

Estructures de Dades i Algorítmica

Curs 2024/25

Jerónimo Hernández González

jeronimo.hernandez@udg.edu

Presentació i Intro a l'STL

SESSIÓ 00 DE LABORATORI

(PDF creat el 24 de setembre de 2024)



© Jerónimo Hernández González [Universitat de Girona], 2024

Obra subjecta a una llicència de Creative Commons Reconeixement-CompartirIgual 4.0

Taula de continguts

1 Presentació

- Objectius
- Professorat
- Calendari

2 Entorn treball

- GNU/Linux
- Eines bàsiques

3 Introducció a l'STL

- El contenedor vector

Objectius de les pràctiques d'EDA

- Ajudar a consolidar conceptes vistos a les classes de teoria.
- Conèixer la biblioteca STL (*Standard Template Library*) de C++.
- Dominar el procés per passar de codi font a executable.
- Saber treballar en entorns heterogenis i escriure codi portable.

Els vostres programes hauran de funcionar bé en un entorn GNU/Linux.

Horaris i professorat

Grup	Horari	Professor
1	dimarts 14:00-16:00	Jerónimo Hernández
2	dimecres 12:00-14:00	Jerónimo Hernández
3	dijous 08:00-10:00	David Fíguls
4	dimarts 16:00-18:00	Jerónimo Hernández
5	dimarts 18:00-20:00	Jerónimo Hernández
6	dijous 10:00-12:00	David Fíguls
7	dimecres 14:30-16:30	Jerónimo Hernández
8	dijous 12:00-14:00	Josu Oca

Tutories

Contacta via email amb el teu professor per quedar:

- David Fíguls: david.figuls@udg.edu
- Jerónimo Hernández: jeronimo.hernandez@udg.edu
- Josu Oca: josu.oca@udg.edu

Calendari: sessions previstes

Sessió	Temàtica	Eines	Activitats	
			Presenta	Entrega
0: 23/set.	Introducció a l'STL	Entorn de treball		
1: 30/set.	Iteradors, vector i list	Joc de proves	E1	
2: 7/oct.			P1	E1
3: 14/oct.	map, set i algoritmes (STL)	Scripts	E2	
4: 21/oct.				E2
5: 4/nov.	Grafs	Doxygen	E3	E3
6: 11/nov.	Algoritmes voraçs	Gestió d'excepcions	E4	P1
7: 18/nov.				E4
8: 25/nov.	Backtracking	Paràmetres	E5	
9: 2/des.			P2	E5
10: 9/des.	Divideix i venç	Temps	E6	
11: 16/des.				E6

L'entrega de la **P2** serà la segona setmana de gener.

* 31/10: Els grups de dijous tindran una sessió addicional de repàs.

Taula de continguts

1 Presentació

- Objectius
- Professorat
- Calendari

2 Entorn treball

- GNU/Linux
- Eines bàsiques

3 Introducció a l'STL

- El contenedor vector

GNU/Linux

Sistema Operatiu als laboratoris

Tot el codi que desenvolueu l'haureu de compilar i executar a `bas.udg.edu` (servidor **GNU/Linux** de l'EPS).

(A Moodle teniu una introducció a l'entorn GNU/Linux de l'EPS)

Treballareu...

- com vulgueu, en qualsevol entorn, sempre que feu les entregues a `bas.udg.edu`.
- El vostre programa ha de compilar a `bas.udg.edu`.
“Al meu ordinador compilava!” no és excusa.
- El vostre programa ha de funcionar a `bas.udg.edu`.
“Al meu ordinador funciona bé!” no és excusa.

Eines bàsiques per...

Escriure codi

- Un **editor de text** (gedit, emacs, notepad++, etc.), o
- un **IDE** (CodeBlocks, CLion, etc.)

Compilar

- Un **compilador de C++** (p.e., el compilador g++ de GCC)
Feu-ne servir un que implementi l'estàndard C++ 14 per compatibilitat amb bas.udg.edu
- Pot estar integrat en el vostre IDE, però a bas.udg.edu heu de compilar per línia de comandes

Eines bàsiques per...

Comunicar-vos amb `bas.udg.edu`

- Un **client** `ssh` (com “Putty”) per obrir una sessió en línia de comandes en un servidor.

