ECUE apprentissage de la programmation - the c++ language

Valérie Roy et Basile Marchand
Mines ParisTech

Plan

Plan

expression statements

a statement can be a single **expression**

```
int main () {
   10 + 2;
   return 0;
}
```

the **expression** is **evaluated** and its **value** is **discarded** after the ";"

side-effects (if any) are completed **before** the next statement is executed

```
int main () {
  int i = 1;
  ++i + 1; // i++ increments i and returns the value of i before increment
  i++ + 1; // ++i increments i and returns the value of i after increment
  return 0;
}
```

statements can be function calls, assignments, definitions, ...

an expression is evaluated

if during the **evaluation** of an expression

the **result** is not **mathematically defined**

or if the **result** is outside the **range** of representable values for its type

then the behavior is **undefined**

compound statements

a **compound** statement is enclosed by **brackets** { and }

```
int main () {
  int i = 12;
  { int j = i;
    { int k = j;
    } // k does not exist any more
  } // j does not exist any more
}
```

you can define **blocks** and **nested** blocks

do not forget to indent your code ! (verify your indentation with *syntax-oriented features* of your editor

```
// bad indentation
#include <iostream>
void foo (int v) {
if (v != 0)
v = v + 12;
std::cout << v;
}
```

```
#include <iostream>
void foo (int v) {
   if (v != 0)
      v = v + 12;
      // wrong indentation
      std::cout << v;
}
```

```
#include <iostream>
void foo (int v) {
   if (v != 0)
      v = v + 12;
   // good indentation
   std::cout << v;
}
```

the `if` selection statements

`if` (*condition*) *one statement* (the condition is implicitly converted to `bool`)

```
int i = 12;
if (i < 17)
   // the then part
   // (a unique statement)
   foo();
```

```
int i = 12;
if (i < 17) {
   // the then part
   // (a block)
   foo();
}
```

`if` (*condition*) *one statement* `else` *one statement*

```
int i = gee();
if (i < 17)
   foo();
else
   // the else part
   bar();
```

```
int i = gee();
if (i < 17) {
   foo();
} else {
   // the else part
   bar();
}
```

the switch Selection Statement

switch (condition) statement

used to compare an **integral** variable to a list of integral values

the variable is compared, in **sequence**, to the values following the case label

when one matches : the computer executes the case part, then it continues ...

```
switch (value) {
case v0:
  // some code
  break;
case v1:
  // some code
  break;
default:
  // some code
}
```

if you want to stop at the first match, use break

the while iteration statements

while (condition) statement

```
#include <iostream>
int main () {
  char c = '\0';
  while (c != 'q') {
    std::cin >> c;
    std::cout << "i␣am␣a␣" << c
              << std::endl;
  }
  return 0;
}
```

for iteration statements

`for ( initialization ; continuation condition ; expression )`

```
#include <iostream>
int main () {
  int tab[4] = {10, 20, 30, 40};
  for (int i = 0; i < 4; i++)
    std::cout << tab[i] << '␣';
  std::cout << std::endl;
  return 0;
}
```

```
int i = 0;
for (;i < 12; ++i) {...}

for (;;++i) {...} // forever loop
for(;;) {...}
```

the for-range statement

c++11 has introduced a very **convenient** *python-like* `for-range`

`for ( range : range initializer ) statement`

it iterates through a container

```
#include <iostream>
int main () {
   int tab [10] = {10, 9, 8, 7,
                   6, 5, 4, 3, 2, 1};
   for (int e : tab)
      std::cout << e << '␣';
   return 0;
}
```