Introdução à Programação 2020/2021

António J. R. Neves
Daniel Corujo

Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

an@ua.pt / dcorujo@ua.pt

http://elearning.ua.pt/

Topics

- Values and types
- Variables
- Keywords
- Operators, Expressions and statements
- Console input/output

- Programs are composed of statements. Statements are terminated with a semi-colon (;), and are collected in sections known as functions
- If possible, a statement should be kept on its own line
- The simple use of indents and line breaks can greatly improve code readability without impacting code performance (in most cases they are discarded by the compiler)
- Blank lines should be used to offset the main components of your code
- All code inside a new block should be indented by one tab more than the code in the enclosing block
- Comments in code can be useful for a variety of purposes: single line: `//...` or multi line: `/* ... */`

- A very simple C++ program

```
#include <iostream>
using namespace std;
int main() {
    cout << "Every age has a language of its own\n";
    return 0;
}
```

- `main` function, statements, whitespaces, string literals (constants), preprocessor directives, header files, comments, - same as C

- A C++ program can be divided into different namespaces. The directive `using namespace std;` says that all the program statements that follow are within the std namespace

- A **value** is one of the basic things a program works with, like a letter or a number (`33, 3.14, 'a'`).
- In Standard C++ there are several basic data types. e.g. **bool**, **char**, **short**, **int**, **long**, **float** and **double**.
- The integer types (**char**, **short**, **int** and **long**) store integers
- The **char** type is capable of holding any member of the execution character set.
- In 64-bit processors, a **char** can hold 8-bit numbers, a **short** can hold 16-bit numbers, an **int** can hold 32-bit numbers and a **long** can hold 64-bit
- The floating point types (**float** and **double**) store inexact representations of real numbers (floats have 32 bits and doubles have 64 bits).
- **bool** will be described later.

- Variables are simply names used to refer to some location in memory – a location that holds a value
- An **assignment statement** creates a new variable and gives it a value.

```
int n = 17;
double pi = 3.1416;
```

- Variable names are made up of letters (upper and lower case) and digits. The underscore character (`'_'`) is also permitted. Names must not begin with a digit.
- Multiple variables can be declared (with or without initialization) with one statement

```
int n, j = 3, i, k;
```

- In reality, **`long`** and **`short`** are modifiers that make it possible for a data type to use either more or less memory
- When the **`const`** qualifier is used, the declared variable must be initialized at declaration. It is then not allowed to be changed
- `#define` versus `const` (it is possible to use the preprocessor directive #define to define a constant)
- **`static`** can be used on functions and variables (for later...)
- several others...

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

- CPP Reference
- This is a list of reserved keywords in C++. Since they are used by the language, these keywords are not available for re-definition or overloading

A - C	D - P	R - Z
alignas (since C++11)	decltype (since C++11)	reflexpr (reflection TS)
alignof (since C++11)	default (1)	register (2)
and	delete (1)	reinterpret_cast
and_eq	do	requires (since C++20)
asm	double	return
atomic_cancel (TM TS)	dynamic_cast	short
atomic_commit (TM TS)	else	signed
atomic_noexcept (TM TS)	enum	sizeof (1)
auto (1)	explicit	static
bitand	export (1) (3)	static_assert (since C++11)
bitor	extern (1)	static_cast
bool	false	struct (1)
break	float	switch
case	for	synchronized (TM TS)
catch	friend	template
char	goto	this
char8_t (since C++20)	if	thread_local (since C++11)
char16_t (since C++11)	inline (1)	throw
char32_t (since C++11)	int	true
class (1)	long	try
compl	mutable (1)	typedef
concept (since C++20)	namespace	typeid
const	new	typename
consteval (since C++20)	noexcept (since C++11)	union
constexpr (since C++11)	not	unsigned
constinit (since C++20)	not_eq	using (1)
const_cast	nullptr (since C++11)	virtual
continue	operator	void
co_await (since C++20)	or	volatile
co_return (since C++20)	or_eq	wchar_t
co_yield (since C++20)	private	while
	protected	xor
	public	xor_eq

- **Operators** are special symbols that represent computations (for example, `+`, `-`, `*`, `/`, `%`, `<=`).
- The values combined by operators are called **operands**.
- For a given operator, operands must have compatible types. The result type depends on the operand types.
- An **expression** is a combination of values, variables, and operators.
- A **statement** is a unit of code (like a phrase in a written language).
- When more than one operator appears in an expression, the *order of evaluation* depends on the rules of precedence.
- Use parentheses to make it obvious!

