

M1103 Amphi 1

Thomas Nowak

Université Paris-Sud

M1103 - Structures de données et algorithmes fondamentaux

Organisation du cours

Site web du cours

- <https://www.lri.fr/~nowak/teaching/algoprogram>

Cours sous Moodle

- <https://cours.iut-orsay.fr/course/view.php?id=301>
- Clé d'inscription : cléGroupe-1A, cléGroupe-1B, ...
- Rendus d'exercices, forum de questions, feedback

Programming C++

AN INTRODUCTION TO PROGRAMMING BY THE INVENTOR OF C++

SECOND EDITION

- ◆ *Preparation for Programming in the Real World*
The book assumes that you are eventually to write non-trivial programs, whether for work, in software development, or in some other technical field.
- ◆ *Focus on Fundamental Concepts and Techniques*
The book explains fundamental concepts and techniques in greater depth than traditional introductions. This approach will give you solid foundations for writing useful, correct, maintainable, and efficient code.
- ◆ *Programming with Today's C++ (C++11 and C++14)*
The book is an introduction to programming in general, including discussions of programming and generic programming. It is also a solid introduction to the C++ programming language, one of the most widely used languages for real-world software. The book presents modern C++ programming techniques from the start, introducing the C++ standard library and C++11 and C++14 features to enable programming tasks.
- ◆ *For Beginners—And Anyone Who Wants to Learn Something New*
The book is primarily designed for people who have never programmed before, and it has been tested with many thousands of learners across my students. It has also been extensively used for self-study. Also, practicing and advanced students have gained new insight and guidance by viewing how a novice approaches the elements of his art.
- ◆ *Provides a Broad View*
The first half of the book covers a wide range of essential concepts, design and programming techniques, language features, and libraries. There will enable you to write programs involving input, output, comparison, and simple graphics. The second half explores more specialized topics (such as test programming, strings, and the C programming language) and provides abundant reference material. Source code and support supplements are available from the author's website.

ABOUT THE AUTHOR
Bjarne Stroustrup is the designer and original implementer of C++ and the author of *The C++ Programming Language, Fourth Edition* (Addison-Wesley, 2013). He is a Managing Director in the Technology Division of Morgan Stanley, a Visiting Professor at Columbia University, a Distinguished Research Professor at Texas A&M University, and a member of the U.S. National Academy of Engineering. Before coming to academia, he worked for decades in AT&T Bell Labs. He is a founding member of the ISO C++ standards committee.

informa.com/wiley
informa.com/authoring
stroustrup.com/Programming

ISBN 978-0-201-92728-0
ISBN 978-0-201-92728-4

**45 DAYS FREE
ACCESS TO ONLINE EDITION
with purchase of this book**
Available on Last Page

9 780321 927280

\$74.99 U.S. • \$79.99 Canada

PEARSON

ALWAYS LEARNING

ADDISON WESLEY

BJARNE STROUSTRUP

THE CREATOR OF C++

Using
C++11
and
C++14

PROGRAMMING

Principles and Practice Using C++

PROGRAMMING

Principles and Practice Using C++

SECOND EDITION

STROUSTRUP

ADDISON WESLEY

Séances présentielles

- amphis : 1h/semaine
- TP : 2 × 2h/semaine
- récitations : 1h30/semaine

Exercices

- 1 mini-feuille d'exercices chaque semaine (≤ 20 minutes)
- 4 feuilles d'exercices
- tests automatiques fournis
- rendus sous Moodle