Recordeu que haureu de compilar i executar el vostre programa a `bas.udg.edu` mitjançant la línia de comandes o Terminal

- Un **client** `sftp` (com “FileZilla”) per pujar/baixar fitxers entre l'ordinador amb que treballem i `bas.udg.edu`.
- Si treballem amb una màquina virtual de GNU-Linux als ordinadors de l'EPS, us estalviareu tot això

(Aquests màquines virtuals treballen sobre `bas.udg.edu` directament)

Taula de continguts

- 1 Presentació
 - Objectius
 - Professorat
 - Calendari
- 2 Entorn treball
 - GNU/Linux
 - Eines bàsiques
- 3 Introducció a l'STL
 - El contenidor vector

Standard Template Library

La **Standard Template Library (STL)** és una biblioteca de C++ que ofereix:

- Contenedors genèrics (vector, deque, list, set, map, ...)
Per emmagatzemar objectes
- Iteradors (directes, inversos, aleatoris, ...)
Per recórrer els contenidors
- Algoritmes genèrics (find, sort, merge, ...)
Per tractar amb els contenidors
- Altres elements (adaptadors, objectes-funció, ...)

Standard Template Library

Abstracció de la implementació

L'**STL** ens permet abstraure'ns de la implementació de les estructures de dades (ED) i centrar-nos en triar la millor ED per a cada problema concret.

Si sabem, per cada contenidor, l'**ordre del cost** de cada operació:

- **Inserir** un element

pot no ser igual al principi, al final o en un punt qualsevol

- **Consultar** un element

pot no ser igual el primer, l'últim o un de qualsevol

- **Eliminar** un element

pot no ser igual el primer, l'últim o un de qualsevol

- o d'altres operacions

i l'**ús que volem donar a l'ED**, podrem triar el millor contenidor

Llibres de referència

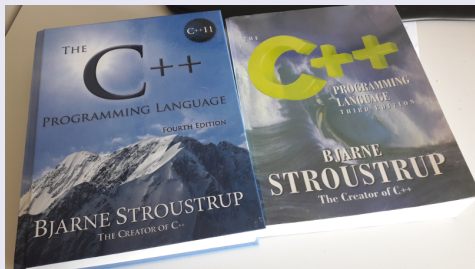
The image shows two books by Bjarne Stroustrup. The book on the left is the fourth edition, titled 'THE C++ PROGRAMMING LANGUAGE' with 'FOURTH EDITION' below it. The cover features a blue sky and a snow-capped mountain. The book on the right is the third edition, also titled 'THE C++ PROGRAMMING LANGUAGE' with 'THIRD EDITION' below it. The cover features a yellow 'C++' logo and a dark, abstract background. Both books list the author as 'BJARNE STROUSTRUP' and 'The Creator of C++'.

Bjarne Stroustrup, **The C++ Programming Language**, Addison-Wesley, *editions diverses*

Standard Template Library

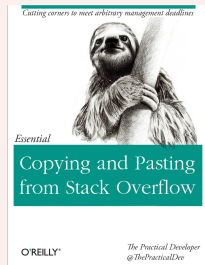
Llibres de referència

Referència bàsica



Bjarne Stroustrup, **The C++ Programming Language**, Addison-Wesley, *edicions diverses*

No recomanat



Copying and Pasting from Stack Overflow, O'Reilly

Standard Template Library

Webs de referència

cppreference.com

cppreference.com

Jordi.negre - Cerca

Pàgina Discussió Mostra Mostra la font Història Accions

CppCon 2020
It's the annual, week-long face-to-face gathering for the entire C++ community. Register now!

C++ reference

C++98, C++03, C++11, C++14, C++17, C++20

- Compiler support (11, 14, 17, 20)
- Reentrancy implementations
- Language**
 - Basic concepts
 - Keywords
 - Preprocessor
 - Expressions
 - Declaration
 - Initialization
 - Functions
 - Statements
 - Classes
 - Overloading
 - Templates
 - Exceptions
- Headers**
- Named requirements**
- Feature test macros (C++20)**
- Language support library**
 - Byte support - `std::byte`
 - Program utilities
 - Resonance comparators (C++20)
 - numeric limits - `std::numeric_limits`
 - initializer_list (C++11)
- Concepts library (C++20)**
- Diagnostics library**
- General utilities library**
 - Smart pointers and allocators
 - Date and time
 - Function objects - `std::hash` (C++11)
 - String conversions (C++14)
 - Utility functions
 - path - `std::filesystem::path` (C++17)
 - optional (C++17) - `std::optional` (C++17)
 - variant (C++17) - `std::variant` (C++17)
- Strings library**
 - basic_string
 - basic_string_view (C++17)
 - byte - multibyte - wide
- Containers library**
 - array (C++11) - `std::array`
 - map - `std::map`
 - priority_queue - `std::priority_queue`
 - Other containers:
 - sequence - `std::vector`
 - associative - `std::set`
 - unordered associative - `std::unordered_set`
- Iterators library**
- Ranges library (C++20)**
- Algorithms library**
- Numerics library**
 - Common math functions
 - Mathematical special functions (C++17)
 - Numeric algorithms
 - Pseudo-random number generation
 - Fixed-point environment (C++11)
 - complex - `std::complex`
 - valarray
- Localizations library**
- Input/output library**
 - Stream-based I/O
 - Synchronized output (C++20)
 - IO manipulators
- Filesystem library (C++17)**
- Regular expressions library (C++11)**
 - basic_regex - `std::regex`
- Atomic operations library (C++11)**
 - atomic - `std::atomic`
 - atomic_flag
- Thread support library (C++11)**

