جامعة
القاهرة الجديدة
التكنولوجية

NEW CAIRO
TECHNOLOGICAL
UNIVERSITY

TUE – The Technological Universities in Egypt

NCTU – New Cairo Technological University

Faculty of Industry and Energy Technology

Information Technology Department

Second-Year

Course: Programming Essentials in C++

Lecture 3

Presented by

Dr. Ghada Maher

Contents:

Example 14: Integer Overflow

This program repeatedly multiplies n by 1000 until it overflows.

```
int main()
{ // prints n until it overflows:
  int n=1000;
  cout << "n = " << n << endl;
  n *= 1000;  // multiplies n by 1000
  cout << "n = " << n << endl;
  n *= 1000;  // multiplies n by 1000
  cout << "n = " << n << endl;
  n *= 1000;  // multiplies n by 1000
  cout << "n = " << n << endl;
}
```

```
n = 1000
n = 1000000
n = 1000000000
n = -727379968
```

This shows that the computer that ran this program cannot multiply 1,000,000,000 by 1000 correctly.

Example 15: Floating-point Overflow

This program is similar to the one in Example 2.12. It repeatedly squares x until it overflows.

```
int main()
{ // prints x until it overflows:
  float x=1000.0;
  cout << "x = " << x << endl;
  x *= x;  // multiplies n by itself; i.e., it squares x
  cout << "x = " << x << endl;
  x *= x;  // multiplies n by itself; i.e., it squares x
  cout << "x = " << x << endl;
  x *= x;  // multiplies n by itself; i.e., it squares x
  cout << "x = " << x << endl;
  x *= x;  // multiplies n by itself; i.e., it squares x
  cout << "x = " << x << endl;
}
```

```
x = 1000
x = 1e+06
x = 1e+12
x = 1e+24
x = inf
```

This shows that, starting with x = 1000, this computer cannot square x correctly more than three times. The last output is the special symbol `inf` which stands for "infinity."

the infinity symbol `inf` instead of the correct value of 1048. Integer overflow "wraps around" to negative integers. Floating-point overflow "sinks" into the abstract notion of infinity.

How to convert 1e+6 to decimal number

Before you continue, note that the number 1e+6 is in scientific notation, also known as standard form. Used to write large or small numbers in another way. In the number 1e+6, the numbers are defined as follows:

1 = coefficient

e = 10 to the power of

6 = exponent

The scientific notation 1e+6 is same as 1 x 10^6 or 1 x 10^6. Thus, to get the answer to 1e+6 as a decimal, we multiply 1 by 10 to the power of 6.

= 1e+6

= 1 × 10^6

= 1000000

Therefore, 1e+6 number on calculator means or 1e+6 in decimal form is:

1000000

Example 16: Round-off Error
This program does some simple arithmetic to illustrate roundoff


```
#include<iostream>
using namespace std;
int main()
{ // tests operators +,-,*,/,and %:
double x = 1000/3.0;cout << "x = " << x << endl; // x = 1000/3
double y = x - 333.0;cout << "y = " << y << endl; // y = 1/3
double z = 3*y - 1.0;cout << "z = " << z << endl; // z = 3(1/3) - 1
if (z == 0) cout << "z == 0.\n";
else
cout << "z does not equal 0.\n";
return 0;
}
```

```
x = 333.333
y = 0.333333
z = -5.68434e-14
z does not equal 0.
```

In exact arithmetic, the variables would have the values $x = 333\ 1/3$, $y = 1/3$, and $z = 0$. But 1/3 cannot be represented exactly as a floating-point value. The inaccuracy is reflected in the residue value for z.

Scope of Variables

- The scope of an identifier is that part of the program where it can be used. For example, variables cannot be used before they are declared, so their scopes begin where they are declared. This is illustrated by the next example

Example 17:

```
int main()
{ // illustrates the scope of variables:
  x = 11;     // ERROR: this is not in the scope of x
  int x;
  { x = 22;   // OK: this is in the scope of x
    y = 33;   // ERROR: this is not in the scope of y
    int y;
    x = 44;   // OK: this is in the scope of x
    y = 55;   // OK: this is in the scope of y
  }
  x = 66;     // OK: this is in the scope of x
  y = 77;     // ERROR: this is not in the scope of y
}
```

Nested and Parallel Scopes

A program may have several objects with the same name as long as their scopes are nested or disjoint. This is illustrated by the next example.