Notation

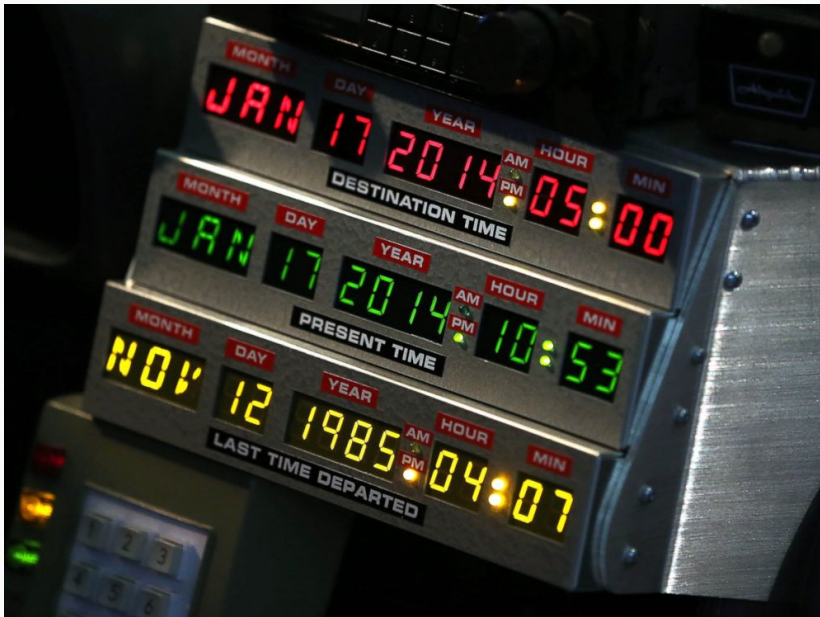
- 10% mini-feuilles
- 30% feuilles
- 60% exam final

Collaboration

- collaboration en groupes de 2 autorisée
 - écrire le nom du binôme dans *tout* fichier rendu
- discussion de votre code autorisée
 - ne jamais copier le code d'un.e autre étudiant.e
 - ne jamais écrire du code pour un.e autre étudiant.e

1. vecteurs
2. objets, classes
3. temps d'exécution
4. recherche, tri

1985 → 2014



Compiler settings

Global compiler settings



Global compiler settings



Profiler settings



Batch builds

Selected compiler

GNU GCC Compiler

Set as default

Copy

Rename

Delete

Reset defaults

Compiler settings

Linker settings

Search directories

Toolchain executables

Custom variables

Build options

Policy:

Compiler Flags

Other compiler options

Other resource compiler options

#defines

General

- | | |
|---|-------------------------------------|
| Have g++ follow the 1998 ISO C++ language standard [-std=c++98] | <input type="checkbox"/> |
| Have g++ follow the C++11 ISO C++ language standard [-std=c++11] | <input type="checkbox"/> |
| Have g++ follow the C++14 ISO C++ language standard [-std=c++14] | <input checked="" type="checkbox"/> |
| Have g++ follow the coming C++0x (aka c++11) ISO C++ language stan | <input type="checkbox"/> |
| Have g++ follow the coming C++1y (aka C++14) ISO C++ language star | <input type="checkbox"/> |
| Have g++ follow the coming C++1z (aka C++17) ISO C++ language star | <input type="checkbox"/> |
| Have gcc follow the 1990 ISO C language standard (certain GNU extensio | <input type="checkbox"/> |
| Have gcc follow the 1999 ISO C language standard [-std=c99] | <input type="checkbox"/> |
| Have gcc follow the 2011 ISO C language standard [-std=c11] | <input type="checkbox"/> |
| In C mode, this is equivalent to -std=c90, in C++ mode, it is equivalent to | <input type="checkbox"/> |
| Position Independent Code [-fPIC] | <input type="checkbox"/> |
| Static libgcc [-static-libgcc] | <input type="checkbox"/> |
| Static libstdc++ [-static-libstdc++] | <input type="checkbox"/> |

NOTE: Right-click to setup or edit compiler flags.

OK

Cancel

☰ General	
Have g++ follow the 1998 ISO C++ language standard [-std=c++98]	<input type="checkbox"/>
Have g++ follow the C++11 ISO C++ language standard [-std=c++11]	<input type="checkbox"/>
Have g++ follow the C++14 ISO C++ language standard [-std=c++14]	<input checked="" type="checkbox"/>
Have g++ follow the coming C++0x (aka c++11) ISO C++ language stan	<input type="checkbox"/>

Vecteurs

Tableau + taille

```
const int MAX = 100;
```

```
int tab[MAX] = {1, 2, 3, 4};
```

```
int nbE = 4;
```

- vecteur = tableau redimensionnable
- “connaît” sa propre taille
- possible de rajouter et de supprimer des éléments
- utilisé comme un tableau
- pas de nombre maximal d'éléments

```
vector<int> vec = {1, 2, 3, 4};
```


