AIOU Stat Deptt

Computer Programming C/C++ (1564)

Zahoor Ahmad

Part : 2

Variables

- Location in memory where value can be stored
- Common data types
  - **`int`** - integer numbers
  - **`char`** - characters
  - **`double`** - floating point numbers
- Declare variables with name and data type before use

```
int integer1;
int integer2;
int sum;
```

- Can declare several variables of same type in one declaration
  - Comma-separated list

```
int integer1, integer2, sum;
```

Variables

- Variable names
  - Valid identifier
    - Series of characters (letters, digits, underscores)
    - Cannot begin with digit
    - Case sensitive
    - Should not be a keyword

Input streem & Assignment Operator

- Input stream object
  - **>>** (stream extraction operator)
    - Used with **std::cin**
    - Waits for user to input value, then press *Enter* (Return) key
    - Stores value in variable to right of operator
      - Converts value to variable data type
- **=** (Assignment Operator)
  - Assigns value to variable
  - Binary operator (two operands)
  - Example:

```
sum = variable1 + variable2;
```

A Simple Program

```
1  // Using cin, Assignment operator and addition
2  // Addition program.
3  #include <iostream>
4
5  // function main begins program execution
6  int main()
7  {
8     int integer1;  // first number to be input by user
9     int integer2;  // second number to be input by user
10    int sum;       // variable in which sum will be stored
11
12    std::cout << "Enter first integer\n";  // prompt
13    std::cin >> integer1;                  // read an integer
14
15    std::cout << "Enter second integer\n"; // prompt
16    std::cin >> integer2;                  // read an integer
17
18    sum = integer1 + integer2;  // assign result to sum
19
20    std::cout << "Sum is " << sum << std::endl; // print sum
21
22    return 0;   // indicate that program ended successfully
23
24 } // end function main
```

Declare integer variables.

Use stream extraction operator with standard input stream to obtain user input.

Calculations can be performed in output statements: alternative for lines 18 and 20:

```
std::cout << "Sum is " <<
integer1 + integer2 <<
std::endl;
```

Stream manipulator **`std::endl`** outputs a newline, then "flushes output buffer."

Concatenating, chaining or cascading stream insertion operations.

Compiling/Running a program

- Use Dev-C++ or any text editor to type in the program
- Save the program into a file with a .cpp extension
- Compile the program using the menu command or Press F9
- If no errors, run the program using menu command or Press F10

Output of the Program

- Enter first integer
- 45
- Enter second integer
- 72
- Sum is 117

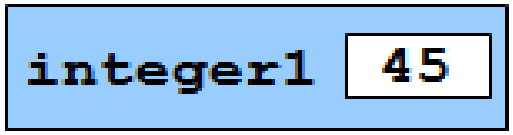
Memory Concepts

- Variable names
  - Correspond to actual locations in computer's memory
  - Every variable has name, type, size and value
  - When new value placed into variable, overwrites previous value
  - Reading variables from memory nondestructive

Memory Concepts

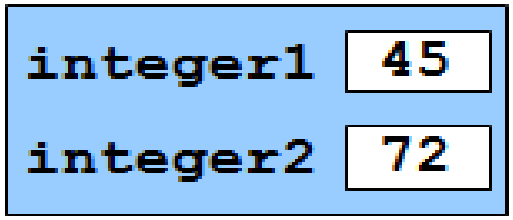
```
std::cin >> integer1;
```

– Assume user entered 45

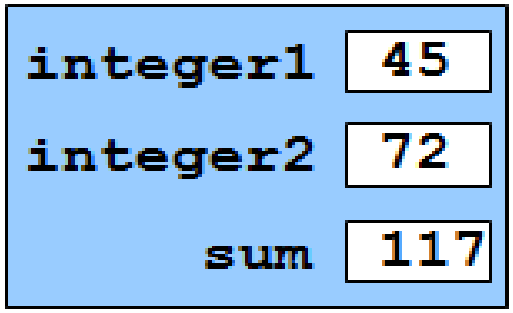



```
std::cin >> integer2;
```

– Assume user entered 72




```
sum = integer1 + integer2;
```



Arithmetic Operators

- `*`
  - Multiplication
- `/`
  - Division
  - Integer division truncates remainder
    - `7 / 5` evaluates to 1
- `%`
  - Modulus operator returns remainder
    - `7 % 5` evaluates to 2
- `+`
  - Addition
- `-`
  - Multiplication

