AAB University

Faculty of Computer Sciences

Object Oriented Programming

Week 3:

C++ Programming, Input/Output and Operators

Asst. Prof. Dr. **Mentor Hamiti**

mentor.hamiti@universitetiaab.com

Last Time?!

- An Introduction To Computer Science
- Algorithms
- Programming Languages
- Programming Paradigms
  - Structural (Procedural) Programming
  - Object-Oriented Programming

Today

- Structure of a program
- Variables
- Memory Concepts
- Arithmetic
- Decision Making
- ……

Object Oriented Programming

- Visual Studio
- DEV C++
- . . .

First Program in C++

- The best way to learn a programming language is by writing programs
  - Although the first program is very simple, it contains all the fundamental components C++ programs have:

```
1 // my first program in C++
2 #include <iostream>
3
4 int main()
5 {
6   std::cout << "Hello World!";
7 }
```

```
Hello World!
```

First Program in C++

```
1 // my first program in C++
2 #include <iostream>
3
4 int main()
5 {
6   std::cout << "Hello World!";
7 }
```

```
Hello World!
```

- Line 1: // my first program in C++
  - *Two slash signs indicate that the rest of the line is a comment inserted by the programmer but which has no effect on the behavior of the program*

- Line 2: #include <iostream>
  - *The directive #include <iostream>, instructs the preprocessor to include a section of standard C++ code, known as header iostream, that allows to perform standard input and output operations, such as writing the output of this program to the screen*

First Program in C++

```
1 // my first program in C++
2 #include <iostream>
3
4 int main()
5 {
6   std::cout << "Hello World!";
7 }
```

```
Hello World!
```

- Line 3: A blank line
  - *Blank lines have no effect on a program. They simply improve readability of the code*

- Line 4: int main ()
  - *The function named main is a special function in all C++ programs; it is the function called when the program is run. The execution of all C++ programs begins with the main function, regardless of where the function is actually located within the code.*

First Program in C++

```
1 // my first program in C++
2 #include <iostream>
3
4 int main()
5 {
6   std::cout << "Hello World!";
7 }
```

```
Hello World!
```

- Lines 5 and 7: { and }
  - *Everything between these braces is the function's body that defines what happens when main is called. All functions use braces to indicate the beginning and end of their definitions*

- Line 6: std::cout << "Hello World!";
  - *This line is a C++ statement. A statement is an expression that can actually produce some effect. In this case, a sentence within quotes ("Hello world!"), is the content inserted into the standard output.*

First Program in C++

- The C++ does not have strict rules on indentation or on how to split instructions in different lines!
  - The following codes would have had exactly the same meaning:

```
1 int main ()
2 {
3   std::cout << " Hello World!";
4 }
```

```
int main () { std::cout << "Hello World!"; }
```

Using namespace std

- The declaration:

```
using namespace std;
```

  - allows all elements in the **std namespace** to be accessed without the std:: prefix

```
1 // my second program in C++
2 #include <iostream>
3 using namespace std;
4
5 int main ()
6 {
7   cout << "Hello World! ";
8 }
```

The return Statement

- According to the C++ standard,
  - *When the return statement is used at the end of main function, the value 0 indicates that the program has terminated successfully*

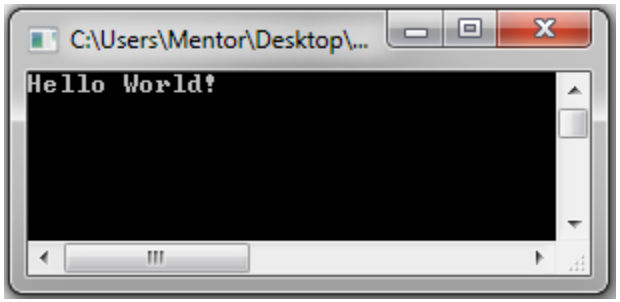
```
return 0; // indicate that program ended successfully
```

