Vectores en C++

Laboratorio
Algoritmos y Estructura de Datos I

DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

1er Cuatrimestre 2019

Estructura de datos

En computación estructura de datos refire a
la organizados de los datos en memoria.

- La organización de los datos en memoria puede facilitar ciertas operaciones sobre ellos:
  a) Agregar elementos
  b) Buscar elementos
  c) Encontrar el valor mínimo
  d) ∞

Vectores en C++

La clase `vector` de C++ es una implementación de las secuencias del lenguaje de especificación, $seq\langle T \rangle$.

- Un vector en C++ es una **estructura de datos** con las siguientes propiedades:
  1. Todos los elementos son del mismo tipo.
  2. Cada elemento está identificado por un índice.
  3. El tamaño puede variar, agregando o eliminando elementos.

Declarar vectores

Para usar vectores hay que incluir la biblioteca,

```
1 #include <vector>
```

Para declaración vectores,

```
1 vector<int> secuenciaDeEnteros;
2 vector<float> secuenciaDeReales;
3 vector<vector<int>> matrizDeEnteros;
```

Inicialmente, el vector no contiene ningún elemento.

Declarar vectores

```cpp
#include <vector>
using namespace std;

int main(){
  vector<int> vector_A; // vector de enteros vacio
  vector<int> vector_B(4); // <0, 0, 0, 0>
  vector<float> vector_C(4); // <0.0, 0.0, 0.0, 0.0>
  vector<bool> vector_D(4,true); // <true, true, true, true>
  return 0;
}
```

Longitud de vectores

La función `size()` implementa la longitud de secuencias, $|.|$, del lenguaje de especificación.

```
1  #include <vector>
2  using namespace std;
3
4  int main(){
5    vector<int> vector_A; // vector de enteros vacio
6    vector<int> vector_B(4); // <0, 0, 0, 0>
7    int longitud_A = vector_A.size(); // longitud_A == 0
8    int longitud_B = vector_B.size(); // longitud_B == 4
9    return 0;
10 }
```

Modificar longitud (agregar)

La función `push_back()` permite agregar elementos al final de un vector ya declarado

```
1  #include <vector>
2  using namespace std;
3
4  int main() {
5    vector<int> cuenta; // crea el vector vacio
6    cuenta.push_back(1); // <1>
7    cuenta.push_back(2); // <1,2>
8    cuenta.push_back(3); // <1,2,3>
9    cuenta.push_back(4); // <1,2,3,4>
10   return 0;
11 }
```

Modificar longitud (eliminar)

La función pop_back() permite eliminar elementos al final de un vector ya declarado

```
1  #include <vector>
2  using namespace std;
3
4  int main() {
5    vector<bool> cuenta; // <>
6    for(int i = 1; i <= 4; i = i + 1){
7      cuenta.push_back(i);
8    } // <1, 2, 3, 4>
9    cuenta.pop_back(); // <1, 2, 3>
10   cuenta.pop_back(); // <1, 2>
11   return 0;
12 }
```

Acceder a los elementos

La funcón `v[i]` lee el `i`-esimo elemento del vector `v`

```
1  #include <vector>
2  using namespace std;
3
4  int main() {
5    vector<int> v; // <>
6    v.push_back(1); // <1>
7    v.push_back(2); // <1, 2>
8    int valor0 = v[0]; // valor0 == 1
9    int valor1 = v[1]; // valor1 == 2
10   return 0;
11 }
```

Modificar un elemento

Para modificar el valor de un elemento ya declarado se usa la misma sintaxis, pero el elementos `v[i]` se escribe de lado izquierdo de la asignación

```
1  #include <vector>
2  using namespace std;
3
4  int main() {
5    vector<int> v; // <>
6    v.push_back(1); // <1>
7    v.push_back(2); // <1, 2>
8    v[0] = 10; // <10, 2>
9    v[1] = 20; // <10, 20>
10   return 0;
11 }
```

