CPPLI : TD 6 : C++ : Fonctions

Nicolas Vansteenkiste Romain Absil Jonas Beleho *
(Esi – He2b)

Année académique 2019 – 2020

Ce TD [1] aborde l'étude des fonctions [2] en C++ [3].

Ex. 6.1 Écrivez la fonction de prototype :

```
bool isPrime(unsigned number);
```

Elle retourne `true` [4] ou `false` selon que son argument est un nombre premier [5] ou non. Répartissez prototype et code dans les fichiers `mathesi.h` et `mathesi.cpp`.

Ex. 6.2 Arrangez-vous pour produire, à l'aide de la fonction `isPrime(unsigned)` de l'Ex. 6.1, la sortie console suivante :

```
Les nombres premiers entre 200 et 349 :
   .   .   .   .   .   .   .   .   .   .
   . 211   .   .   .   .   .   .   .   .
   .   .   . 223   .   .   . 227   . 229
```

*Et aussi, lors des années passées : Monica Bastreghi, Stéphan Monbaliu, Anne Rousseau et Moussa Wahid.

1. https://poesi.esi-bru.be/pluginfile.php/1320/mod_folder/content/0/td06_cpp/td06_cpp.pdf (consulté le 1er novembre 2019).
2. https://en.cppreference.com/w/cpp/language/functions (consulté le 1er novembre 2019).
3. https://fr.wikipedia.org/wiki/C%2B%2B (consulté le 1er novembre 2019).
4. https://en.cppreference.com/w/cpp/language/bool_literal (consulté le 1er novembre 2019).
5. https://en.wikipedia.org/wiki/Prime_number (consulté le 1er novembre 2019).

```
  .   .   . 233   .   .   .   .   . 239
  . 241   .   .   .   .   .   .   .   .
  . 251   .   .   .   .   . 257   .   .
  .   .   . 263   .   .   .   .   . 269
  . 271   .   .   .   .   . 277   .   .
  . 281   . 283   .   .   .   .   .   .
  .   .   . 293   .   .   .   .   .   .
  .   .   .   .   .   .   . 307   .   .
  . 311   . 313   .   .   . 317   .   .
  .   .   .   .   .   .   .   .   .   .
  . 331   .   .   .   .   . 337   .   .
  .   .   .   .   .   .   . 347   . 349
```

Pour la mise en forme, utilisez les manipulateurs de flux [6] !

Ex. 6.3 Écrivez la fonction de prototype :

```
std::pair<int, int> euclidianDivision(int dividend, int divisor);
```

Elle calcule la division euclidienne [7] du dividende `dividend` par le diviseur `divisor`. Le champ `first` de la `std::pair` [8] retournée est la division entière de `dividend` par `divisor` ; son champ `second` correspond à leur modulo.

Répartissez prototype et code dans les mêmes fichiers `mathesi.h` et `mathesi.cpp` que ceux de l'Ex. 6.1.

Rem. : Que faire si `divisor` est nul ? Levez une exception du type `std::domain_error` [9] avec un message adéquat.

Ex. 6.4 Arrangez-vous pour produire, à l'aide de la fonction de l'Ex. 6.3, la sortie console suivante :

```
euclidianDivision : division by zero
27 = 27 *  1 +  0
27 = 13 *  2 +  1
27 =  9 *  3 +  0
27 =  6 *  4 +  3
27 =  5 *  5 +  2
27 =  4 *  6 +  3
27 =  3 *  7 +  6
27 =  3 *  8 +  3
27 =  3 *  9 +  0
27 =  2 * 10 +  7
27 =  2 * 11 +  5
```

6. https://en.cppreference.com/w/cpp/io/manip (consulté le 1er novembre 2019).
7. https://fr.wikipedia.org/wiki/Division_euclidienne (consulté le 1er novembre 2019).
8. https://en.cppreference.com/w/cpp/utility/pair (consulté le 1er novembre 2019).
9. https://en.cppreference.com/w/cpp/error/domain_error (consulté le 1er novembre 2019).

```
27 =  2 * 12 +  3
27 =  2 * 13 +  1
27 =  1 * 14 + 13
27 =  1 * 15 + 12
27 =  1 * 16 + 11
27 =  1 * 17 + 10
27 =  1 * 18 +  9
27 =  1 * 19 +  8
27 =  1 * 20 +  7
27 =  1 * 21 +  6
27 =  1 * 22 +  5
27 =  1 * 23 +  4
27 =  1 * 24 +  3
27 =  1 * 25 +  2
27 =  1 * 26 +  1
27 =  1 * 27 +  0
```

Ex. 6.5 Pour chaque ligne de la fonction `main()` du source surcharge_01.cpp[10], dites si elle compile ou non. Si non, dites pourquoi ; si oui, dites quel affichage est alors produit.

```cpp
/*!
 * \file surcharge_01.cpp
 * \brief surcharge de fonctions
 */
#include <iostream>

void f(int) {
    std::cout << "f(int)" << std::endl;
}

int main() {
    f(3);
    f(4.5);
    f(true);
    f(3LL);
    f('T');
    f(.3F);
    f(5U);
    short s {44};
    f(s);
```

10. https://poesi.esi-bru.be/pluginfile.php/1320/mod_folder/content/0/td06_cpp/surcharge_01.cpp (consulté le 1er novembre 2019).

```
21      f(2e-2L);
22  }
```

Répondez à cette question d'abord sur papier, puis confrontez vos réponses aux résultats obtenus avec `gcc`.

