# Laboratorio de Programación

## Parte 1: Vectores en C++

Algoritmos y Estructuras de Datos I

Departamento de Computación, FCEyN, Universidad de Buenos Aires.

# Estructura de datos

En computación estructura de datos refiere 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) $\infty$

# 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. Es posible agregar o eliminar elementos, que hace variar el tamaño del vector.

# 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

```
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    vector<float> vector_C(4); // <0.0, 0.0, 0.0, 0.0>
8    vector<bool> vector_D(4,true); // <true, true, true, true>
9    return 0;
10 }
```

# 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<int> 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 función `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 }
```