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

int main()
{

    cout << "Hello World! ";

    cin.get();
    return 0;
}
```



©1992-2014 by Pearson Education, Inc. All Rights Reserved.

Comments

- C++ supports two ways of commenting code:

```
1 // line comment
2 /* block comment */
```

  - *Comments do not affect the operation of the program; however, they provide an important tool to document directly within the source code what the program does and how it operates*

Escape sequences

- Some common escape sequences

Escape sequence	Description
\n	Newline. Position the screen cursor to the beginning of the next line.
\t	Horizontal tab. Move the screen cursor to the next tab stop.
\r	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
\a	Alert. Sound the system bell.
\\	Backslash. Used to print a backslash character.
\'	Single quote. Use to print a single quote character.
\"	Double quote. Used to print a double quote character.

©1992-2014 by Pearson Education, Inc. All Rights Reserved.

Program 2

- Program that displays the sum of two integers

```
// Program that displays the sum of two integers
#include <iostream>
using namespace std;

int main()
{
    int number1;        // first integer to add
    int number2;        // second integer to add
    int sum;            // sum of number1 and number2

    cout << "Enter first integer: ";    // prompt user for data
    cin >> number1;                     // read first integer from user into number1

    cout << "Enter second integer: ";   // prompt user for data
    cin >> number2;                     // read second integer from user into number2

    sum = number1 + number2;            // add the numbers; store result in sum
    cout << "Sum is " << sum <<endl;    // display sum; end line

    cin.get();
    return 0;
}
```

```
Enter first integer: 45
Enter second integer: 72
Sum is 117
```


©1992-2014 by Pearson Education, Inc. All Rights Reserved.

Variables

- Variable Declarations

```
int number1; // first integer to add
int number2; // second integer to add
int sum; // sum of number1 and number2
```

- A variable name is any valid identifier that is not a keyword
- An identifier is a series of characters consisting of letters, digits and underscores (_) that does not begin with a digit
- C++ is case sensitive—uppercase and lowercase letters are different, so a1 and A1 are different identifiers.

©1992-2014 by Pearson Education, Inc. All Rights Reserved.

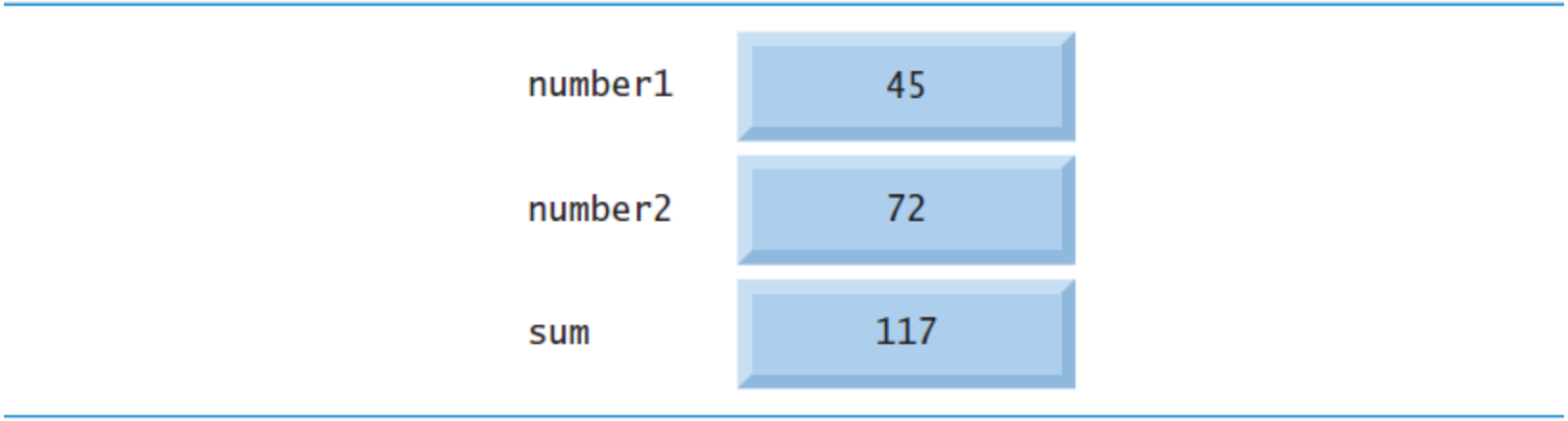
Memory Concepts

- A variable has a names, a type, a size and a value
  - Variable names such as number1, number2 and sum actually correspond to locations in the computer's memory
  - When a value is placed in a memory location, the value overwrites the previous value in that location; thus, placing a new value into a memory location is said to be destructive
  - When a value is read out of a memory loca-tion, the process is nondestructive

©1992-2014 by Pearson Education, Inc. All Rights Reserved.

Memory Concepts

- Memory locations after calculating and storing the sum of number1 and number2



©1992-2014 by Pearson Education, Inc. All Rights Reserved.

Arithmetic

- Most programs perform arithmetic calculations
  - Arithmetic operators

C++ operation	C++ arithmetic operator	Algebraic expression	C++ expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	bm or $b \cdot m$	`b * m`
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	`x / y`
Modulus	%	$r \bmod s$	`r % s`

©1992-2014 by Pearson Education, Inc. All Rights Reserved.

Arithmetic

- Precedence of arithmetic operators

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. [*Caution:* If you have an expression such as `(a + b) * (c - d)` in which two sets of parentheses are not nested, but appear "on the same level," the C++ Standard does *not* specify the order in which these parenthesized subexpressions will be evaluated.]
*, /, %	Multiplication, Division, Modulus	Evaluated second. If there are several, they're evaluated left to right.
+ -	Addition Subtraction	Evaluated last. If there are several, they're evaluated left to right.

©1992-2014 by Pearson Education, Inc. All Rights Reserved.

Arithmetic

- Example:

