Introduction to Programming (in C++)

Data and statements

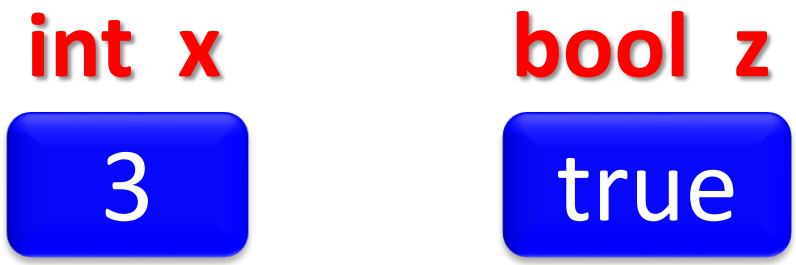
Jordi Cortadella, Ricard Gavaldà, Fernando Orejas
Dept. of Computer Science, UPC

Outline

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

Variables and literals

- Variable: symbolic name to represent data values.
- A variable is usually associated with a memory location.
- Intuition: think of a variable as a box containing values of a certain type.



- In C++ (and many other languages), variables must be declared before they are used.
- Literal: a constant of a certain type.
  - Examples: **-4**, **3.14159**, **4.1e-8**, ***true***, **"Greenland"**

Types

- 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++: int, double, bool, char, string, …

Expressions

- Expression: a combination of literals, variables, operators and functions that is evaluated and returns a value

- Examples

```
a + 3*(i - 1)
sqrt(x)*log(4*n)
(i - 3) <= x
(a != b) and (s <= "abc")
```

STATEMENTS

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

- *Assignment* is the fundamental statement of *imperative* languages:

⟨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

Assignment

Examples

```
int x, i, j;
    ...

// x=3, i=8, j=-2

x = 3*i + j;

// x=22, i=8, j=-2

x = x - i;

// x=14, i=8, j=-2

j = 0;

// x=14, i=8, j=0
```

Variable initialization

- Variables can be initialized with an expression in their declaration:

  double pi = 3.14159;
  double two_pi = 2∗pi;
  string my_name = "Jordi";

- Recommendation: declare the variables when needed (not before). Initialize the variable in the same declaration whenever possible.

Sequence of statements

- A sequence of statements (not necessarily assignments) is executed sequentially:

```
statement_1;
statement_2;
. . .
statement_n;
```

Example: swapping the value of two variables

Solution 1

```
int x, y;

// Precondition: x=X, y=Y
x = y;
y = x;
// Postcondition: x=Y, y=X
```

?

- Why is this solution incorrect?

Solution 2

```
int x, y;

// Precondition: x=X, y=Y
int z = x;
x = y;
y = z;
// Postcondition: x=Y, y=X
```

- A temporary variable is required

Swapping two integers with only two variables

```
// Pre:  x=A, y=B

x = x - y;

// x=A-B, y=B

y = x + y;

// x=A-B, y=A

x = y - x

// Post: x=B, y=A
```

Basic I/O in C++

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

- Simple operations:

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

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

Examples of I/O in C++

```
#include <iostream>
using namespace std;
...
int x, y;
double z;
...
cin >> x >> y >> z;
cout << x*y << z + 1 << endl;
...
```

```
> in_out
3 -4 2.75
-123.75
```

Examples of I/O in C++

```
#include <iostream>
using namespace std;
...
int x, y;
double z;
...
cin >> x >> y >> z;
cout << x*y << ", " << z+1 << endl;
...
```

Some aesthetic formatting is usually required

```
> in_out
3 -4 2.75
-12, 3.75
```

Quotient and remainder

```
// Input:  reads two integer numbers (a, b)
// Output: writes the quotient and remainder
//         of a/b

int main() {
    int a, b;
    cin >> a >> b;
    cout << “Quotient: " << a/b
         << ", Remainder: " << a%b << endl;
}
```

Revisiting time decomposition

```
// Input:  reads an integer N >= 0 that represents
//         a certain time in seconds
// Output: writes the decomposition of N in
//         hours (h), minutes (m) and seconds (s)
//         such that 0 <= m < 60 and 0 <= s < 60.

int main() {
    int N;
    cin >> N;
    int s = N%60;
    N = N/60;
    cout << N/60 << " " << N%60 << " " << s << endl;
}
```

Conditional statement

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

- ⟨condition⟩ is a Boolean expression
- Semantics: if the condition evaluates *true*, ⟨statement1⟩ is executed, otherwise ⟨statement2⟩ is executed.

