Ayudantía 3

Carlos Lagos - carlos.lagosc@usm.cl

String

Strings en C++

En C++, los strings son una parte fundamental del manejo de texto. Pueden ser manipulados usando diversos métodos y funciones estándar.

Operaciones Básicas con `string`

A continuación se muestran algunas operaciones básicas que se pueden realizar sobre un objeto `string`.

```
string s1 = "hola";
```

Strings en C++

`s1.length()`

El método `length()` retorna el largo del string `s1`, es decir, la cantidad de caracteres que contiene.

Ejemplo:

```
size_t largo = s1.length(); // 4
int largo = (int) s1.length();
```

Strings en C++

`s1.empty()`

El método `empty()` verifica si el string `s1` está vacío, retornando `true` si no contiene caracteres.

Ejemplo:

```
bool vacio = s1.empty(); // false
```

Strings en C++

`s1[i]`

Se puede acceder al i-ésimo carácter de `s1` usando el operador `[]`.

Ejemplo:

```
char c = s1[1]; // 'o'
```

Strings en C++

`s1 + s2`

El operador `+` permite concatenar dos strings, resultando en un nuevo string que combina ambos.

Ejemplo:

```
string s2 = " mundo";
string resultado = s1 + s2; // "hola mundo"
```

Strings en C++

`s1 == s2`

El operador `==` compara dos strings, retornando `true` si ambos son iguales.

Ejemplo:

```
bool iguales = (s1 == s2); // false
```

Strings en C++

`s1.find(...)`

`find` retorna la posición de la primera aparición de un carácter o un substring en `s1`.

Ejemplo con carácter:

```
size_t posicion = s1.find('h'); // 0
```

Ejemplo con substring:

```
size_t posicion_sub = s1.find("la"); // 2
```

Archivos

Lectura y Escritura de Archivos en C++

En C++, se pueden realizar operaciones de lectura y escritura de archivos utilizando las clases `fstream`, `ifstream` y `ofstream` del estándar de la biblioteca de C++.

Clases de Flujo de Archivos

- `ifstream`: Clase para lectura de archivos.
- `ofstream`: Clase para escritura de archivos.
- `fstream`: Clase para lectura y escritura de archivos.

Abriendo un Archivo

Para abrir un archivo para lectura, escritura o ambas, se utiliza un objeto de la clase correspondiente.

```
ifstream archivo_lectura;
archivo_lectura.open("datos.txt", ios::in);

ofstream archivo_escritura;
archivo_escritura.open("resultado.txt", ios::out);
```

Abriendo un Archivo en Modo Binario

Para trabajar con archivos en modo binario, se utiliza el indicador `ios::binary` al abrir el archivo. Esto es útil cuando se maneja información que no es texto, como estructuras o imágenes.

```
ifstream archivo_lectura_binario;
archivo_lectura_binario.open("datos.bin", ios::in | ios::binary);

ofstream archivo_escritura_binario;
archivo_escritura_binario.open("resultado.bin", ios::out | ios::binary);
```

Lectura y Escritura en Modo Binario

Para leer y escribir en archivos binarios, se utilizan las funciones `read` y `write`. Estas permiten manipular datos en bruto directamente desde o hacia el archivo.

Lectura y Escritura en Modo Binario

Escritura Binaria:

```
int numero = 42;
archivo_escritura_binario.write((char*)&numero, sizeof(numero));
```

Lectura Binaria:

```
int numero_leido;
archivo_lectura_binario.read((char*)&numero_leido, sizeof(numero_leido));
```

Uso de `fstream`

La clase `fstream` se puede utilizar para abrir archivos en modo lectura, escritura o ambos.

```
fstream archivo;
archivo.open("datos.txt", ios::in | ios::out);
```

Lectura de Archivos

Para leer datos desde un archivo, se utilizan operaciones de lectura de la misma manera que con `cin`.

```
tipo_dato dato;
archivo_lectura >> dato;
```

Por ejemplo:

```
int numero;
archivo_lectura >> numero;
```

Escritura en Archivos

Para escribir datos en un archivo, se utilizan operaciones de escritura de la misma manera que con `cout`.

```
tipo_dato dato;
archivo_escritura << dato;
```

Por ejemplo:

```
int numero = 10;
archivo_escritura << numero;
```

Cierre de Archivos

Es importante cerrar los archivos después de usarlos para liberar los recursos del sistema.

```
archivo_lectura.close();
archivo_escritura.close();
```

Uso de Archivos en C++

La lectura y escritura de archivos en C++ es fundamental para trabajar con datos persistentes y archivos de configuración.

```cpp
int main() {
    ifstream archivo_lectura("datos.txt");
    // ...
    archivo_lectura.close();
    return 0;
}
```

Ejercicios

Ejercicios

Dado un archivo `vectores.txt`, con un entero n en la primera línea seguido de n líneas con dos enteros x_i y y_i que representan vectores desde el origen con coordenadas (x_i, y_i):

1. Imprime los vectores en formato:
   `vectores = [<x_1,y_1>, ..., <x_n,y_n>]` usando strings.
2. Define un `struct` para representar cada vector.
3. Escribe un archivo binario `datos.dat` con n (en binario) y las instancias del `struct` (en binario).

Ejercicios

Usando el archivo `datos.dat`, que contiene vectores en formato binario:

1. Lee los vectores del archivo.
2. Cuenta la cantidad de pares de vectores que son perpendiculares entre sí.

FIN