CS1112: Programación 2

Unidad 4: Vectores

Sesión de Teoría - 7

Profesor:
José Antonio Fiestas Iquira jfiestas@utec.edu.pe

Material elaborado por:
Maria Hilda Bermejo, José Fiestas, Rubén Rivas, Henry Gallegos

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Índice:

- Unidad 4: Vectores
  - Definición, construcción y operadores básicos

4.1

Unidad 4: Vectores

- Definición, construcción y operadores básicos

Logro de la sesión:

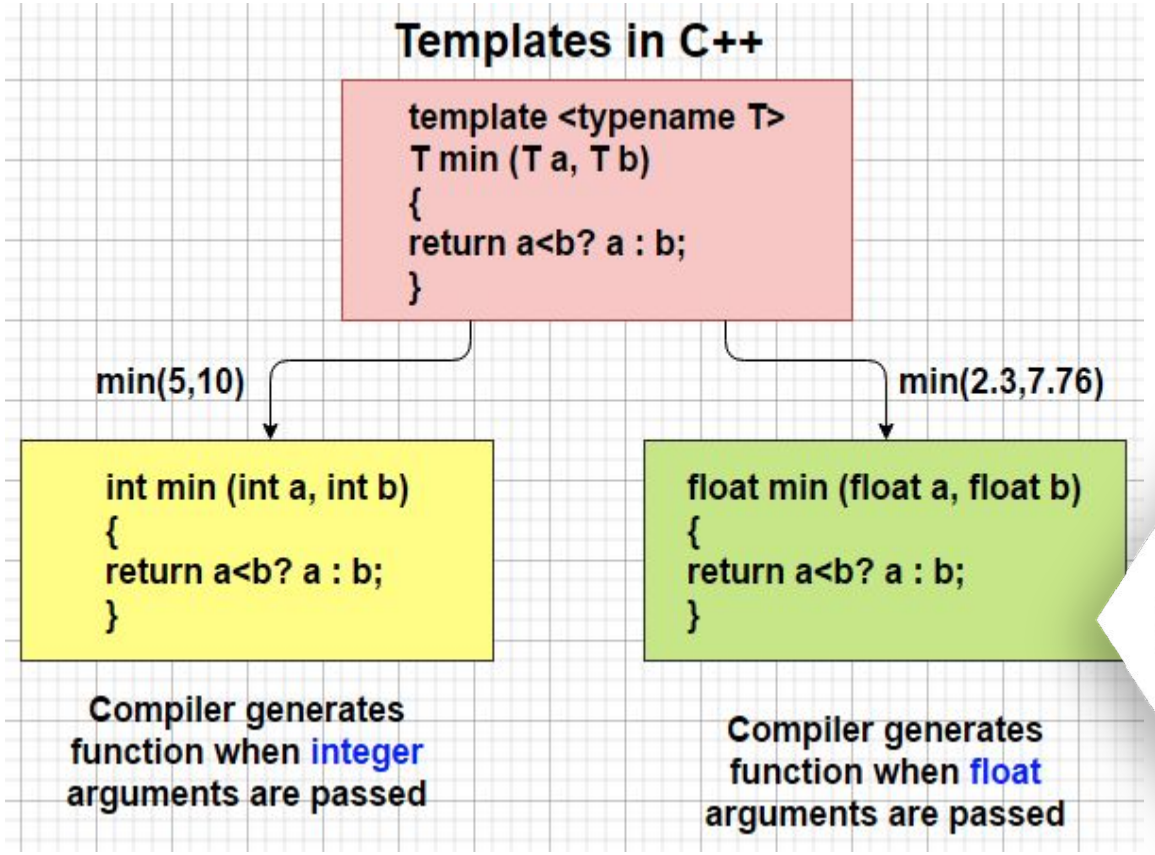
Al finalizar la sesión, los alumnos podrán:

- Usar vectores en sus variaciones para optimizar el desempeño de un código en C++.

Antecedentes ..1

A principios de los años 90 se implementa en C++ los template, estructura para el manejo de datos en forma genérica.