Example 18:

```
int x = 11;                                          // this x is global

int main()
{ // illustrates the nested and parallel scopes:
  int x = 22;
  { // begin scope of internal block
    int x = 33;
    cout << "In block inside main(): x = " << x << endl;
  }                                                  // end scope of internal block
  cout << "In main(): x = " << x << endl;
  cout << "In main(): ::x = " << ::x << endl;
}                                                    // end scope of main()
In block inside main(): x = 33
In main(): x = 22
In main(): ::x = 11
```

References:

John R. Hubbard, "*Schaum's outline of* theory and problems of programming with C++", Second Edition, Schaum's Outline Series, McGRAW-HILL, *New York San Francisco Washington.*

جامعة
القاهرة الجديدة
التكنولوجية

NEW CAIRO
TECHNOLOGICAL
UNIVERSITY

TUE – The Technological Universities in Egypt

NCTU – New Cairo Technological University

Faculty of Industry and Energy Technology

Information Technology Department

Second-Year

Course: Programming Essentials in C++

Lecture 3

Presented by

Dr. Ghada Maher

Contents:

- C++ Decision Making Statements (Selection control statements)
  - The if Statement
  - The if..else Statement
  - Keywords
  - Comparison Operators
  - Statement Blocks
  - Compound Conditions
  - Short-circuiting
  - Boolean Expressions
  - Nested Selection Statements
  - The else if Construct
  - The switch Statement
  - The Conditional Expression Operator

The if Statement

The if statement allows conditional execution. Its syntax is

if (condition) statement;

where condition is an integral expression and statement is any executable statement. The statement will be executed only if the value of the integral expression is nonzero. Notice the required parentheses around the condition.

Example 1: Testing for Divisibility


```
int main()   //This program tests if one positive integer is not divisible by another
{ int n,d;
cout << "Enter two positive integers: ";
cin >> n >> d;
if (n%d)
cout << n << " is not divisible by " << d << endl; }
```

Run:

On the first run, we enter 66 and 7:

```
Enter two positive integers: 66 7
66 is not divisible by 7
```

On the first run, we enter 56 and 7:

```
Enter two positive integers: 56 7
```

The value 56%7 is computed to be 0, which is interpreted to mean “false,” so the divisibility message isnot printed.

The if…..else Statement

- The if……..else statement causes one of two alternative statements to execute depending upon whether the condition is true. Its syntax is

if (*condition***)** *statement1***;**

else *statement2***;**

where *condition* is an integral expression and *statement1* and *statement2* are executable statements. If the value of the condition is nonzero then *statement1* will execute; otherwise *statement2* will execute.

Example 2: Testing for Divisibility Again


```
int main()
{ int n,d;
cout << "Enter two positive integers: ";
cin >> n >> d;
if (n%d) cout << n << " is not divisible by " << d << endl;
else cout << n << " is divisible by " << d << endl;
}
```

Run:

```
Enter two positive integers: 56 7
56 is divisible by 7
```

Keywords:

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

Keywords:

- Keywords like **if** and **else** are found in nearly every programming language.
- Other keywords such as **dynamic_cast** are unique to C++. The 74 keywords of C++ include all 32 of the keywords of the C language.
- There are two kinds of keywords: reserved words and standard identifiers.
- **A *reserved word*** is a keyword that serves as a structure marker, used to define the syntax of the language.
    - The key words **if** and **else** are reserved words.
- **A *standard identifier*** is a keyword that names a specific element of the language.
    - The keywords **bool** and **int** are standard identifiers because they are names of standard types in C++.
- See the following table for more information on the C++ keywords.