<http://en.cppreference.com>

The C++ Resources Network

cplusplus.com

Search: [C++]

register log in

Welcome to cplusplus.com

Information
Tutorials
Reference
Articles
Forum

Information	Tutorials
General information about the C++ programming language, including non-technical documents and descriptions: <ul style="list-style-type: none">Description of the C++ languageHistory of the C++ languageF.A.Q., Frequently Asked Questions	Learn the C++ language from its basics up to its most advanced features: <ul style="list-style-type: none">C++ Language: Collection of tutorials covering all the features of this versatile and powerful language, including detailed explanations of pointers, functions, classes and templates, among others...More...
Reference	Articles
Description of the most important classes, functions and objects of the Standard Language Library, with descriptive fully-functional short programs as examples: <ul style="list-style-type: none">C library: The popular C library, is also part of the C++ language library.STDLIB library: The standard C++ library for input/output operations.String library: Library defining the string class.Standard containers: Vectors, lists, maps, sets...More...	User-contributed articles, organized into different categories: <ul style="list-style-type: none">AlgorithmsStandard libraryC++11Windows APIOther... You can contribute your own article!
Forum	C++ Search
Message boards where members can exchange knowledge	Search this website:

<http://www.cplusplus.com>

vector

Trenquem el gel amb l'STL

El contenidor `vector` ve a substituir les taules de C++.
Avui veurem...

- com es crea (constructor)
i com es redimensiona
- com es consulta
- com s'hi insereixen elements
- com n'eliminem

Primer programa amb vector

Mesurant distàncies en un *track* de GPS

- Disposem d'un *track* de GPS:

```
42.24518686 1.71632082 1673.078
42.24518686 1.71632082 1673.078
42.24518962 1.71633289 1673.012
...
42.29380625 1.80525986 1555.453
42.29385629 1.80522230 1556.615
42.29388345 1.80517805 1557.111
```

on cada línia és un punt GPS (tripleta lat, lon, ele).

- Volem un programa que
 - 1 llegeixi una sèrie de punts GPS,
 - 2 els guardi en una taula, i
 - 3 calculi la longitud del *track*.

15/20

La classe PuntGPS

```
class PuntGPS {  
public:  
    PuntGPS();  
    PuntGPS(double latitud, double longitud, double elevacio);  
  
    double latitud() const; // retorna latitud  
    double longitud() const; // retorna longitud  
    double elevacio() const; // retorna elevacio  
  
    // distancia entre this i p1  
    double operator - (const PuntGPS &p1) const;  
  
private:  
    ...  
};
```

La classe PuntGPS

Incís

latitud(), longitud() i elevacio() **no** són *getters*. Diferents implementacions poden guardar l'estat de l'objecte. #abstracció

Atributs privats 1

```
class PuntGPS {  
public:  
...  
private:  
    // latitud (graus)  
    double _latitud;  
    // longitud (graus)  
    double _longitud;  
    // elevacio (metres)  
    double _elevacio;  
};
```

Atributs privats 2

```
class PuntGPS {  
public:  
...  
private:  
    // _dades[0]: latitud (graus)  
    // _dades[1]: longitud (graus)  
    // _dades[2]: elevacio (metres)  
    double _dades[3];  
};
```

La classe PuntGPS

Implementació càlcul distància

```
#include <cmath>
#include "PuntGPS.h"

const double PI = acos(-1.0);

PuntGPS::PuntGPS(){_latitud=_longitud=_elevacio=0.0;}
PuntGPS::PuntGPS(double lat, double lon, double elev) {
    _latitud=lat;
    _longitud=lon;
    _elevacio=elev;
}

double PuntGPS::latitud() {return _latitud;}
double PuntGPS::longitud() {return _longitud;}
double PuntGPS::elevacio() {return _elevacio;}

double PuntGPS::operator - (const PuntGPS &p1){
    return sqrt(pow(_latitud-p1._latitud,2)+
                pow(_longitud*cos(_latitud/180*PI) -
                    p1._longitud*cos(p1._latitud/180*PI),2))*
        111319.9;
}
```

