Nuevas estructuras de datos en c++

Ayuda para la guía de laboratorio #3

pair<class T1, class T2>

pair<class T1, class T2>

- Está incluído en el header `<utility>`, por lo que hay que agregar `#include <utility>` para usarlo.
- Esta estructura de datos es un caso particular de una tupla; corresponde a una tupla de dos elementos.
- El primer elemento (T1) puede ser de tipo diferente al segundo elemento (T2)
- Para acceder a los elementos, usamos el operador “.” (punto) seguido del nombre del atributo que corresponda:
  - Para acceder al primer elemento, se usa el atributo *first*.
  - Para acceder al segundo elemento, se usa el atributo *second*.

pair<int, int> - Ejemplo

- Declarar una variable “v” de tipo pair<int,int>:

```
pair<int,int> v;
```

- Asignar “6” al primer elemento y “3” al segundo elemento:

```
v.first = 6;
v.second = 3;
```

- Mostrar por pantalla “(a,b)” donde “a” es el primer elemento y “b” el segundo elemento:

```
cout << "(" << par.first << ", " << par.second << ")" << endl;
```

pair<int, int> - Ejemplo

- Diferentes formas de inicializar una variable de tipo pair<T1,T2>:

```
pair<int,int> par1 = make_pair(4, 5);
pair<int,int> par2(5, 6);
pair<int,int> par3(par2); // es una copia de par2
pair<int,char> par4(1, ‘b’); //notar que los elementos son de diferente tipo
```

vector<pair<int, int>> - Ejemplo

- Declarar una variable “v” de tipo vector<pair<int,int>>:

```
vector<pair<int,int>> v;
```

- Asignar al vector *v* un elemento de tipo pair<int,int> cuyo primer elemento tenga “1” y “2” el segundo elemento:

```
pair<int,int> par(1,2); // Inicializo par de tipo pair<int,int>
v.push_back(par); //agrego par al vector v
```

- Mostrar por pantalla el primer elemento de “v” de la forma “(a, b)”:

```
cout << "(" << v1[0].first << ", " << v1[1].second << ")" << endl;
```

string

string

- Está incluído en el header `<string>` por lo que es necesario agregar `#include <string>`
- Es una estructura de datos que representa una secuencia de caracteres.
- Lo usamos para almacenar texto

string - Ejemplo

- Declarar una variable de tipo string:

```
string s;
```

- Asignar un valor a la variable s:

```
s = "Hola Mundo!";
```

- Imprimir por pantalla el valor s:

```
cout << s << endl;
```

- Concatenar dos strings con el operador “+”:

```
string s = "Hola";
string s2 = " Mundo!";
string s3 = s + s2;
cout << s3; //Imprime “Hola Mundo!”
```

string - Ejemplo

- Se puede acceder a la i-ésima posición (comenzando desde cero) de diferentes formas:
  - opción 1: `s[i]`
  - opción 2: `s.at(i)`
- Ejemplo

```
string s = "Hola Mundo!";
cout << s.at(3) << endl; // Imprime “a”
cout << s[0] << endl; // Imprime “H”
```

string - Ejemplo

- Para conocer el tamaño de un string se pueden usar los métodos size() y length() (tienen el mismo comportamiento)

```
string s = "Hola";
int size = s.size();
int size2 = s.length();
cout << size << endl; // Imprime 4
cout << size2 << endl; // Imprime 4
```

string - Ejemplo

- Se puede asignar el valor de entrada de un usuario a una variable de tipo string:

```
string input;
cout << "Ingrese su nombre: ";
cin >> input; // El usuario ingresa "Cosme Fulanito"
cout << "Su nombre es: " << input; // ¿Qué se imprime por pantalla?
```

string - Ejemplo

- Se puede asignar el valor de entrada de un usuario a una variable de tipo string:

```
string input;
cout << "Ingrese su nombre: ";
cin >> input; // El usuario ingresa "Cosme Fulanito"
cout << "Su nombre es: " << input; // Imprime “Cosme”
```

- *cin* considera a los espacios (en blanco, tabulaciones, etc) como caracteres de terminación ¡por lo que sólo puede mostrar una palabra!

string - Ejemplo

- Para leer una entrada de texto solemos usar el método getline(istream& is, string& str): extrae el texto de *is* y lo almacena en *str*.
- En este ejemplo asigna la entrada del usuario (*cin*) al parámetro *input*.

```
string input;
cout << "Ingrese su nombre: ";
getline(cin, input); // El usuario ingresa "Cosme Fulanito"
cout << "Su nombre es: " << input; // Ahora sí imprime "Cosme Fulanito"
```

Función toupper()

- Está incluída en el header `<cctype>` por lo que es necesario agregar `#include <cctype>`.
- *int toupper (int c)*: convierte el parámetro que recibe (un caracter) a mayúsculas. Si no tiene una versión en mayúsculas, entonces devuelve el mismo caracter.
- Notar que devuelve un *entero* que representa el caracter *c* en mayúsculas.

Función toupper()

- Convertir un string a mayúsculas:

```
string input = "Cosme Fulanito";
cout << input << endl; // Imprime "Cosme Fulanito"
string toUpperString;
for(int i=0;i<input.size(); ++i)
{
// Convierte a mayúsculas cada caracter.
// La operación “+=” hace la conversión automática de int a char
   toUpperString += toupper(input[i]);
}
cout << toUpperString << endl; // Imprime "COSME FULANITO"
```

Material de consulta

- Es común no conocer estructuras de datos de un lenguaje particular.
- Debemos acostumbrarnos a buscar esto u otros detalles técnicos de un lenguaje de programación en sitios de internet.
- Particularmente para detalles de c++, recomendamos la página http://www.cplusplus.com/
- O libros como el de B. Stroustrup, **The C++ Programming Language**.