Keyword	Description	Example
`and`	A synonym for the AND operator `&&`	`(x>0 and x<8)`
`and_eq`	A synonym for the bitwise AND assignment operator `&=`	`b1 and_eq b2;`
`asm`	Allows information to be passed to the assembler directly	`asm ("check");`
`auto`	Storage class for objects that exist only within their own block	`auto int n;`
`bitand`	A synonym for the bitwise AND operator `&`	`b0 = b1 bitand b2;`
`bitor`	A synonym for the bitwise OR operator |	`b0 = b1 bitor b2;`
`bool`	A boolean type	`bool flag;`
`break`	Terminates a loop or a `switch` statement	`break;`
`case`	Used in a `switch` statement to specify control expression	`switch (n/10)`
`catch`	Specifies actions to take when an exception occurs	`catch(error)`
`char`	An integer type	`char c;`
`class`	Specifies a class declaration	`class X { ... };`
`compl`	A synonym for the bitwise NOT operator ~	`b0 = compl b1;`
`const`	Specifies a constant definition	`const int s = 32;`
`const_cast`	Used to change objects from within immutable member functions	`pp = const_cast<T*>(p)`
`continue`	Jumps to beginning of next iteration in a loop	`continue;`
`default`	The "otherwise" case in a `switch` statement	`default: sum = 0;`
`delete`	Deallocates memory allocated by a `new` statement	`delete a;`
`do`	Specifies a `do..while` loop	`do {...} while ...`
`double`	A real number type	`double x;`
`dynamic_cast`	Returns a `T*` pointer for a given pointer	`pp = dynamic_cast<T*>p`
`else`	Specifies alternative in an `if` statement	`else n = 0;`
`enum`	Used to declare an enumeration type	`enum bool { ... };`
`explicit`	Used to prevent a constructor from being invoked implicitly	`explicit X(int n);`
`export`	Allows access from another compilation unit	`export template<class T>`
`extern`	Storage class for objects declared outside the local block	`extern int max;`

Keyword	Description	Example
`false`	One of the two literals for the `bool` type	`bool flag=false;`
`float`	A real number type	`float x;`
`for`	Specifies a `for` loop	`for ( ; ; ) ...`
`friend`	Specifies a `friend` function in a class	`friend int f();`
`goto`	Causes execution to jump to a labeled statement	`goto error;`
`if`	Specifies an `if` statement	`if (n > 0) ...`
`inline`	Declares a function whose text is to be substituted for its call	`inline int f();`
`int`	An integer type	`int n;`
`long`	Used to define integer and real types	`long double x;`
`mutable`	Allows immutable functions to change the field	`mutable string ssn;`
`namespace`	Allows the identification of scope blocks	`namespace Best { int num; }`
`new`	Allocates memory	`int* p = new int;`
`not`	A synonym for the NOT operator !	`(not(x==0))`
`not_eq`	A synonym for the inequality operator !=	`(x not_eq 0)`
`operator`	Used to declare an operator overload	`X operator++();`
`or`	A synonym for the OR operator ||	`(x>0 or x<8)`
`or_eq`	A synonym for the bitwise OR assignment operator |=	`b1 or_eq b2;`
`private`	Specifies `private` declarations in a class	`private: int n;`
`protected`	Specifies `protected` declarations in a class	`protected: int n;`
`public`	Specifies `public` declarations in a class	`public: int n;`
`register`	Storage class specifier for objects stored in registers	`register int i;`
`reinterpret_cast`	Returns an object with given value and type	`pp = reinterpret_cast<T*>(p)`
`return`	Statement that terminates a function and returns a value	`return 0;`
`short`	An integer type	`short n;`
`signed`	Used to define integer types	`signed char c;`
`sizeof`	Operator that returns the number of bytes used to store an object	`n = sizeof(float);`

