Estructuras de Datos Estándar en C++

Lautaro Lasorsa

Curso OIA UNLaM 2021

Contenidos

Estructuras de datos

Estructuras de datos

Una estructura de datos es una forma de organizar la información con el objetivo de poder manipularla con mayor facilidad, comodidad y eficiencia.
Por lo general una estructura de datos se compone de elementos atómicos que para la estructura tienen características similares.

El vector

El ejemplo más sencillo y que ya vimos en este curso es el vector, y cada uno de esos elementos individuales son las posiciones del mismo (que, para el vector, son simplemente posiciones sin profundizar en su contenido)

Biblioteca Estandar de C++

Estructuras ya implementadas

Como vimos con el vector, muchas estructuras de datos ya están implementadas en C++ y se encuentran almacenadas en la biblioteca estándar, por lo que podemos usarlas siempre que lo deseemos.

Abstracción (encapsulamiento)

Con las estructuras y funciones que encontremos en la biblioteca estándar de C++ ocurre que hay una capa de abstracción. Estas estructuras y funciones nos ofrecen garantías (por ejemplo, de rendimiento) y nos dan una forma de utilizarlas, pero no nos interesa su implementación interna.

Código para incluir toda la biblioteca estándar de C++

```
1 | #include <bits\stdc++.h>
```

Contenidos

Anidación

¿Qué es la anidación?

En estructuras de datos se habla de anidación cuando los elementos de una estructura son a su vez estructuras de datos.
Pueden ser del mismo o de distinto tipo de estructura de datos.

Ajemplo

El ejemplo más sencillo que podemos encontrar de anidación es la matriz.
Una matriz puede pensarse como un vector de vectores, es decir, un vector en el que cada posición es a su vez otro vector.

Contenidos

Matriz en C++

Implementación

Si nuestro objetivo es crear una matriz de enteros en C++, de *N* filas y *M* columnas, la forma más usual de hacerlo es la siguiente:

```
1 | vector<vector<int> > matriz(N,vector<int>(M,0));
```

Y podremos hacer, entre otras, las siguientes acciones con la matriz:

```
1 | matriz.size(); //Nos da la cantidad de filas, N.
2 | matriz[0].size(); //Nos da la cantidad de columnas,M.
3 | /*Especificamente, nos da el largo de la primera fila.
4 | Lo que es importante en el caso de que las filas
5 | tengan distinta longitud */
6 | matriz[i];//Accedemos a la i-esima fila como vector.
7 | matriz[i][j]//Accedemos al j-esimo elemento de la i-
  |     esima fila
```

Más operaciones sobre matrices.

Biblioteca algorithm

También podemos utilizar las funciones de la biblioteca algorithm con las matrices.

```
1        sort(matriz.begin(),matriz.end());
2        //Ordena las filas de la matriz en orden lexicografico
3        sort(matriz[i].begin(),matriz[i].end());
4        // Ordena la i-esima fila de la matriz
5        reverse(matriz.begin(),matriz.end());
6        //Invierte el orden de las filas de la matriz.
7        reverse(matriz[i].begin(),matriz[i].end());
8        //Invierte la i-esima fila de la matriz.
```

Inicializar matriz manualmente

Código

También podemos inicializar una matriz de forma manual

```
vector<vector<int> > matriz ={
    {1,2,3,4,5},
    {6,7,8,9,10},
    {11,12,13,14,15,16},
    {},
    {17,18,19,20}
};
// Notar que las filas pueden tener diferente longitud
```

Contenidos

Orden en una fila o cola

Primero en Entrar, Primero en Salir

En una fila lo normal es que la primer persona en llegar a la fila sea la primera persona en ser atendida.
Este orden se denomina usualmente FIFO, del ingles First In, First Out.

En C++

En C++, está idea se implementa en la estructura queue, es decir cola.

Contenidos

Inicializar matriz manualmente

Queue en C++

Por ejemplo, si deseamos crear una cola de enteros el código sería el siguiente

```
1 |     queue<int> Q;// La cola comienza vacia
```

Dada la queue Q, tendremos las siguientes operaciones.

```
1 |     Q.empty();// Nos indica si no hay elementos en la cola
2 |     Q.size();//Nos da la cantidad de elementos en la cola
3 |     Q.push(x);//Agrega x al principio de la cola.
4 |     Q.front();//Nos devuelve el elemento al final de la
  |         cola
5 |     Q.pop();//Elimina el elemento al final de la cola.
```

Contenidos

Prioridad

Elementos prioritarios

Si bien como dijimos antes lo usual es que cuando hay personas esperando a ser atendidas se atienda primero a quienes llegaron antes, hay veces que el orden de atención depende de la importancia que se le asigne a quienes estan esperando.
Es decir, hay un orden de prioridad entre quienes esperan ser antedidos.

En C++

En C++, está idea se implementa en la estructura priorty_queue, es decir cola de prioridad.
Cuya idea es que puede saber siempre quién es el elemento prioritario entre los que estan dentro de la estructura.