Solució amb taules estàtiques

Taules estàtiques: la mida, en temps de compilació

taula1.cpp



```
#include <iostream>
#include "PuntGPS.h"
using namespace std;
const int MAX_PUNTS=100;

int main() {
    PuntGPS punts[MAX_PUNTS];
    double lat, lon, ele, distancia=0.0;
    int numPunts=0;
    cin >> lat >> lon >> ele;
    while(!cin.eof()) {
        punts[numPunts++] = PuntGPS(lat, lon, ele);
        cin >> lat >> lon >> ele;
    }
    if (numPunts > MAX_PUNTS) cout << "ERROR: massa punts" << endl;
    for (int i=1; i<numPunts; i++) {
        distancia += punts[i-1] - punts[i];
    }
    ...
}
```

Solució amb taules estàtiques

Taules estàtiques: la mida, en temps de compilació

taula1.cpp



```
#include <iostream>
#include "PuntGPS.h"
using namespace std;
const int MAX_PUNTS=100;

int main() {
    PuntGPS punts[MAX_PUNTS];
    double lat, lon, ele, distancia=0.0;
    int numPunts=0;
    cin >> lat >> lon >> ele;
    while(!cin.eof()) {
        punts[numPunts++] = PuntGPS(lat, lon, ele);
        cin >> lat >> lon >> ele;
    }
    if (numPunts > MAX_PUNTS) cout << "ERROR: massa punts" << endl;
    for (int i=1; i < numPunts; i++) {
        distancia += punts[i-1] - punts[i];
    }
    ...
}
```

Problemes de rang: i si hi ha més de MAX_PUNTS punts?

Si accedim amb $i \geq \text{MAX_PUNTS}$, tindrem **error de segmentació** o (pitjor) no es generarà cap error i els resultats seran **imprevisible**.

(Fins i tot pot semblar que funcioni!)

Solució amb taules dinàmiques

Taules dinàmiques: la mida, en temps d'execució

taula2.cpp



```
#include <iostream>
#include "PuntGPS.h"
using namespace std;

int main() {
    PuntGPS *punts;
    double lat, lon, ele, distancia=0.0;
    int numPunts;
    cerr << "Numero de punts? "; cin >> numPunts;
    punts=new PuntGPS[numPunts];
    for(int i=0; i<numPunts; i++){
        cin >> lat >> lon >> ele;
        punts[i] = PuntGPS(lat,lon,ele);
    }
    for (int i=1;i<numPunts;i++) {
        distancia += punts[i - 1]-punts[i];
    }
    ...
    delete [] punts;
}
```


Solució amb taules dinàmiques

Taules dinàmiques: la mida, en temps d'execució

taula2.cpp



```
#include <iostream>
#include "PuntGPS.h"
using namespace std;

int main() {
    PuntGPS *punts;
    double lat, lon, ele, distancia=0.0;
    int numPunts;
    cerr << "Numero de punts? "; cin >> numPunts;
    punts=new PuntGPS[numPunts];
    for(int i=0; i<numPunts; i++){
        cin >> lat >> lon >> ele;
        punts[i] = PuntGPS(lat,lon,ele);
    }
    for (int i=1;i<numPunts;i++) {
        distancia += punts[i - 1]-punts[i];
    }
    ...
    delete [] punts;
}
```

Més control sobre el rang

Sabem d'abans quants punts hi haurà!

És factible saber-ho sempre *a priori*??

Gestió de memòria dinàmica

Cal gestionar-la bé (new[] i delete[])

Solució amb vector

Contenedor vector

vector.cpp



```
#include <iostream>
#include "PuntGPS.h"
#include <vector>
using namespace std;

int main() {
    vector<PuntGPS> punts;
    double lat, lon, ele, distancia=0.0;
    cin >> lat >> lon >> ele;
    while (!cin.eof()){ // final fitxer cin
        punts.push_back(PuntGPS(lat,lon,ele));
        cin >> lat >> lon >> ele;
    }
    for (int i=1;i<punts.size();i++)
        distancia += punts[i-1]-punts[i];
    cout << punts.size() << "punts, distancia=";
    cout << distancia << "metres" << endl;
    cout << "distancia linia recta entre primer i ultim:";
    cout << punts.front()-punts.back() << "metres" << endl;
}
```

Estructures de Dades i Algorítmica

Curs 2024/25

Jerónimo Hernández González

jeronimo.hernandez@udg.edu

Presentació i Intro a l'STL

SESSIÓ 00 DE LABORATORI

(PDF creat el 24 de setembre de 2024)



© Jerónimo Hernández González [Universitat de Girona], 2024

Obra subjecta a una llicència de Creative Commons Reconeixement-CompartirIgual 4.0