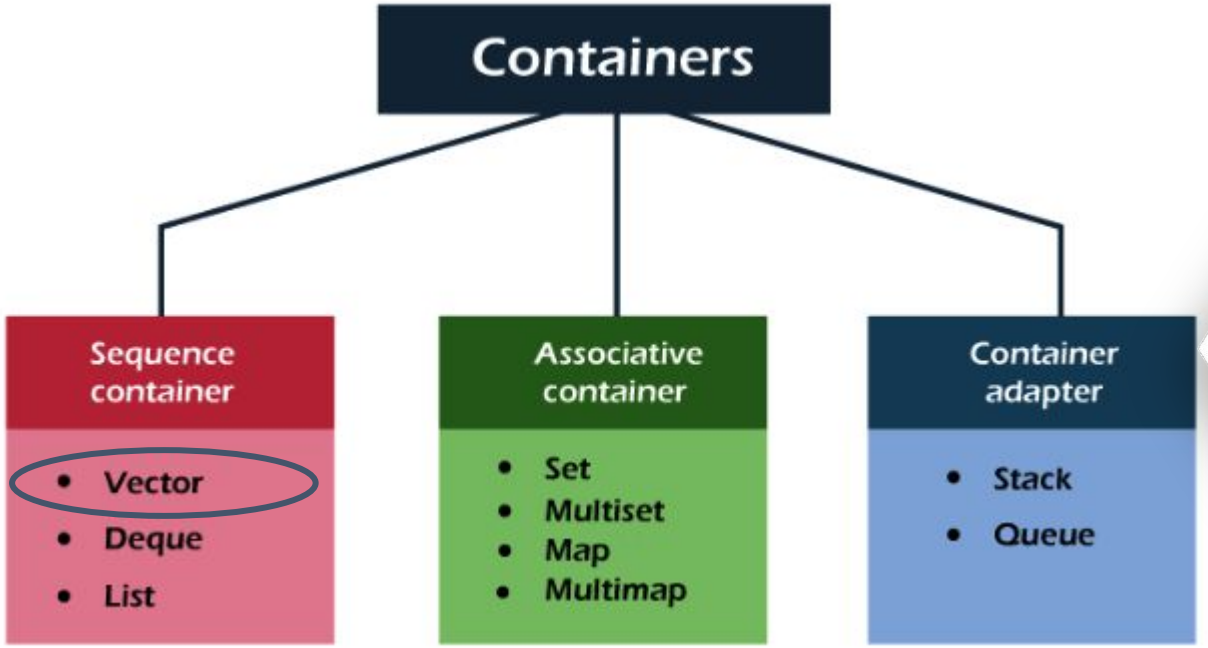
template mejoró la abstracción y permite desarrollar algoritmos seguros que no estén asociados a ningún tipo específico de datos.



Antecedentes ..2

Alex Stephanov, implementa una librería genérica en C++, denominada STL (Standard Template Library).

STL incluye diversos **tipos de clases**, entre ellos los **vectores**, que es un contenedor alternativo al **built-in array** originario de C, con estructura genérica, manejo seguro y eficiente de memoria.



UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

¿Que es un vector? características

- Es un contenedor del **tipo de clase vector** y que puede almacenar cualquier tipo de dato.
- Es una estructura secuencial indexada, de almacenamiento contiguo
- Controla automática y eficientemente el redimensionamiento y gestión de la memoria.
- Cuenta con **métodos y operadores** que permiten las acciones crear, leer, borrar, actualizar y manipular el contenedor.

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

¿Cómo se construye un contenedor vector?

Se construyen a partir de `std::vector` + **tipo de dato**:

```
std::vector<tipo de dato> vec;
std::vector<int> vec1; // Vector de enteros
std::vector<string> vec2; // Vector de strings
```

`std::vector` es un **tipo de clase** definido dentro de la librería estándar de C++ (STL) en el archivo header vector (`#include <vector>`)

