
******         ******           ******
1              0,5000            0,3750
2              0,7500           -0,2656
3              0,6250            0,0722
4              0,6875           -0,0930
5              0,6562           -0,0093
6              0,6406            0,0317
7              0,6484            0,0112
8              0,6523            0,0009

Press any key to continue...
```

59