PSL

ECUE apprentissage de la programmation - the c++ language

Valérie Roy et Basile Marchand
Mines ParisTech

Plan

1. the core-language
   - The program entry point

Plan

1. the core-language
   The program entry point

The program entry point

the unique entry point of a program is the `main` function

a **program** shall contain a **global** and **unique function** called `main`

it is the **start** point of the **program**

it **must return** an integer, otherwise it is **implementation-defined** [a]

a. i.e. « defined by the compiler you are using » ≡ non-portable

the simpliest definition of the `main` function is :

```
int main () {
  /* ... */
  return 0;
}
```

What the function `main` should return

a classical `main` function returns the value `0`

note that the c++ examples of those slides will **not** always `return 0` for the sake of **slides' readibility**

the function `main` cannot be called within a program

there can only be one `main` function in an executable

example of a program

file toto.cxx

```
#include <iostream>
int main () {
  std::cout << "Hello World !";
  return 0;
}
```

lines begining by # contain directives to the c++ **pre-processor**

`#include <iostream>` asks the c++ pre-processor to **include** here the whole content of the file `iostream.h`

`<iostream>` is a part of the c++ standard library that deals with input output streams

for example, `<iostream>` defines the standard output stream `cout` [a]

a. here `std::cout` because of the standard library namespace

Note that the standard library is defined in a **namespace** named `std`

Note that a **first** way to program is by using the c++ standard library

the preprocessing language [1]

preprocessing directives are **lines** starting with #

consists of **directives** to be executed and **macros** to be expanded

to cover more than one line, a preprocessing directive must end by \

preprocessing language is mainly used for

- inclusion of header files (by substitution)
- macro expansion (abbreviations for c-code fragments)
- conditional compilation (to include/exclude parts of program according to conditions)

1. https ://gcc.gnu.org/onlinedocs/cpp/

edit, compile and execute a program (1)

open a **file** with an editor (gedit, emacs, vim, ...) and **type** this program

file toto.cxx

```
#include <iostream>
int main () {
  std::cout << "Hello World !";
  return 0;
}
```

open a **terminal** on your operating system and **compile** and **link** your program

here you obtain an executable named `hello`

c++ 2003

```
g++ toto.cxx -o hello
```

c++ 11 (you might need a command-line parameter)

```
g++ -std=c++11 toto.cxx -o hello
```

call the **executable** (note that $$ is the prompt on the terminal)

```
$$ ./hello
Hello World !
```

passing arguments to a program

note that you can pass *command-line* **arguments** to your program when you are calling the **executable**

file toto.cxx

```
#include <iostream>
int main () {
  std::cout << "Hello␣World␣!";
  return 0;
}
```

```
g++ -std=c++11 toto.cxx -o hello
```

```
$$ ./hello 1 Hello 17.9
```

```
Hello World !
```

another definition of the `main` function

you can access command-line arguments from your program

```
int main (int argc, char* argv []) { /* ... */ }
```

`argc-1` is the **number** of **arguments passed** to the **program**

arguments are **supplied** in the array `argv` from `argv[0]` to `argv[argc-1]`

the `argv` array contains **character strings**

note that argv[0] is the name used to invoke the program

print the command-line arguments in `main`

print the number of arguments `argc-1` and the name of the executable `argv[0]`

```
#include <iostream>
int main (int argc, char* argv []) {
  std::cout << "the name of the program is: " << /* your code here */;
  std::cout << "the number of arguments is: " << /* your code here */;
}
```

```
g++ -std=c++11 toto.cxx -o hello
```

```
$$ ./hello 1 2 3
the name of the program is: ./hello
the number of arguments is: 3
```

this is the correction :

print the command-line arguments in `main`

print the number of arguments `argc-1` and the name of the executable `argv[0]`

```
#include <iostream>
int main (int argc, char* argv []) {
  std::cout << "the name of the program is: " << /* your code here */;
  std::cout << "the number of arguments is: " << /* your code here */;
}
```

```
g++ -std=c++11 toto.cxx -o hello
```

```
$$ ./hello 1 2 3
the name of the program is: ./hello
the number of arguments is: 3
```

this is the correction :

```
#include <iostream>
int main (int argc, char* argv[]) {
  std::cout << "the name of the program is: " << argv[0] << '\n';
  std::cout << "the number of arguments is: " << argc - 1;
  return 0;
}
```