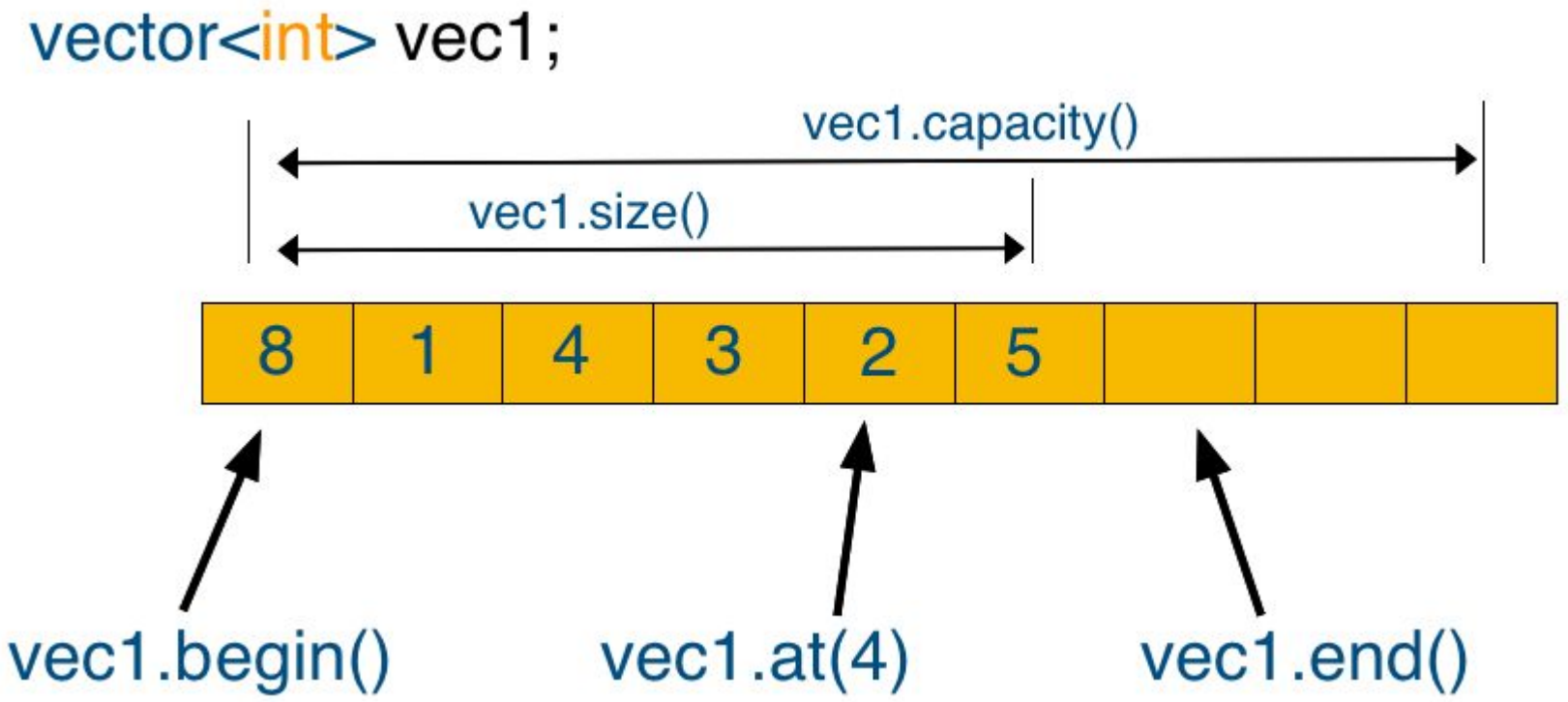
NOTA: Un **tipo de clase** es un nivel adicional de abstracción de datos, representado por **template** y que define una familia de **clases**.

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Arrays vs vectores (ejemplo00-01)

Definir (crear) un vector:

```
vector<int> vec1;  //es un tipo de clase genérica
int arreglo[100]={}; //es una coleccion de elementos del mismo tipo
```



Operaciones básicas

4 Operaciones básicas conocidas como CRUD

(Create, Read, Update & Delete)

(Creación, Lectura, Actualizar y Eliminar)

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Operaciones básicas CRUD - Create - Crear