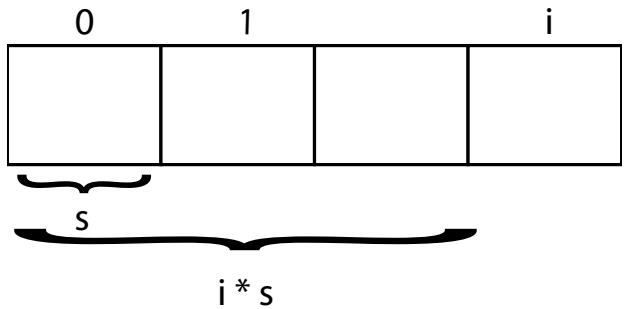
Elementos como variables

Los elementos de un vector pueden ser utilizados como si fueran una variable:

```
1   #include <vector>
2  using namespace std;
3
4  int main() {
5    vector<int> v(4); // <0,0,0,0>
6    v[0] = 7; // <7,0,0,0>
7    v[1] = v[0] * 2; // <7,14,0,0>
8    v[2] = v[2] + 1; // <7,14,1,0>
9    v[3] = -60; // <7,14,1,-60>
10   return 0;
11 }
```

Qué hace C++ internamente

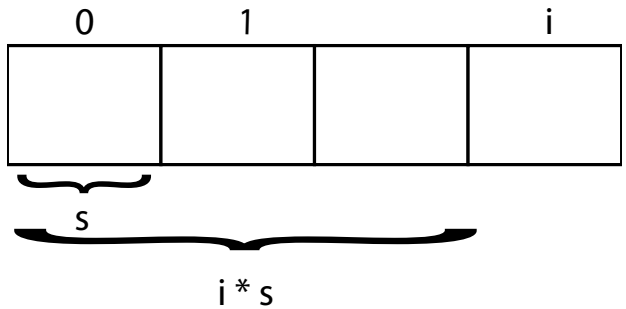
Los elementos se guardan en memoria en forma consecutiva.



- Si un elemento de tipo T ocupa s bytes, entonces el elemento en la posición i se encuentra en la posición $i \times s$ después del inicio:

Qué hace C++ internamente

Los elementos se guardan en memoria en forma consecutiva.



- Si un elemento de tipo T ocupa s bytes, entonces el elemento en la posición i se encuentra en la posición $i \times s$ después del inicio:
- Luego, obtener un elemento cualquiera tiene un tiempo de ejecución constante, independientemente del tamaño del vector.

Qué hace C++ internamente

Si los elementos se guardan forma consecutiva,
¿Qué pasa si agrandamos el vector mediante un `push_back()`?

Qué hace C++ internamente

Si los elementos se guardan forma consecutiva,
¿Qué pasa si agrandamos el vector mediante un `push_back()`?

- Es posible que (internamente) el vector deba ser copiado a otro lugar de la memoria, con costo de ejecución proporcional al tamaño del vector.

Posiciones inválidas

¿Qué ocurre si accedemos a una posición no definida?

```
1  #include <vector>
2  using namespace std;
3
4  int main() {
5    vector<int> cuenta; // <>
6    cuenta.push_back(1); // <1>
7    cuenta[2000] = 10; // ?
8    int valor = cuenta[2000]; // ?
9    return 0;
10 }
```

Posiciones inválidas

¿Qué ocurre si accedemos a una posición no definida?

```
1  #include <vector>
2  using namespace std;
3
4  int main() {
5    vector<int> cuenta; // <>
6    cuenta.push_back(1); // <1>
7    cuenta[2000] = 10; // ?
8    int valor = cuenta[2000]; // ?
9    return 0;
10 }
```

- C++ no define qué ocurre cuando accedemos a posiciones fuera de rango, el comportamiento está **indefinido**.

Posiciones inválidas

¿Qué ocurre si eliminamos elementos de un vector vacío?

```
1  #include <vector>
2  using namespace std;
3
4  int main() {
5    vector<int> v; // <>
6    v.push_back(1); // <1>
7    v.pop_back(); // <>
8    v.pop_back(); // ?
9    return 0;
10 }
```

Posiciones inválidas

¿Qué ocurre si eliminamos elementos de un vector vacío?

```cpp
#include <vector>
using namespace std;

int main() {
  vector<int> v; // <>
  v.push_back(1); // <1>
  v.pop_back(); // <>
  v.pop_back(); // ?
  return 0;
}
```

- C++ no define qué ocurre cuando queremos eliminar elementos de vectores vacios, el comportamiento está **indefinido**.

Posiciones inválidas

La **precondición** de leer o escribir una posición (`[...]`) es que la posición haya sido previemente definida

- Algunos posibles resultados al leer o escribir una posición fuera de rango en C++:
  - Puede ser correcto
  - Puede dar un error (exception) durante la ejecución
  - Puede generar un `segmentation fault` y terminar la ejecución del programa.
  - Puede colgarse

Copiar vectores

¿Cómo copiamos un vector a en otro vector b?