```
Step 1.   y = 2 * 5 * 5 + 3 * 5 + 7;      (Leftmost multiplication)
          2 * 5 is 10
          y = 10 * 5 + 3 * 5 + 7;
Step 2.                                   (Leftmost multiplication)
          10 * 5 is 50
          y = 50 + 3 * 5 + 7;
Step 3.                                   (Multiplication before addition)
          3 * 5 is 15
          y = 50 + 15 + 7;
Step 4.                                   (Leftmost addition)
          50 + 15 is 65
          y = 65 + 7;
Step 5.                                   (Last addition)
          65 + 7 is 72
          y = 72
Step 6.                                   (Last operation—place 72 in y)
```


©1992-2014 by Pearson Education, Inc. All Rights Reserved.

- Equality and Relational Operators
  - The **if** statement allows a program to take alternative action based on whether a condition is true or false
    - *If the condition is **true**, the statement in the body of the if statement is executed*
    - *If the condition is **false**, the body statement is not executed*
  - Conditions in if statements can be formed by using:
    - *Equality operators and*
    - *Relational operators*

©1992-2014 by Pearson Education, Inc. All Rights Reserved.

- Equality and Relational Operators

Standard algebraic equality or relational operator	C++ equality or relational operator	Sample C++ condition	Meaning of C++ condition
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y

©1992-2014 by Pearson Education, Inc. All Rights Reserved.

Program 3

```
// Comparing integers using if statements, relational operators and equality operators
# include<iostream>
using namespace std;

int main()
{
    int number1; // first integer to compare
    int number2; // second integer to compare

    cout << "Enter two integers to compare: "; // prompt user for data
    cin >> number1 >> number2; // read two integers from user

    if ( number1 == number2 )
        cout << number1 << " == " << number2 << endl;

    if( number1 != number2)
        cout << number1 << " != " << number2 << endl;

    if( number1 < number2)
        cout << number1 << " < " << number2 << endl;

    if ( number1 > number2 )
        cout << number1 << " > " << number2 << endl;

    if( number1 <= number2)
        cout << number1 << " <= " << number2 << endl;

    if( number1 >= number2)
        cout << number1 << " >= " << number2 << endl;

    cin.get();
    return 0;
}
```


©1992-2014 by Pearson Education, Inc. All Rights Reserved.

Program 3

- Comparing integers using if statements, relational operators and equality operators