- Los vectores a diferencia de los arreglos, inicializan automáticamente los arreglos con el valor por default del tipo de dato (cero en caso de números).

```
// Definir un vector de tamaño cero, sin valores
vector<int> v1;
// Definir un vector con de valores predefinidos.
vector<int> v2 = {1, 2, 3, 4};
// Definir un vector de tamaño predeterminado o tamaño fijo
int n = 14;
vector<int> v3(n, 5);  // Crea un vector (v3) de tamaño 14 y
                       // valores inicializados en 5
vector<int> v4(12);
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

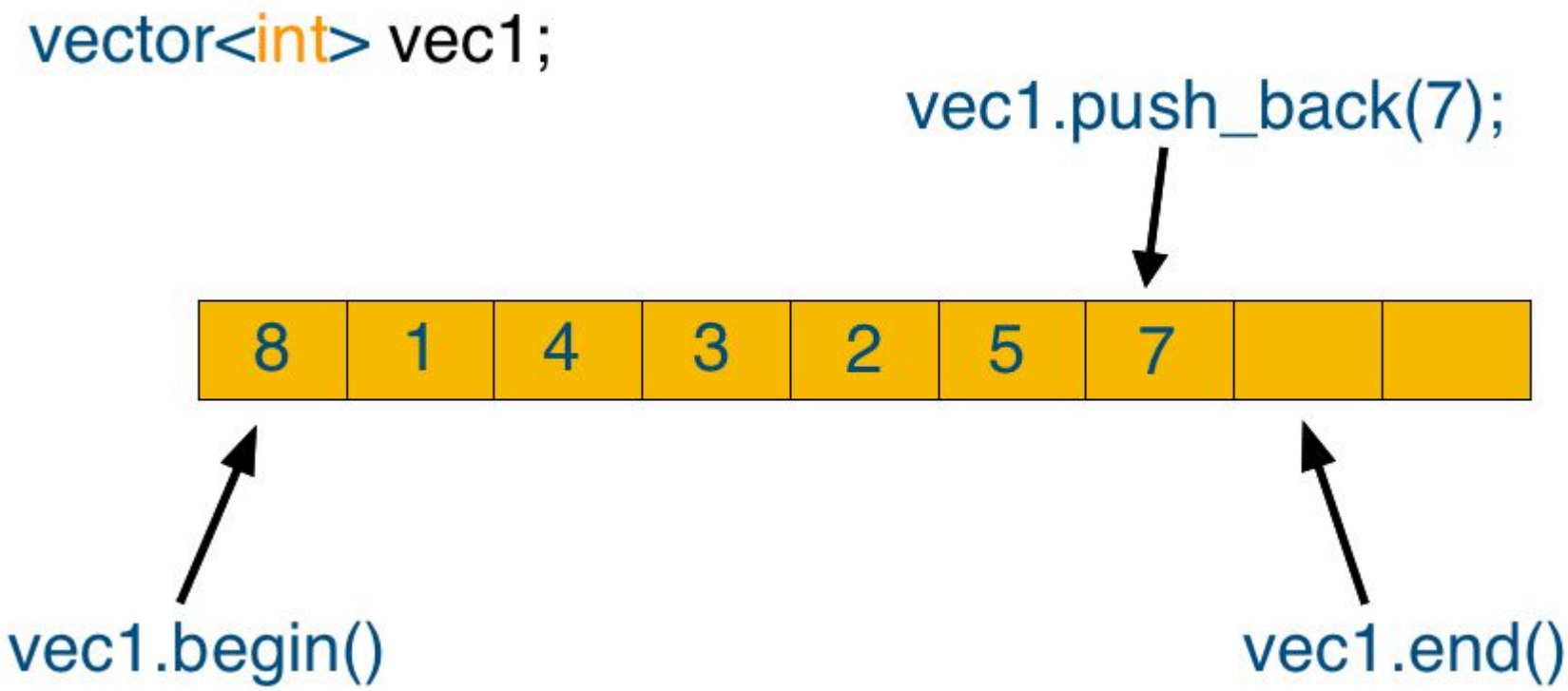
Operaciones básicas CRUD - Create - Agregar

- El vector permite agregar en cualquier posición (al final o entre valores) pero no al inicio directamente.
- Al momento de agregar se incrementa el tamaño y se redimensiona automáticamente, si es necesario.

```
vector<int> v1;
// Agregar al final
v1.push_back(10);
v1.emplace_back(10);          // desde C++11
// Agregar en cualquier posición, ejemplo en la segunda posición
v1.insert(begin(v)+1, 20);
v1.emplace(begin(v)+1, 20);  // desde C++11
// Redimensionar y Reservar espacio para futuros ítems en el vector
v1.resize(10, 4); // cambia el tamaño de 2 a 10 e inicializa
                  // valores adicionales a 4
```

Agregar elementos a un vector (ejemplo03)

```
vec1.push_back();   //agrega un elemento al final del vector
```



Operaciones básicas CRUD - Read - Subíndices

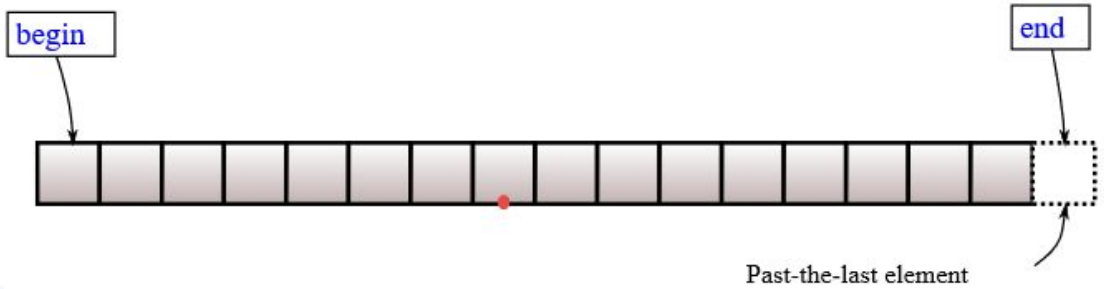
- El vector permite la lectura de posiciones aleatorias utilizando los subíndices entre braquets como los arreglos de C o utilizando el método at(position).
- La librería estándar (STL) permite utilizar los iteradores para acceder a sus valores con ayuda del for de rangos:

```
std::vector<int> v1(10);
// Inicializar con secuencia numérica desde el 1 (#include <numeric>)
std::iota(begin(v1), end(v1), 1);  // 1 es el valor inicial
// Acceso por medio de subíndices (forma tradicional)
std::cout << “El valor en la segunda posición es:” << v1[1] << endl;
// Utilizando metodo at (controla los límites)
std::cout << “El valor en la segunda posición es:” << v1.at(1) << endl;
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Operaciones básicas CRUD - Read - iteradores

- La librería estándar (STL) permite utilizar los iteradores para acceder a sus valores con ayuda del for de rangos:




```
std::vector<int> v1 = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
auto iter = begin(v1); // tienen un comportamiento similar a un puntero
// Se accede con el operador de-referencia igual que punteros
std::cout << "primera es:" << *iter << endl;                    // 10
// La aritmética de punteros funciona igual
std::cout << "tercera posición es:" << *(iter + 2) << endl;     // 30
iter++; // El operador ++ y -- funcionan en iteradores
std::cout << "segunda posición es:" << *iter << endl;           // 20
// C++ cuenta con funciones para determinar la distancia o ubicación
// respecto a otro iterador
auto dist = distance(iter, end(v1));
std::cout << dist;                                             // 9
```

Operaciones básicas CRUD - Read - Recorrido

```
std::vector<int> v = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
// Recorrido tradicional
for (int i = 0; i < v.size(); ++i)
  std::cout << v[i] << " ";
std::cout << std::endl;
// Recorrido por iteradores - antes de C++11
for (auto iter = begin(v); iter != end(v); ++iter)
  std::cout << *iter << " ";
std::cout << std::endl;
// Recorrido por iteradores ESCRITURA - usando for de rangos
for (auto & i :v)
  i += 2;
// Recorrido por iteradores SÓLO LECTURA - usando for de rangos
for (const auto & i :v)
  std::cout << i << " ";
std::cout << std::endl;
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Operaciones básicas CRUD - Update

```
std::vector<int> v = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
// Utilizando subíndices y at
for (int i = 0; i < v.size(); ++i)
  v[i] += 1;                    // {11, 21, 31, 41, 51, 61, 71, 81, 91, 101}
for (int i = 0; i < v.size(); ++i)
  v.at(i) += 1;                 // {12, 22, 32, 42, 52, 62, 72, 82, 92, 102}
// Utilizando iteradores
for (auto iter = begin(v); iter != end(v); ++iter)
  *iter += 1;                   // {13, 23, 33, 43, 53, 63, 73, 83, 93, 103}
// Utilizando loop por rangos
for (auto & i :v)
  i += 2;                       // {15, 25, 35, 45, 55, 65, 75, 85, 95, 105}
for (const auto & i :v)
  std::cout << i << " ";        // 15 25 35 45 55 65 75 85 95 105
std::cout << std::endl;
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Operaciones básicas CRUD - Delete