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

Common operators

assignment	increment decrement	arithmetic	logical	comparison	member access	other

`a = b`
`a += b`
`a -= b`
`a *= b`
`a /= b`
`a %= b`
`a &= b`
`a |= b`
`a ^= b`
`a <<= b`
`a >>= b`
	`++a`
	`--a`
	`a++`
	`a--`
		`+a`
		`-a`
		`a + b`
		`a - b`
		`a * b`
		`a / b`
		`a % b`
		`~a`
		`a & b`
		`a | b`
		`a ^ b`
		`a << b`
		`a >> b`
			`!a`
			`a && b`
			`a || b`
				`a == b`
				`a != b`
				`a < b`
				`a > b`
				`a <= b`
				`a >= b`
				`a <=> b`
					`a[b]`
					`*a`
					`&a`
					`a->b`
					`a.b`
					`a->*b`
					`a.*b`
						`a(...)`
						`a, b`
						`? :`

Special operators

`static_cast` converts one type to another related type
`dynamic_cast` converts within inheritance hierarchies
`const_cast` adds or removes cv qualifiers
`reinterpret_cast` converts type to unrelated type
C-style cast converts one type to another by a mix of `static_cast`, `const_cast`, and `reinterpret_cast`
`new` creates objects with dynamic storage duration
`delete` destructs objects previously created by the new expression and releases obtained memory area
`sizeof` queries the size of a type
`sizeof...` queries the size of a parameter pack (since C++11)
`typeid` queries the type information of a type
`noexcept` checks if an expression can throw an exception (since C++11)
`alignof` queries alignment requirements of a type (since C++11)

https://en.cppreference.com/w/cpp/language/expressions

- The identifier `cout` is an object of `ostream` class predefined in C++ to correspond to the standard output stream
- A stream is an abstraction that refers to a flow of data
- The standard output stream normally flows to the screen display
- The operator << is called the *insertion* or *put to* operator
- It directs the contents of the variable on its right to the object on its left
- The compiler determines the data type of variable to be output and selects the appropriate stream insertion operator to display the value
- The << operator is overloaded to output data items of built-in types `integer, float, double, strings, ...`

- The `ios` class contains the majority of the features you need to operate C++ streams
- The three most important features are the formatting flags, the error-status flags, and the file operation mode
- **Formatting flags** are a set of enum definitions in `ios` - they act as on/off switches that specify choices for various aspects of input and output format and operation (for example, `std::left`, `std::right`, ...)
- https://en.cppreference.com/w/cpp/io/manip
- Manipulators are helper functions that make it possible to control input/output streams using operator << or operator >> (eg. `setw()`, `setfill()`, ...)

```
std::cout << "Left fill:\n"
          << std::left
          << std::setfill('*')
          << std::setw(12) << 123  << '\n';

std::cout << "Right fill:\n"
          << std::right
          << std::setw(12) << 123  << '\n';
```

Output:

```
Left fill:
123*********
Right fill:
*********123
```

- The identifier `cin` is an object of `istream` class, predefined in C++ to correspond to the standard input stream
- This stream represents data coming from the keyboard
- The `>>` is the *extraction* or *get from* operator
- It takes the value from the stream object on its left and places it in the variable on its right

```
cin >> ftemp; //the program wait for the user to type
```

- The insertion operator can be used repeatedly in the same statement allowing the user to enter a series of values

```
cin >> n >> d; // wait for two things
```

- The compiler determines the data type of the entered value and selects the appropriate stream extraction operator to extract the value and store it in the given variables.

- https://en.cppreference.com/w/cpp/numeric

- **`cstdlib`** provides miscellaneous utilities. This header was originally in the C standard library as **`stdlib.h`**. Symbols defined here are used by several library components (e.g. **`abs`**, **`labs`**, **`div`**, **`...`**)

- **`cmath`** header is part of the numeric library. It allows the use of several mathematical operations (**`sqrt`**, **`log`**, **`exp`**, **`sin`**, **`cos`**, ...).

A complete example in C++

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

```
#include <iostream>
#include <cmath>

using namespace std;

int main() {
    double x;

    cout << "Give me a number: ";
    cin >> x;
    cout << "The square root of " << x << " is " <<
sqrt(x) << endl;
    return 0;
}
```