Vecteurs

```
#include <iostream>
#include <vector>

using namespace std;

int main()
{
    vector<int> vec = {1, 2, 3, 4};

    cout << "premier element : " << vec[0] << endl;
    cout << "taille de vec : " << vec.size() << endl;
}
```

Rajout d'un élément

- vec avant rajout :

1	2	3	4
---	---	---	---

- rajout :

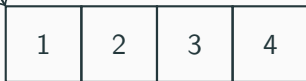
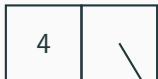
```
vec.push_back(5);
```

- vec après rajout :

1	2	3	4	5
---	---	---	---	---

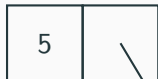
Vecteurs

taille tableau



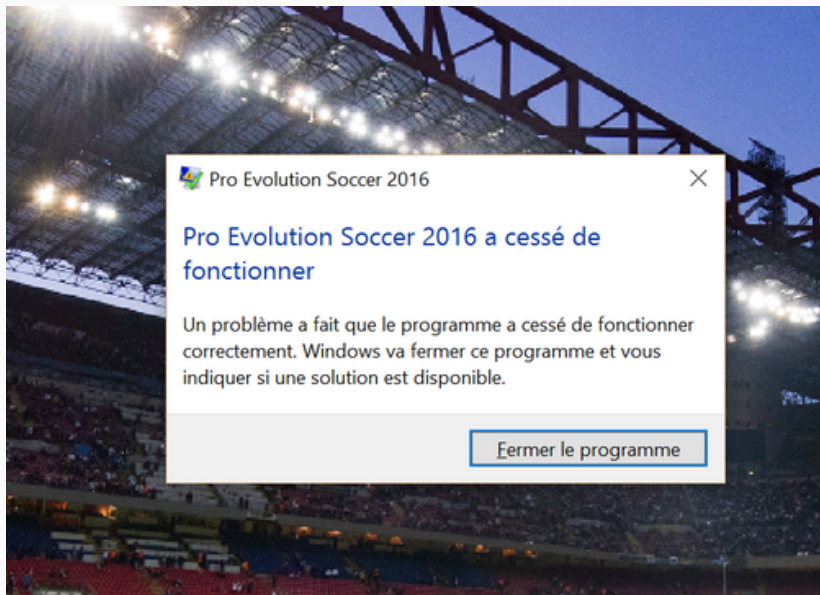
Vecteurs

taille tableau



Accès à un élément

```
vector<int> vec = {1, 2, 3, 4};  
vec[0];  
vec[4];
```



Accès vérifié à un élément

- `vec.at(i)` = `vec[i]` si `i` est un indice valide
- si `i` n'est pas valide, alors `vec.at(i)` lance une exception
- le comportement de `vec[i]` n'est pas défini si `i` n'est pas valide
- `vec.at(i)` contient une vérification d'indice, `vec[i]` n'en contient pas

Suppression d'un élément

- suppression du dernier élément : `vec.pop_back()`

Insertion et suppression à un indice arbitraire

- insertion de `val` à l'indice `i` :

```
vec.insert(vec.begin() + i, val);
```

- suppression de l'élément à l'indice `i` :

```
vec.erase(vec.begin() + i);
```

- `vec.begin()` est une *position* (ou un *itérateur*) qui pointe vers le premier élément du vecteur `vec`

- vec avant suppression :

1	2	3	4
---	---	---	---

- suppression :

```
vec.erase(vec.begin() + 2);
```

- vec après suppression :

1	2	4
---	---	---

Vecteurs et fonctions

- peuvent être retournés d'une fonction
- peuvent être passés à une fonction en paramètre
- pour éviter de copier des vecteurs grands : passer en *référence*
- ou en référence *constante* si on ne veut pas changer le vecteur

- passage par copie :

```
void f(vector<int> vec)
```

- passage par référence :

```
void f(vector<int> &vec)
```

- passage par référence constante :

```
void f(const vector<int> &vec)
```

Premières feuilles d'exercices

- Mini-feuille 1, date limite de rendu sous Moodle : 16 novembre 2018
- Feuille 1, date limite de rendu sous Moodle : 23 novembre 2018
- traduction anglaise à partir de la semaine prochaine

Questions
