Mittuniversitetet
MID SWEDEN UNIVERSITY

Lecture #2
Elementary Programming

Chapter: 2, 19.6

Objectives

Mittuniversitetet
MID SWEDEN UNIVERSITY

- To use identifiers to name elements such as variables.
- To write C++ programs that perform simple computations.
- To read input from the keyboard.
- To name constants using the **const** keyword.
- To declare variables using numeric data types.
- To write integer literals, floating-point literals.
- To use augmented assignment operators (**+=**, **-=**, ***=**, **/=**, **%=**).
- To distinguish between post-increment and pre-increment and between post-decrement and pre-decrement.
- To convert numbers to a different type using casting.
- To obtain the current system time using **time(0)**.

Identifiers

Mittuniversitetet
MID SWEDEN UNIVERSITY

- An identifier is a sequence of characters that consists of letters, digits, and underscores (_).
- An identifier must start with a letter or an underscore. It cannot start with a digit.
- An identifier cannot be a reserved word
- An identifier can be of any length, but your C++ compiler may impose some restriction. Use identifiers of 31 characters or fewer to ensure portability.

Declaring Variables

Mittuniversitetet
MID SWEDEN UNIVERSITY

```
int number;         // Declare number to be an integer variable;

double radius; // Declare radius to be a double variable;

char aChar;         // Declare aChar to be a character variable;
```

Variables

Mittuniversitetet
MID SWEDEN UNIVERSITY

```
// Compute the first area
radius = 1.0;
area = radius * radius * 3.14159;

// Compute the second area
radius = 2.0;
area = radius * radius * 3.14159;
```

Assignment Statements

Mittuniversitetet
MID SWEDEN UNIVERSITY

```
number = 1;          // Assign 1 to number;
radius = 1.0;    // Assign 1.0 to radius;
aChar = 'A';         // Assign 'A' to aChar;
```

Declaring and Initializing

```
int number1 = 1;
double price = 1.4;
```

Named Constants

Mittuniversitetet
MID SWEDEN UNIVERSITY

Syntax:

```
const datatype CONSTANTNAME = VALUE;
```

Example:

```
const double PI = 3.14159;
const int SIZE = 3;
```

A complete example

Mittuniversitetet
MID SWEDEN UNIVERSITY

Computing the Area of a Circle

Lets put it together!

ComputeArea

Reading Input from the Keyboard

Mittuniversitetet
MID SWEDEN UNIVERSITY

You can use the **cin object** to read input from the keyboard.

```
std::cout << "Enter a radius: ";
std::cin >> radius
```

ComputeAreaWithConsoleInput

Reading Multiple Input in One Statement

Mittuniversitetet
MID SWEDEN UNIVERSITY

```
std::cin >> number1 >> number2 >> number3;

  // Compute average
  double average = (number1 + number2 + number3) / 3;
```

ComputeAverage

sizeof Operator

Mittuniversitetet
MID SWEDEN UNIVERSITY

You can use the **sizeof operator** to find the size of a type. For example, the following statement displays the size of int, long, and double on your machine.

```
std::cout << sizeof(int) << " " << sizeof(long) << " " << sizeof(double);
```

Example: Displaying Time

Mittuniversitetet
MID SWEDEN UNIVERSITY

Write a program that obtains hours and minutes from seconds.

```
int seconds = 500;
int minutes = seconds / 60;
int remainingSeconds = seconds % 60;
```

DisplayTime

Augmented Assignment Operators

Mittuniversitetet
MID SWEDEN UNIVERSITY

Operator	*Example*	*Equivalent*
+=	count += 8	count = count + 8
-=	count -= 8.0	count = count - 8.0
*=	count *= 8	count = count * 8
/=	count /= 8	count = count / 8
%=	count %= 8	count = count % 8

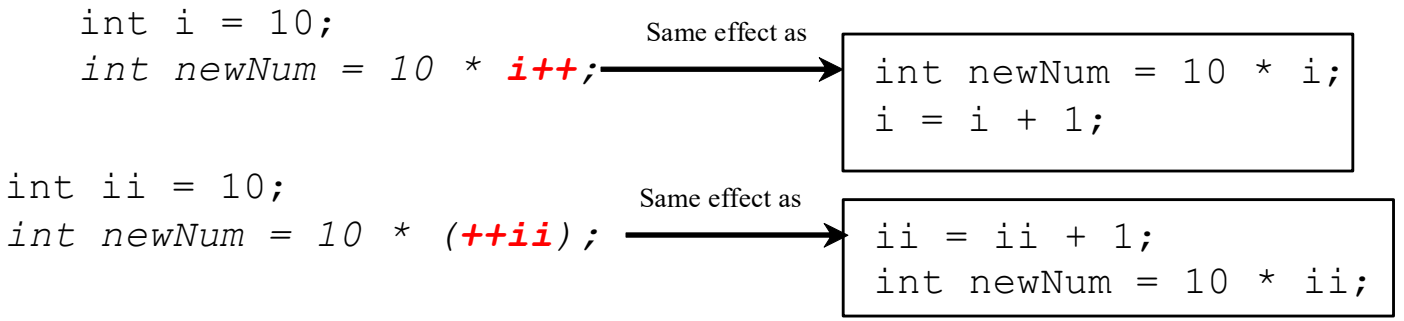
Increment and Decrement Operators

Mittuniversitetet
MID SWEDEN UNIVERSITY

Operator	Name	Description
++var	preincrement and evaluates	The expression (++var) increments var by 1 to the *new* value in var *after* the increment.
var++	postincrement *original* value	The expression (var++) evaluates to the in var and increments var by 1.
--var	predecrement and evaluates	The expression (--var) decrements var by 1 to the *new* value in var *after* the decrement.
var--	postdecrement *original* value	The expression (var--) evaluates to the in var and decrements var by 1.

Increment and Decrement Operators, cont.

Mittuniversitetet
MID SWEDEN UNIVERSITY



Using increment and decrement operators makes expressions short, but it also makes them complex and difficult to read. Avoid using these operators in expressions that modify multiple variables, or the same variable for multiple times such as this: int k = ++i + i.

Type Casting

Mittuniversitetet
MID SWEDEN UNIVERSITY

Implicit casting:

double price = 3; (type widening)

Explicit casting:

int aNumber = static_cast<int>(3.0); (type narrowing)

int anotherNumber= (int)3.9; (Fraction part is truncated) // Old way, not safe

Casting does not change the variable being cast. For example, dec is not changed after casting in the following code:

```
double dec = 4.5;
Int thirdNumber= static_cast<int>(dec);  // d is not changed
```

Controlling precision of floating-point values

Mittuniversitetet
MID SWEDEN UNIVERSITY

```
#include <iomanip>

...

const double PI = 3.14159;

std::cout<< std::fixed << std::setprecision(2)<< PI <<std::endl;
```

std::fixed means that the output will have a fixed number of digits

std::setprecision(n) is a predefined function, means that the output will have n digits after the decimal point.

More predefined functions

Mittuniversitetet
MID SWEDEN UNIVERSITY

Every library has its own predefined functions. We call them with appropriate parameters

```
#include <cmath>    //Matematical predefined functions

returned_value = sqrt(double number)

returned_value = pow(double base, double exponent)
```

Thank you