Conditional statement: example

```
int a, b, m;
...
// Calculation of the maximum of two numbers
// Pre: a=A, b=B

if (a >= b) m = a;
else m = b;

// Post: a=A, b=B, m=max(A,B)
```

The *else* part is optional

```
// Input:  reads an integer number
// Output: writes the absolute value
//         of the number

int main() {
    int a;
    cin >> a;
    if (a < 0) a = -a;
    cout << a << endl;
}
```

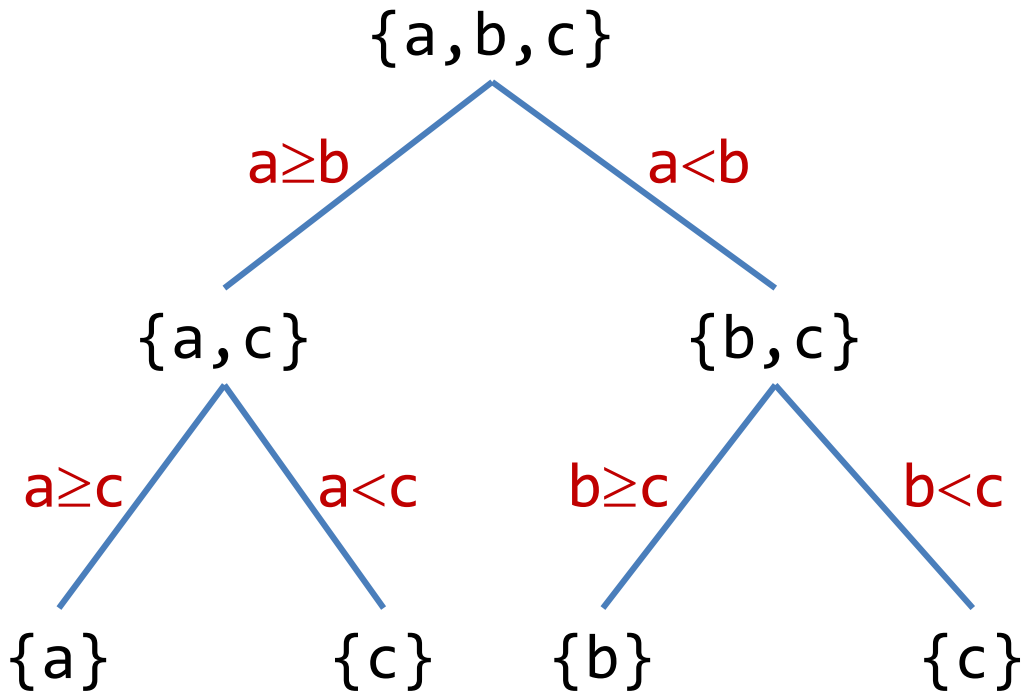
Min and max of two numbers

```
int a, b, minimum, maximum;

// Pre:  a=A, b=B
// Post: a=A, b=B,
//       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 { … }

Max of three numbers (I)

```
int a, b, c, m;
// Pre:  a=A, b=B, c=C
// Post: a=A, b=B, c=C, m=max(A,B,C)
```



Decision tree

Max of three numbers (I)

```
int a, b, c, m;

// Pre:  a=A, b=B, c=C
// Post: a=A, b=B, c=C, m=max(A,B,C)

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

Max of three numbers (II)

```
int a, b, c, m;

// Pre:  a=A, b=B, c=C
// Post: a=A, b=B, c=C, m=max(A,B,C)

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

Max of three numbers (III)

```
int a, b, c, m;

// Pre:  a=A, b=B, c=C
// Post: a=A, b=B, c=C, m=max(A,B,C)

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