Keyword	Description	Example
`static`	Storage class of objects that exist for the duration of the program	`static int n;`
`static_cast`	Returns a `T*` pointer for a given pointer	`pp = static_cast<T*>p`
`struct`	Specifies a structure definition	`struct X { ... };`
`switch`	Specifies a `switch` statement	`switch (n) { ... }`
`template`	Specifies a `template` class	`template <class T>`
`this`	Pointer that points to the current object	`return *this;`
`throw`	Used to generate an exception	`throw X();`
`true`	One of the two literals for the `bool` type	`bool flag=true;`
`try`	Specifies a block that contains exception handlers	`try { ... }`
`typedef`	Declares a synonym for an existing type	`typedef int Num;`
`typeid`	Returns an object that represents an expression's type	`cout << typeid(x).name();`
`typename`	A synonym for the keyword `class`	`typename X { ... };`
`using`	Directive that allows omission of namespace prefix	`using namespace std;`
`union`	Specifies a structure whose elements occupy the same storage	`union z { ... };`
`unsigned`	Used to define integer types	`unsigned int b;`
`virtual`	Declares a member function that is defined in a subclass	`virtual int f();`
`void`	Designates the absence of a type	`void f();`
`volatile`	Declares objects that can be modified outside of program control	`int volatile n;`
`wchar_t`	Wide (16-bit) character type	`wchar_t province;`
`while`	Specifies a `while` loop	`while (n > 0) ...`
`xor`	A synonym for the bitwise exclusive OR operator ^	`b0 = b1 xor b2;`
`xor_eq`	A synonym for the bitwise exclusive OR assignment operator ^=	`b1 xor_eq b2;`

Comparison Operators

The six *comparison operators* are:

- **x < y** // x is less than y
- **x > y** // x is greater than y
- **x <= y** // x is less than or equal to y
- **x >= y** // x is greater than or equal to y
- **x == y** // x is equal to y
- **x != y** // x is not equal to y

- Note that in C++ the single equal sign "=" is the *assignment operator*, and the double equal sign "==" is the *equality operator*:

```
x = 33; // assigns the value 33 to x
x == 33; // evaluates to 0 (for false) unless 33 is the value of x
```

(This distinction is critically important)

Example 3: The Minimum of Two Integers


```
int main()      // This program prints the minimum of the two integers entered
{ int m,n;
cout << "Enter two integers: ";
cin >> m >> n;
if (m < n) cout << m << " is the minimum." << endl;
else cout << n << " is the minimum." << endl; }
```

Run:

```
Enter two integers: 77 55
55 is the minimum.
```

Example 4: Logical Error


```
int main()
{ int n;
cout << "Enter an integer: ";
cin >> n;
if (n = 22) cout << n << " = 22" << endl; // LOGICAL ERROR!
else cout << n << " != 22" << endl;
}
```

Run:

```
Enter an integer: 77
22 = 22
```

Example 5: The Minimum of Three Integers


```
int main()
{ int n1,n2,n3;
cout << "Enter three integers: ";
cin >> n1 >> n2 >> n3;
int min=n1; // now min <= n1
if (n2 < min) min = n2; // now min <= n1 and min <= n2
if (n3 < min) min = n3; // now min <= n1,min <= n2,and min <= n3
cout << "Their minimum is " << min << endl;}
```

Run:

```
Enter three integers: 77 33 55
Their minimum is 33
```

Statement Blocks

A *statement block* is a sequence of statements enclosed by braces**{ }**, like this:

{ int temp=x; x = y; y = temp; **}**

In C++ programs, a statement block can be used anywhere that a single statement can be used.

Example 6: A Statement Block within an if Statement


```
int main()
{ int x,y;
cout << "Enter two integers: ";
cin >> x >> y;
if (x > y) { int temp=x; x = y; y = temp; } // swap x and y
cout << x << " <= " << y << endl;}
```

Run:

```
Enter two integers: 66 44
44 <= 66
```

Example 7: Using Blocks to Limit Scope


```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int n=44;
6      cout << "n = " << n << endl;
7      {
8          int n; // scope extends over 4 lines
9          cout << "Enter an integer: ";
10         cin >> n;
11         cout << "n = " << n << endl;
12
13     }
14         {
15         cout << "n = " << n << endl; // the n that was declared first
16         }
17         {
18             int n; // scope extends over 2 lines
19             cout << "n = " << n << endl;
20         }
21         cout << "n = " << n << endl; // the n that was declared first
22         return 0;
23
24 }
```

Run:

```
n = 44
Enter an integer: 77
n = 77
n = 44
n = 4251897
n = 44
```