Rappelez-vous la forme en C++ des littéraux entiers [11], flottants [12], booléens [13] et caractères [14], ainsi que les règles de choix [15] de fonction en cas de surcharge de fonction (*function overload*).

Ex. 6.6 Pour chaque ligne de la fonction `main()` du source `surcharge_02.cpp` [16], dites si elle compile ou non. Si non, dites pourquoi ; si oui, dites quel affichage est alors produit.

```
1   /*!
2    * \file surcharge_02.cpp
3    * \brief surcharge de fonctions
4    */
5   #include <iostream>
6
7   void f(int) {
8       std::cout << "f(int)" << std::endl;
9   }
10
11  void f(long double) {
12      std::cout << "f(long double)" << std::endl;
13  }
14
15  int main() {
16      f(3);
17      f(4.5);
18      f(true);
19      f(3LL);
20      f('T');
21      f(.3F);
```

11. https://en.cppreference.com/w/cpp/language/integer_literal (consulté le 1er novembre 2019).
12. https://en.cppreference.com/w/cpp/language/floating_literal (consulté le 1er novembre 2019).
13. https://en.cppreference.com/w/cpp/language/bool_literal (consulté le 1er novembre 2019).
14. https://en.cppreference.com/w/cpp/language/character_literal (consulté le 1er novembre 2019).
15. http://www-01.ibm.com/support/knowledgecenter/SSGH3R_13.1.2/com.ibm.xlcpp131.aix.doc/language_ref/implicit_conversion_sequences.html (consulté le 1er novembre 2019).
16. https://poesi.esi-bru.be/pluginfile.php/1320/mod_folder/content/0/td06_cpp/surcharge_02.cpp (consulté le 1er novembre 2019).

```
    f(5U);
    short s {44};
    f(s);
    f(2e-2L);
}
```

Répondez à cette question d'abord sur papier, puis confrontez vos réponses aux résultats obtenus avec **gcc**.

Ex. 6.7 Pour chaque ligne de la fonction `main()` du source `surcharge_03.cpp`[17], dites si elle compile ou non. Si non, dites pourquoi ; si oui, dites quel affichage est alors produit.

```
/*!
 * \file surcharge_03.cpp
 * \brief surcharge de fonctions
 */
#include <iostream>

void f(short) {
    std::cout << "f(short)" << std::endl;
}

void f(double) {
    std::cout << "f(double)" << std::endl;
}

int main() {
    f(3);
    f(4.5);
    f(true);
    f(3LL);
    f('T');
    f(.3F);
    f(5U);
    short s {44};
    f(s);
    f(2e-2L);
}
```

Répondez à cette question d'abord sur papier, puis confrontez vos réponses aux résultats obtenus avec **gcc**.

17. https://poesi.esi-bru.be/pluginfile.php/1320/mod_folder/content/0/td06_cpp/surcharge_03.cpp (consulté le 1er novembre 2019).

Ex. 6.8 Pour chaque ligne de la fonction `main()` du source `surcharge_04.cpp`[18], dites si elle compile ou non. Si non, dites pourquoi ; si oui, dites quel affichage est alors produit.

```cpp
/*!
 * \file surcharge_04.cpp
 * \brief surcharge de fonctions
 */
#include <iostream>

void f(int) {
    std::cout << "f(int)" << std::endl;
}

void f(unsigned) {
    std::cout << "f(unsigned)" << std::endl;
}

void f(long long) {
    std::cout << "f(long long)" << std::endl;
}

void f(double) {
    std::cout << "f(double)" << std::endl;
}

void f(long double) {
    std::cout << "f(long double)" << std::endl;
}

int main() {
    f(3);
    f(4.5);
    f(true);
    f(3LL);
    f('T');
    f(.3F);
    f(5U);
    short s {44};
    f(s);
    f(2e-2L);
```

18. https://poesi.esi-bru.be/pluginfile.php/1320/mod_folder/content/0/td06_cpp/surcharge_04.cpp (consulté le 1er novembre 2019).

```
38 }
```

Répondez à cette question d'abord sur papier, puis confrontez vos réponses aux résultats obtenus avec **gcc**.