Universitat de Girona
Departament d'Informàtica, Matemàtica Aplicada i Estadística

Estructures de Dades i Algorítmica

Curs 2024/25

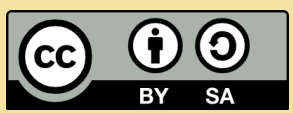
Presentació i Intro a l'STL

Sessió 00 de Laboratori

Jerónimo Hernández González

jeronimo.hernandez@udg.edu

(PDF creat el 24 de setembre de 2024)



Taula de continguts

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 voraços	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 vençe	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 adiccional de repàs.

Taula de continguts

GNU/Linux

Sistema Operatiu als laboratoris

Tot el codi que desenvolupeu 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 treballeu i `bas.udg.edu`.

- Si treballeu 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

Standard Template Library

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

- Contenidors 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 abstreure'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

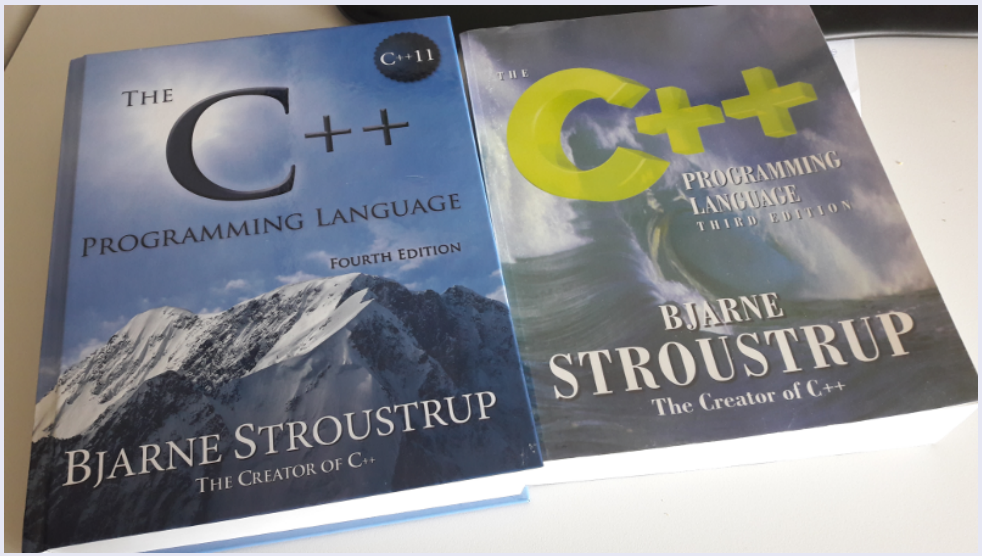
  pot no ser igual el primer, l'últim o un de qualsevol
- o d'altres operacions

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

Standard Template Library

Llibres de referència

Referència bàsica

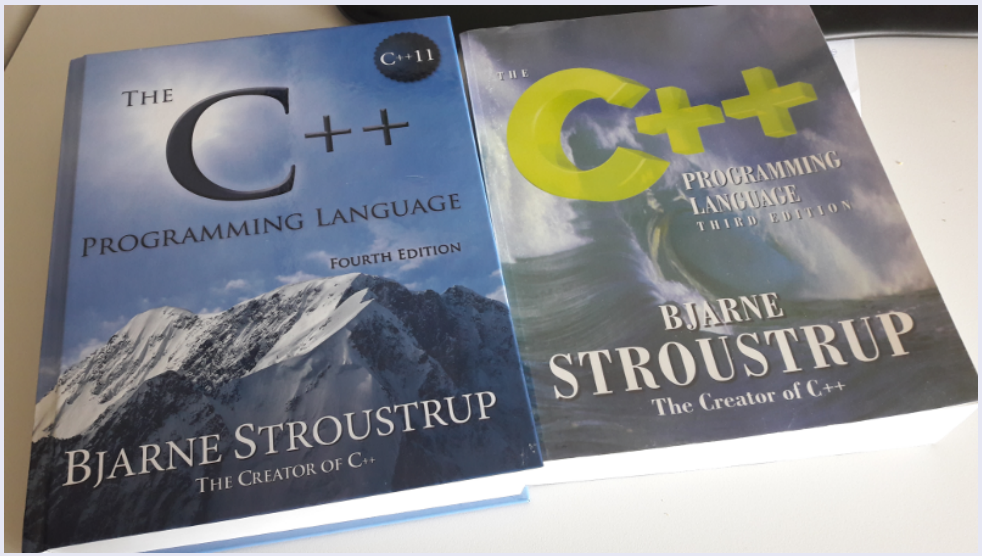


Bjarne Stroustrup, **The C++ Programming Language**, Addison-Wesley, *edicions diverses*

