VECTOR

Marco teórico:

Definición de Vector:

Un vector es un contenedor secuencial que actúa como un arreglo que cambia de tamaño.

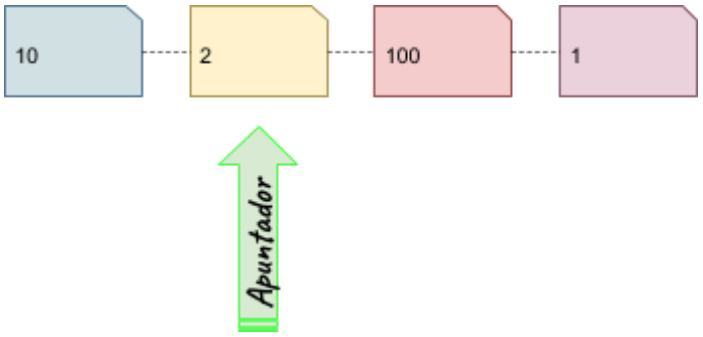
Intuición de Vector:

Como los arreglos, previamente vistos, un vector sirve para almacenar datos de forma contigua en la memoria, pero para poder cambiar de tamaño entonces este hace uso de los **dynamic allocated arrays**, esto quiere decir que haremos uso de arreglos dinámicos.

En líneas generales, usar **memoria dinámica** implica crear variables que no son **declaradas** antes del tiempo de ejecución, sino durante de él. El problema de la memoria dinámica es el costo en tiempo, ya que la reservación, búsqueda y recuperación de información se ralentiza.

Cuando usamos la memoria dinámica los espacios de memoria se van asignando de forma relativamente aleatoria, pero cuando usamos **dynamic allocated arrays señalamos** los espacios de memoria de manera contigua (sin reservar), así por medio de apuntadores es como se puede acceder a estos, logrando depositar, consultar o retirar información.

De forma intuitiva, como ya se mencionaba se puede ver con la siguiente ilustración, donde se puede ver que a diferencia de cuando se usaban los arreglos estáticos en este caso los espacios de memoria no se reservan de forma contigua pero no fija, lo que nos permite borrar la información de alguno y en cierto sentido desconectarlo de la secuencia que vemos



Declaración y funciones importantes:

Los vectores pertenecen a la biblioteca **#include <vector.h>** la cual a parte de tener una implementación bastante optimizada de estos, nos proporciona diversas funciones (operaciones) que nos permiten operarlos de una forma relativamente rápida.

Por parte de la implementación, esta consiste de tres secciones:

1. Nombre de la estructura vector
2. Tipo de dato <tipo de dato>
3. Nombre de variable miVector

En cuento código, este se escribe de la siguiente forma:

vector<tipo de dato> miVector

Cabe destacar que los tipos de datos pueden variar en cuanto se desee.

Para hablar de las funciones importantes, hablaremos de aquellas en la biblioteca ya mencionada, pero también es importante mencionar algunas fuera de ella. En cuanto las de la misma biblioteca las dividiremos como:

- Iteradores

  begin - Nos devuelve un iterador que apunta al primer elemento del vector
  Uso: miVector.begin()
  end - Nos devuelve un iterador que apunta al último elemento del vector.
  Uso: miVector.end()

- Capacidad

  .size() - Nos retorna el número de elementos almacenados en nuestro vector.
  Uso: miVector.size()

  .empty() - Nos devuelve un True o False, dependiendo de si el vector tiene elementos o no.
  Uso: miVector.empty()

- Acceso a elementos

  [] - Al igual que en los arreglos estáticos, se usa este operador para poder acceder al elemento que deseamos.
  Uso: miVector[i], con i el índice de una entrada existente.

  .front() - Nos devuelve el primer elemento en el vector.
  Uso: miVector.front()

  .back() - Nos devuelve el último elemento en el vector.
  Uso: miVector.back()

- Modificadores

.push_back() - introduce el valor que le indiquemos por detrás del vector.
Uso: miVector.push_back(valor)

.pop_back() - Elimina el último elemento del vector.
Uso: miVector.pop_back()

.erase() - Permite eliminar el valor que deseemos, de la posición que queramos.
Uso: miVector.erase(iterador apuntando a la posición deseadas)
miVector.erase(iterador apuntanto a inicio de rango, iterador apuntando al final)

.clear() - Elimina todos los elementos del vector.
Uso: miVector.clear()

Con respecto a las funciones fuera de la biblioteca ya mencionada cabe destacar las siguientes, que se ecuntran en la biblioteca algorithm:

count() - Retorna las veces que aparece un elemento en elvector.
Uso: count(inicio de rango, final de rango, elemento)

find() - Devuelve un apuntador al primer elemento encontrado en un rango, si no lo
encuentra devuelve el último elemento del vector.
Uso: find(inicio de rango, final de rango, valor)

search() - Devuelve un apuntador al primer elemento de un rango determinado, si no encuentra devuelve el último elemento del vector.
Uso: search(inicio de conjunto, final de conjunto, inicio del rango, final del rango)

reverse() - Invierte el orden de los elementos.
Uso: reverse(inicio de rango, final de rango)