Contenidos

Inicializar matriz manualmente

Priority_Queue en C++

Por ejemplo, si deseamos crear una cola de prioridad de enteros el código sería el siguiente

```
priority_queue<int> PQ;// La cola comienza vacia
```

Dada la priority_queue PQ, tendremos las siguientes operaciones.

```
PQ.empty();// Nos indica si no hay elementos en PQ
PQ.size();//Nos da la cantidad de elementos en PQ
PQ.push(x);//Agrega x a PQ.
PQ.top();//Nos devuelve el mayor elemento en PQ.
PQ.pop();//Elimina el mayor elemento en PQ.
```

Contenidos

Idea

¿Qué es un conjunto?

Por conjunto entendemos el tener varios elementos, sin importar su orden y sin contar repetidos. Por ejemplo:

```
1 {1, 2, 3}
2 {3, 2, 1}
3 {1, 1, 1, 1, 2, 3}
```

Notar que, como no nos importa el orden de los elementos y no consideramos elementos repetidos, los 3 conjuntos del ejemplo son iguales.

En C++

En C++, está idea se implementa en la estructura set, es decir conjunto.

Contenidos

Inicializar matriz manualmente

Set C++

Por ejemplo, si deseamos crear un conjunto de enteros el código sería el siguiente

```
1 set<int> S;// El conjunto comienza vacio
```

Dado el set S, tendremos las siguientes operaciones.

```
1 S.empty();// Nos indica si no hay elementos en el
      conjunto
2 S.size();//Nos da la cantidad de elementos en el
      conjunto
3 S.insert(x);//Agrega x al conjunto S.
4 (S.find(x)!=S.end());//Nos indica si x esta en el
      conjunto
5 S.erase(x);//Elimina el elemento x, si existe, del
      conjunto
```

Más operaciones sobre set

Otras operaciones con el set de C++

Podemos crear un set indicando sus elementos iniciales, de la siguiente forma:

```
set<int> S = {1,2,3};
// Crea el set con los elementos {1,2,3}
```

También podemos comparar dos conjuntos, que serán iguales si tienen los mismos elementos.

```
(S1==S2);
\\ Nos indica si los sets S1 y S2 son iguales
```

Por último, podemos también acceder facilmente al elemento más chico del set de la siguiente forma.

```
(*S.begin());
\\ Nos da el valor del elemento mas chico del conjunto
```

Contenidos

Mapear un conjunto en otro

¿Qué es mapear?

La idea de mapear se puede pensar de forma similar a lo que entendemos por función.
Es asignarle, a cada elemento de un conjunto, un elemento de otro conjunto.

En C++

En C++, está idea se implementa en la estructura map.
Podemos pensarlo también como un vector donde el índice puede ser un elemento de cualquier tipo, y donde las posiciones se crean a medida que accedemos a ellas.

Contenidos

Estructura en C++

Map C++

Por ejemplo, si deseamos crear un map que relacione strings con enteros lo haremos de la siguiente forma.

```
1 |     map<string,int> M;// El map comienza vacio
```

Dado el mapa M, tendremos las siguientes operaciones.
A los elementos que usamos como índices se los llama key o llave.

```
1 |     M.empty();// Nos indica si no hay elementos en el map
2 |     M.size();//Nos da la cantidad de elementos en el map
3 |     M[x];//Nos permite acceder a la posicin x del map
4 |     (M.find(x)!=M.end());//Indica si x es llave en el map
5 |     M.erase(x);//Elimina la posicion con llave x del map
```

Si queremos leer una posición no inicializada nos dará el valor por defecto de ese tipo de dato.(ej, los int se inicializan 0)

Contenidos

Iterar estructuras en C++

Estructuras iterables

En C++, hay estructuras en las que podemos recorrer los elementos que estas estructuras contienen.
La forma más sencilla de hacerlo es de la siguiente forma, por ejemplo con el vector v:

```
1        for(auto x : v){
2            /// Codigo
3        }
```

En este caso, x tomará el valor de cada uno de los elementos que contiene v en el orden que están en v. (para cada valor se realiza una iteración del ciclo, lógicamente)

Iterar por referencia

Operador

En cambio, si lo que deseamos es crear referencias a los valores de v (de tal forma que si las modificamos se modifican los valores en v) debemos hacer lo siguiente:

```
1 |     for(auto & x : v){
2 |         /// Codigo
3 |     }
```

Iterar sets

Set

Identicamente a como hicimos con el vector, podemos hacer con el set. Tener en cuenta que estarán siempre ordenadas de menor a mayor.

```
for(auto x : S){
    /// Codigo, pasa x por valor
}
```

```
for(auto & x : S){
    /// Codigo, pasa x por referencia
}
```

Iterar maps

Map

En el caso del map los elementos estarán ordenados según el valor de la llave (de menor a mayor), y al iterar lo que obtendremos es un pair donde el primer elemento es la llave y el segundo el valor.

```
for(auto x : M){
    /// Codigo, pasa x por valor
    /// x.first es la llave y x.second el valor
}
```

```
for(auto & x : M){
    /// Codigo, pasa x por referencia
    /// x.first es la llave y x.second el valor
}
```