Opción 1: Copiar elemento a elemento.

```
1 vector<double> b;
2 for(int i=0; i<a.size(); i=i+1) {
3   b.push_back(a[i]);
4 }
```

Copiar vectores

¿Cómo copiamos un vector a en otro vector b?

Opción 1: Copiar elemento a elemento.

```
1 vector<double> b;
2 for(int i=0; i<a.size(); i=i+1) {
3   b.push_back(a[i]);
4 }
```

Opción 2: Usar el operador de asignación =.

```
1 vector<double> b;
2 b = a;
```

- Ambas opciones tienen el mismo resultado.

Funciones con vectores

¿Cómo declaramos una función que retorne un vector?

```
1 vector<int> funcionQueRetornaVector(vector<int> v){
2     vector<int> res;
3     ...
4     return res;
5 }
```

Funciones con vectores

¿Cómo declaramos una función que retorne un vector?

```
1 vector<int> funcionQueRetornaVector(vector<int> v){
2     vector<int> res;
3     ...
4     return res;
5 }
```

- Internamente: el vector `v` pasado como parámetro se recibe por copia.
- Dentro del código: la función devuelve por el vector `res` declarado dentro de su scope.

Retorno de vectores (por copia)

```
1  #include <vector>
2  using namespace std;
3
4  vector<int> crearVector(int n) {
5    vector<int> res;
6    for (int i=1; i<=n;i=i+1) {
7      v.push_back(i);
8    }
9    return res;
10 }
11
12 int main() {
13   vector<int> cuenta = crearVector(5); // <1,2,3,4,5>
14   return 0;
15 }
```

Vectores como parámetros (por copia)

Atención: los vectores, como cualquier parámetro, siempre se reciben por copia a no ser que pidamos lo contrario.

```
1  #include <vector>
2  using namespace std;
3
4  void modificarVector(vector<int> a) {
5    a[0]=35;
6  }
7
8  int main() {
9    vector<int> v(3,10); // <10,10,10>
10   modificarVector(v); // ?
11   return 0;
12 }
```

Vectores como parámetros (por copia)

Atención: los vectores, como cualquier parámetro, siempre se reciben por copia a no ser que pidamos lo contrario.

```
1  #include <vector>
2  using namespace std;
3
4  void modificarVector(vector<int> a) {
5    a[0]=35;
6  }
7
8  int main() {
9    vector<int> v(3,10); // <10,10,10>
10   modificarVector(v); // ?
11   return 0;
12 }
```

- La función `cambiarVector()` no afecta el estado del vector en el `main()`

Vectores como parámetros (por referencia)

Para modificar el vector hay que pasarlo por referencia &.

```
1  #include <vector>
2  using namespace std;
3
4  void modificarVector(vector<int>& a) {
5    a[0]=35;
6  }
7
8  int main() {
9    vector<int> v(3,10); // <10,10,10>
10   modificarVector(v); // <35,10,10>
11   return 0;
12 }
```

Sumar los elementos de una secuencia

proc *suma*(in $s : seq\langle\mathbb{Z}\rangle$, out $res : \mathbb{Z}$){
  Post $\{res = \sum_{i=1}^{|s|} n[i]\}$
}

Solución usando `while`:

Sumar los elementos de una secuencia

proc *suma*(in $s : seq\langle\mathbb{Z}\rangle$, out $res : \mathbb{Z}$){

  Post $\{res = \sum_{i=1}^{|s|} n[i]\}$

}

Solución usando `while`:

```
1  int suma(vector<int> v) {
2    int res = 0;
3    int i =0;
4    while( i < v.size() ) {
5      res = res + v[i];
6      i = i + 1;
7    }
8    return res;
9  }
```

Resumen: Vectores en C++

`vector<int> v;`	Declara un vector sin elementos
`vector<int> v(n);`	Declara un vector con `n` elementos
`vector<int> v(n,x);`	Declara un vector con `n` elementos con el valor `x`
`v.size();`	Informa la longitud del vector
`v.push_back(x);`	Almacena el valor `x` al final del vector
`v.pop_back();`	Elimina la última posición del vector
`int a = v[i];`	Lee la posición `i` del vector `v` (y la guarda en `a`)
`v[i] = x;`	Reemplaza la posición `i` del vector con el valor `x`
`v1 = v2 ;`	Vacía `v1` y copia en `v1` todos los elementos de `v2` (`v1` y `v2` deben tener el **mismo tipo**).