```
Enter two integers to compare: 3 7
3 != 7
3 < 7
3 <= 7
```

```
Enter two integers to compare: 22 12
22 != 12
22 > 12
22 >= 12
```

```
Enter two integers to compare: 7 7
7 == 7
7 <= 7
7 >= 7
```


©1992-2014 by Pearson Education, Inc. All Rights Reserved.

Decision Making

- Precedence and associativity of the operators

Operators				Associativity	Type
()				*[See caution in Fig. 2.10]*	grouping parentheses
*	/	%		left to right	multiplicative
+	-			left to right	additive
<<	>>			left to right	stream insertion/extraction
<	<=	>	>=	left to right	relational
==	!=			left to right	equality
=				right to left	assignment

©1992-2014 by Pearson Education, Inc. All Rights Reserved.

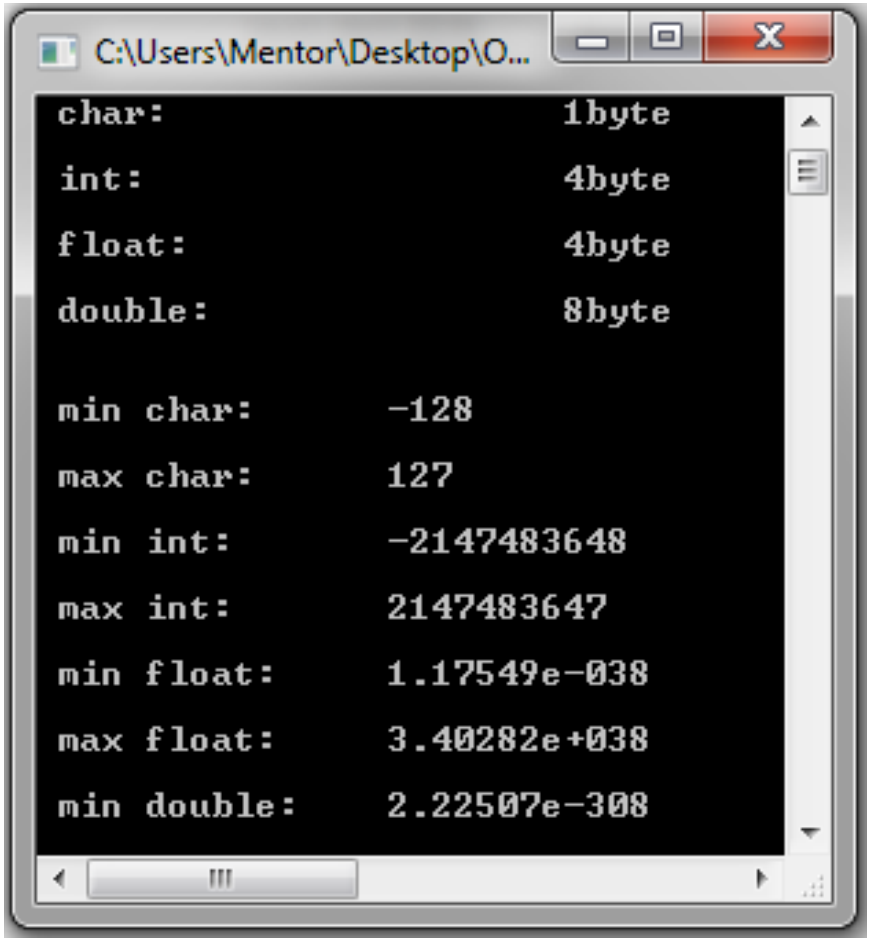
Exercise 1

- Write a program that prints a box, an oval, an arrow and a diamond as follows:

©1992-2014 by Pearson Education, Inc. All Rights Reserved.

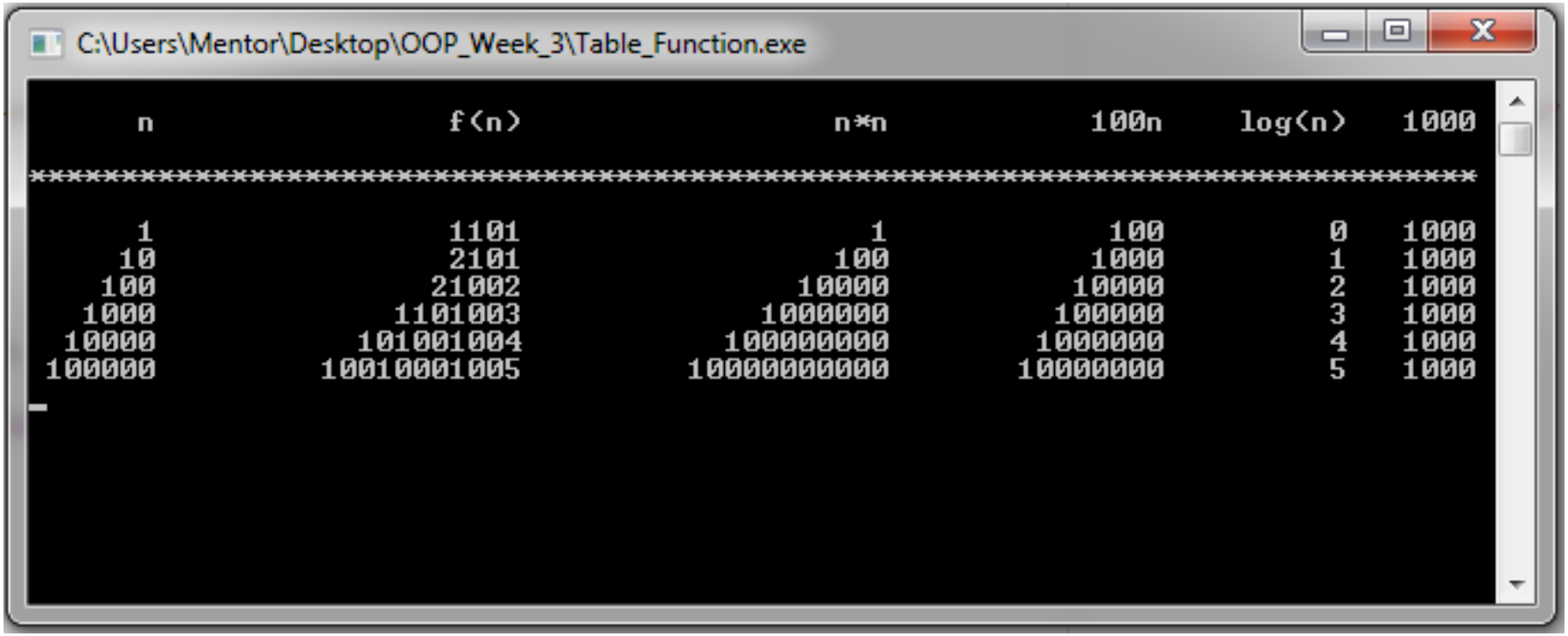
Exercise 2

- Write a program that calculate the number of bytes reserved in memory for the following type of variables:
  - char
  - int
  - float
  - double
- Find also the min and max values?!



Exercise 3

- Write a program that result with the following output:



- Use: **# include <iomanip>** and **# include <cmath>**

- Questions?!

mentor.hamiti@universitetiaab.com