Standard Template Library

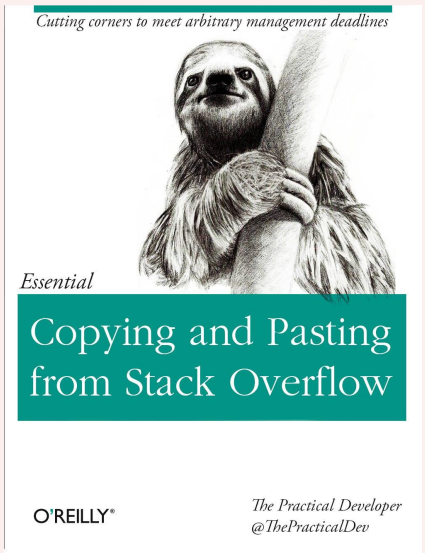
Llibres de referència

Referència bàsica



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

No recomanat

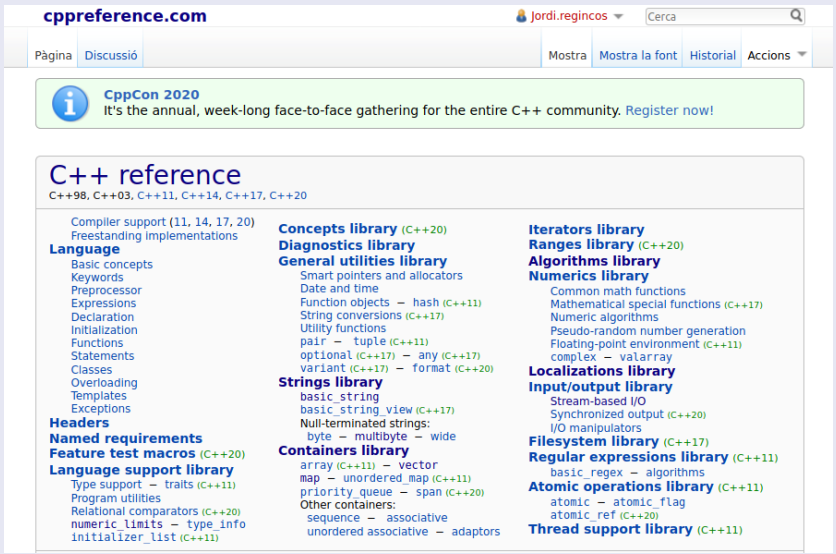


Copying and Pasting from Stack Overflow, O'Reilly

Standard Template Library

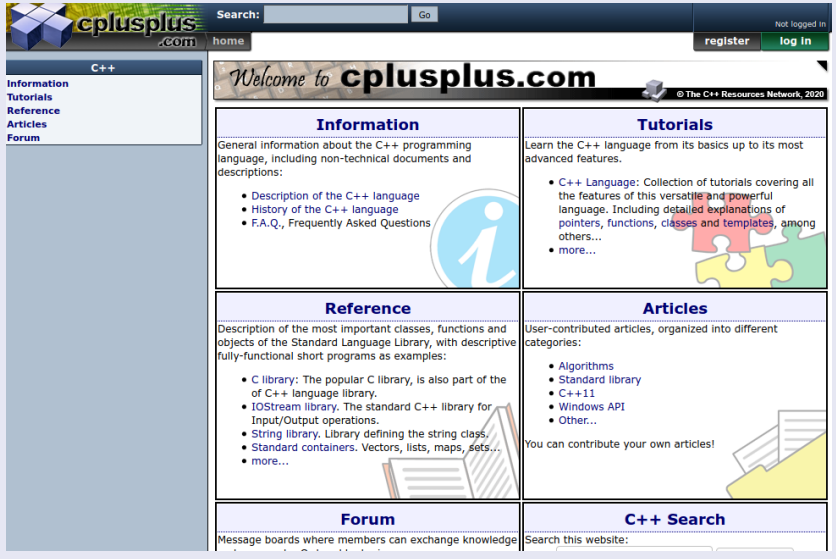
Webs de referència

cppreference.com



http://en.cppreference.com

The C++ Resources Network



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*.

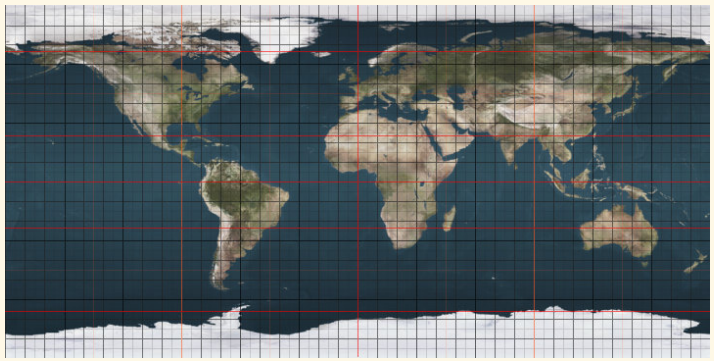
(Enllaç a l'origen de les dades)

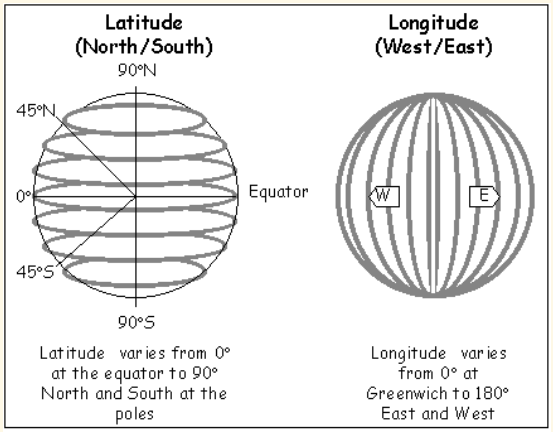
Primer programa amb `vector`

Mesurant distàncies en un *track* de GPS

Per al **càlcul de la distància** entre dos punts GPS:

- Suposem que la Terra és una esfera perfecta
- Suposem que 2 punts GPU son suficientment propers per fer servir la distància euclídea #varietat(math) (com si fos un pla, #terraplanisme?)
- Un grau de **latitud** correspon a 111.319,9 metres
- Un grau de **longitud** correspon a 111.319,9 metres **a l'equador** i decreix proporcionalment al cosinus de la latitud quan ens desplacem cap als pols.





(Enllaç a l'origen de les dades)

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>=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

Contenidor 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;
}
```

Universitat de Girona
Departament d'Informàtica, Matemàtica Aplicada i Estadística

Estructures de Dades i Algorítmica

Curs 2024/25

Presentació i Intro a l'STL

Sessió 00 de Laboratori

Jerónimo Hernández González

jeronimo.hernandez@udg.edu

(PDF creat el 24 de setembre de 2024)