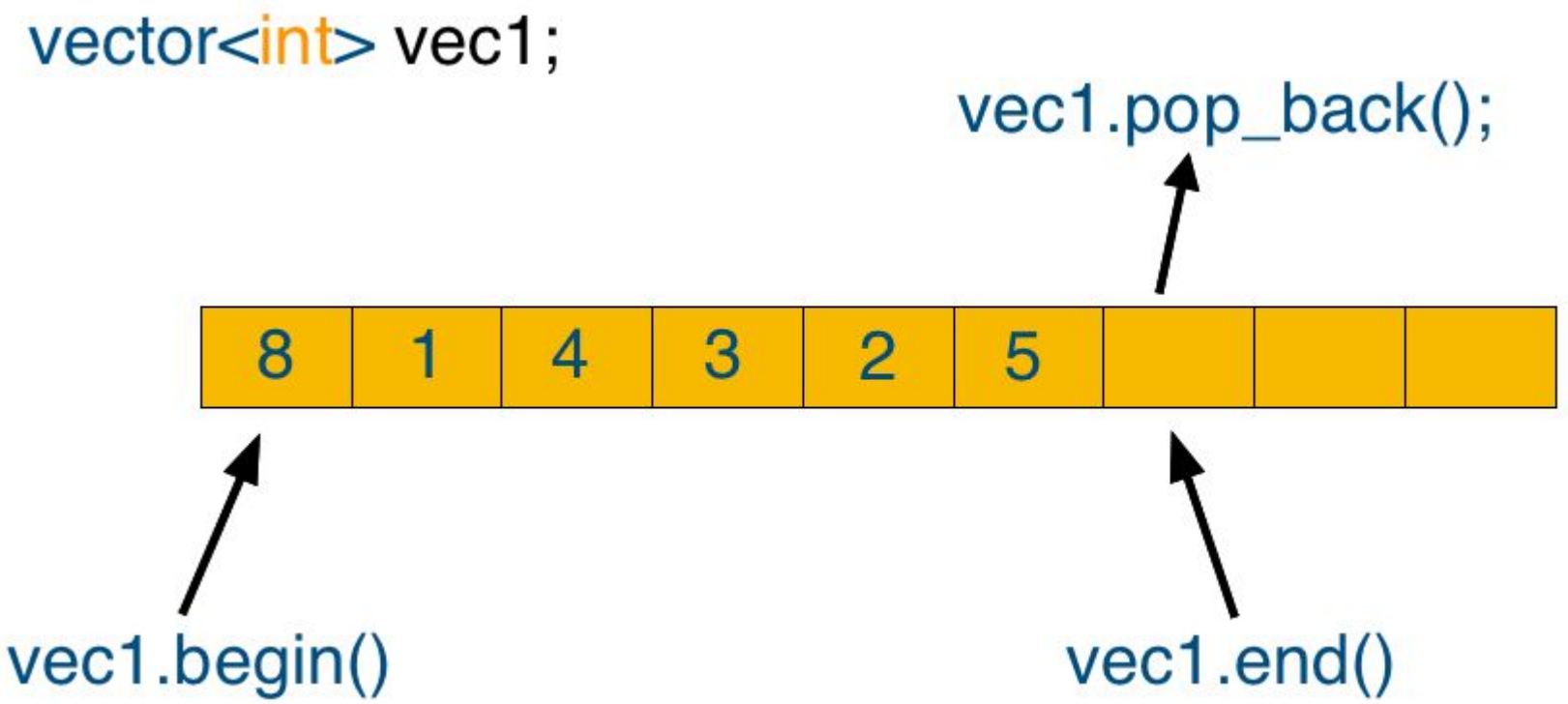
- El vector permite el borrado en cualquier posición (entre valores) pero no al inicio, así también puede borrar su contenido en totalidad.
- El borrado reduce el tamaño del vector pero no afecta el espacio

```
std::vector<int> v = {10, 20, 30, 40, 50, 60, 70, 80, 90};
v.pop_back();              // Borrado al final {10, 20, 30, 40, 50, 60, 70, 80}
// Borrado en cualquier posición, ejemplo en la segunda posición
v.erase(begin(v)+1);          // {10, 30, 40, 50, 60, 70, 80}
for (const auto & i :v)
  std::cout << i << " ";     // 10 30 40 50 60 70 80
std::cout << std::endl;
// (clear) Borrado total y (shrink_to_fit) reduce lo reservado
v.clear();
v.shrink_to_fit();
for (const auto & i :v)
  std::cout << i << " ";     // No imprime nada
std::cout << std::endl;
```

Borrar elementos de un vector (ejemplo03)

Borrar elementos del vector:

```
vec1.pop_back();  //borra un elemento del final del vector
```



Ejercicio 1

La tabla de caracteres ASCII asigna a cada carácter de 8 bits un código desde el 0 hasta el 255, a este código se conoce como código ASCII.
Si realizamos el **typecasting** de un **char** a **int** podremos verificar que por ejemplo la letra 'A' tiene código ASCII 65 o que la ñ tiene código ASCII 164.

Escribir un programa que lea un texto y haciendo uso de un vector contar las veces que se repita cada carácter del texto que se ha leído.

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Resumen

En esta sesión aprendimos:

- Cómo construir un vector
- Las operaciones básicas (CRUD)
  - Creación
  - Lectura y recorrido
  - Actualizar
  - Borrar datos

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

¡Nos vemos en la siguiente clase!

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA