Alfredo Vellido : www.cs.upc.edu/~avellido

Fonaments d'Informàtica: 2ºP

Semanas 3-…: Tipos estructurados (vectores)

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Qué son los vectores y por qué los necesitamos

- Las tablas unidimensionales (arrays) **nos obligan a reservar un número (limitado) de posiciones en memoria; es decir, tienen una longitud máxima** inamovible por definición
- Hemos visto que los **strings** son tablas “especiales” de caracteres que **no tienen longitud predefinida**.
- Los **vectores** se benefician de “lo mejor de ambos mundos”: son como tablas con elementos accesibles por índice, **pero no tienen longitud predefinida.** Eso hace que podamos definir vectores de una longitud casi arbitraria.

Qué son los vectores y cómo los declaramos

- La **clase vector** forma parte de la biblioteca estándar de C++, y en particular de la **STL: la Standard Template Library**

```
#include<vector>

…

vector<int> v; // … ó …

vector<int> v(10); // … ó …

vector<string> vst(3,"monja"); // … ó …

vector<string> vstcop(vst);

vector<string> vstcop2=vst; // lo mismo que anterior
```

¿Cómo acceder a vectores?

- Accedemos a casillas de **vectores** de la misma forma que accederíamos a casillas de tablas, mediante indexación.

```
vector<int> v(10);

v[0]=5;

int a = v[0]; cout << a;
```

El final de un vector y qué hacer con él

- No sabemos necesariamente a priori cuál es la longitud de un **vector** en un momento dado de la ejecución de un programa … pero tenemos herramientas para saberlo …

```
vector<int> v(10);

tam = v.size();

for (int i=0; i<tam; i++) …
```

El final de un vector y qué hacer con él

- Y tenemos un **método** para añadir elementos al final de un **vector** …

```
vector<int> v;

v.push_back(7);

cout << v[0];
```

El final de un vector y qué hacer con él

- Así como un **método** para quitar elementos al final de un **vector** …

```
vector<char> v(4,'z');

v.pop_back();

cout << v.size() << endl; // qué se ve?
```

El final de un vector y qué hacer con él

- Y por supuesto un **método** para cargárnoslo todo …

```
vector<char> v(4,'z');

v.clear();

cout << v.size() << ' ' << v.empty() << endl;
```

El final de un vector y qué hacer con él

- Quitar y quitar elementos al final de un **vector** …

```
vector<char> v(4,'z');
for (int i=0; i<v.size(); i++) v.pop_back();
cout << v.size() << endl; // qué se ve?
```

El final de un vector y qué hacer con él

- ¿Qué veremos en pantalla?

```
vector<int> v(10);
int tam = v.size(); cout << tam << endl;
for (int i=0; i<tam; i++) v.push_back(i);

for (int i=0; i<tam; i++) cout << v[i]; cout << endl;
int tam2 = v.size(); cout << tam2 << endl;
for (int i=0; i<tam2; i++) cout << v[i]; cout << endl;
```

Jugando con grandes vectores desde ficheros

- Almacenar en un **fichero** la secuencia invertida de **ecoli** usando vectores.

```
#include <iostream>
#include <vector>
#include <fstream>
using namespace std;

int main() {
   vector<char> v_ecoli;
   ifstream fin("ecoli.txt");
   char nuc;
   fin >> nuc;
   while (!fin.eof()) {
      v_ecoli.push_back(nuc);
      fin >> nuc;}

   ofstream fout("ecoli_reves.txt");
   int tam = v_ecoli.size();
   for (int i = 0; i < tam; i++) {
      fout << v_ecoli[tam-1-i];
      if (i%1024 == 1023) fout << endl;
   }
}
```

Vectores como parámetros de subprogramas

```
<tipo_retorno> nom_subp([const] vector<tipo>& v1 …)
```

Vectores como parámetros de subprogramas

Escribir un subprograma que concatene y retorne concatenados dos vectores pasados como parámetros.

```
vector<int> concatena(const vector<int>& v1, const vector<int>&  v2)
{
   vector<int> resultado = v1;
   for (int i = 0; i < v2.size(); i++) {
      resultado.push_back(v2[i]);
   }
   return resultado;
}
```

Vectores como parámetros de subprogramas

Escribid una función ***apariciones*** que reciba un vector de strings ***v*** y una palabra ***p*** y retorne cuántas veces aparece ***p*** en ***v***.

Vectores como parámetros de subprogramas

Escribid un **subprograma** que reciba un vector de reales y me retorne los valores mínimo y máximo que están almacenados en el mismo (asumimos que al menos hay un valor almacenado en el vector).

Ahora escribid un programa completo que me lea una secuencia de valores reales de un fichero, cargándolos en un vector, y que me muestre por pantalla sus valores máximo y mínimo usando el subprograma diseñado previamente.