

Introduction to Programming (in C++)

Data and Statements

Professors de Pro1

Dept. of Computer Science, UPC

- Variables, data types and expressions
- Statements:
 - Assignment
 - Input/output
 - Conditional statement

Variables, Data Types and Expressions

Variables and Literals

- **Variable:** symbolic name to represent data values.
 - A variable is usually associated with a memory location.
 - Think of a variable as a **box** containing values of a certain type.
 - In C++, variables must be **declared** before they are used.
- **Literal:** a constant of a certain type.
 - Examples: -4, 3.14159, 4.1e-8, true, "Greenland"

int x

42

bool z

true

double pi

3.14159

- A **data type** specifies:
 - The set of values that data of that type can have
 - The type of operations that can be performed with the data
- Every programming language has a set of basic data types.
- Basic data types in C++:

Type	Examples
<code>int</code>	<code>..., -2, -1, 0, 1, 2, ...</code>
<code>double</code>	<code>3.14, -0.5, 2.718</code>
<code>bool</code>	<code>true, false</code>
<code>char</code>	<code>'a', 'Z', '3'</code>
<code>string</code>	<code>"Hello", "UPC"</code>

Definition

An **expression** is a combination of literals, variables, operators and functions that is evaluated and returns a value.

Arithmetic expressions

`a + 3*(i - 1)` (int)
`sqrt(x)*log(4*n)` (double)

Boolean expressions

`(i - 3) <= x` (bool)
`(a != b) and (s <= "abc")`

Statements

- Any programming language has a set of basic **statements** to manipulate data (read, write and transform).
- A program consists of a combination of data and statements to perform some task.
- A program can become a new statement (**function**) that can be used in other programs.

Assignment

Syntax

```
variable = expression;
```

- **Semantics:**

- The value of the expression is evaluated
- The result is stored in the variable
- The previous value of the variable is **lost**

Variable initialization

```
double pi = 3.14159;  
double two_pi = 2*pi;  
string my_name = "Jordi";
```

Recommendation

Declare and initialize variables at the point they are first needed.

Assignment – Example

```
int x, i, j;  
// x=3, i=8, j=-2  
x = 3 + i + j;  
// x=9, i=8, j=-2  
x = x - i;  
// x=1, i=8, j=-2  
j = 0;  
// x=1, i=8, j=0
```

Swapping Two Variables

Incorrect solution

```
// Pre:  x=X, y=Y
x = y;
y = x;
// Post: x=Y, y=Y (WRONG!)
```

Correct solution

```
// Pre:  x=X, y=Y
int z = x;
x = y;
y = z;
// Post: x=Y, y=X
```

Swap with only two variables

```
x = x - y;    // x=X-Y, y=Y
y = x + y;    // x=X-Y, y=X
x = y - x;    // x=Y,   y=X
```

Basic I/O in C++

- `cin` and `cout` represent the program's default input and output devices (usually keyboard and display).

Simple I/O operations

```
// Read and store in a variable
cin >> variable;

// Write the value of an expression
cout << expression;
```

Example

```
int x, y;  double z;
cin >> x >> y >> z;
cout << x*y << ", " << z+1 << endl;
// Output for input "3 -4 2.75":    -12, 3.75
```

Division and Remainder

```
// Input:  reads two integer numbers (a, b)
// Output: writes the division and remainder of a/b
int main() {
    int a, b;
    cin >> a >> b;
    cout << "Division: " << a/b
         << ", Remainder: " << a%b << endl;
}
```

Revisiting time decomposition

```
// Reads an integer N >= 0 (seconds)
// Writes hours, minutes, seconds
int main() {
    int N;
    cin >> N;
    int s = N%60;
    N = N/60;
    cout << N/60 << " " << N%60 << " " << s << endl;
}
```

Conditional Statement

Conditional Statement

Syntax

```
if (condition) statement1;  
else statement2;
```

- condition is a **Boolean expression**
- **Semantics:** if the condition evaluates true, statement1 is executed; otherwise statement2 is executed.
- The else part is optional.

Maximum of two numbers

```
// Pre:  a=A, b=B  
if (a >= b) m = a;  
else m = b;  
// Post: m = max(A, B)
```

Blocks of Statements

```
int a, b, minimum, maximum;  
// Pre:  a=A, b=B  
// Post: minimum=min(A,B), maximum=max(A,B)  
if (a >= b) {  
    minimum = b;  
    maximum = a;  
}  
else {  
    minimum = a;  
    maximum = b;  
}
```

- Blocks of statements are enclosed inside { ... }

Maximum of Three Numbers

Approach I (nested if)

```
if (a >= b) {  
    if (a >= c) m = a;  
    else m = c;  
}  
else {  
    if (b >= c) m = b;  
    else m = c;  
}
```

Approach II (else if)

```
if (a >= b and a >= c)  
    m = a;  
else if (b >= c)  
    m = b;  
else m = c;
```

Approach III (update)

```
if (a >= b) m = a;  
else m = b;  
// m = max(a,b)  
if (c > m) m = c;
```