Operator Precedence

- Operators in parentheses evaluated first
  - Nested/embedded parentheses
    - Operators in innermost pair first
- Multiplication, division, modulus applied next
  - Operators applied from left to right
- Addition, subtraction applied last

Operator Precedence

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. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
`*`, `/`, or `%`	Multiplication Division Modulus	Evaluated second. If there are several, they re evaluated left to right.
`+` or `-`	Addition Subtraction	Evaluated last. If there are several, they are evaluated left to right.

A couple of Examples

If a = 6, b = 2, c = 5, d = 4, e = 8 & f = 3

Then find the values of following expressions

- Z = a + b * c / d – e % f;
- Z = (a + b) * ((c / d) – e) % f;
- Z = a + b * (c – d) + (e + (3 + 7 * 2)) – f;

Writing a Small Program

- Write a program that asks the user to enter two numbers, obtains the two numbers from the user and prints the sum, product, difference, and quotient of the two numbers.

Decision Making : Equality and Relational Operators

Standard algebraic equality operator or relational operator	C++ equality or relational operator	Example of 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`

Decision Making : Equality and Relational Operators

- `if` structure
  - Make decision based on truth or falsity of condition
    - if condition met, body executed
    - else, body not executed
- Equality and relational operators
  - Equality operators
    - Same level of precedence
  - Relational operators
    - Same level of precedence
  - Associate left to right

```
1  // Using std
2  // Using if statements, relational
3  // operators, and equality operators.
4  #include <iostream>
5
6  using std::cout; // program uses cout
7  using std::cin; // program uses cin
8  using std::endl; // program uses endl
9
10 // function main begins program execution
11 int main()
12 {
13    int num1;  // first number to be read from user
14    int num2; // second number to be read from user
15
16    cout << "Enter two integers, and I will tell you\n"
17         << "the relationships they satisfy: ";
18    cin >> num1 >> num2;   // read two integers
19
20    if ( num1 == num2 )
21       cout << num1 << " is equal to " << num2 << endl;
22
23    if ( num1 != num2 )
24       cout << num1 << " is not equal to " << num2 << endl;
25
```

using statements eliminate need for **std::** prefix.

Declare variables.

Can write **cout** and **cin** without **std::** prefix.

if structure compares values of **num1** and **num2** to test for equality.

If condition is true (i.e., values are equal), execute this statement.

if structure compares values of **num1** and **num2** to test for inequality.

If condition is true (i.e., values are not equal), execute this statement.

```
26     if ( num1 < num2 )
27        cout << num1 << " is less than " << num2 << endl;
28
29     if ( num1 > num2 )
30        cout << num1 << " is greater than " << num2 << endl;
31
32     if ( num1 <= num2 )
33        cout << num1 << " is less than or equal to "
34             << num2 << endl;
35
36     if ( num1 >= num2 )
37        cout << num1 << " is greater than or equal to "
38             << num2 << endl;
39
40     return 0;   // indicate that program ended successfully
41
42  } // end function main
```

Statements may be split over several lines.

```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

Values for first run

```
Enter two integers, and I will tell you
the relationships they satisfy: 7 7
7 is equal to 7
7 is less than or equal to 7
7 is greater than or equal to 7
```

Values for second run

Another Program

- Write a program that reads an integer and determines and prints whether it is odd or even.

- Hint:
- Use the modulus operator. An even number is a multiple of two. Any multiple of two leaves a remainder of zero when divided by 2