Karla Verónica Quiros Delgado

Archivos

Puntos importantes:

1. Para guardar un objeto se crea un método de tipo void el cual recibe una referencia (&) de la clase ostream. En este método guardaremos cada atributo del objeto usando el siguiente formato:
   La clase ostream es similar “cout”, ostream es una clase con un flujo de salida, es decir, un flujo de datos de salida, datos que se envían a un archivo.

```
1.  void Persona::guardar(ostream& salida)
2.  {
3.    salida << nombre << '\t'; // se guarda el nombre de la persona, se usa el tabulador para
    separar los datos
4.    salida << cedula << '\t';// se guarda la cedula de la persona, se usa el tabulador para
    separar los datos
5.    salida << edad << '\n';// se guarda la edad de la persona, se usa el salto de linea para
    separar los datos
6.    // el salto de linea es para que cuando se lea el archivo se separe los datos de cada
    persona
7.  }
8.
```

Se necesita usar tabulador para separar los atributos de la clase, siempre al guardar el ultimo atributo se usa salto de línea, para cuando usemos el método guardar en un array o lista, matriz etc. Se separen la información de los objetos por medio de un salto de línea.

2. Para leer un objeto creamos un método static el cual nos permitirá leer un objeto antes de ser creado, siempre se retornará un puntero del tipo de objeto que se lee, es decir si vamos a leer una persona se hará un método static de tipo Persona*, que devolverá una persona que fue leída.
   Usaremos getline para leer cada atributo, se recuperan strings siempre del archivo de texto, es por esta razón que usaremos la ayuda de métodos que conviertan string a números para proceder con la creación del objeto que se lee en caso de tener atributos distintos a String.
   Se utiliza un formato como, por ejemplo:
   Al igual que cuando guardamos separamos con tabulador cada atributo y al final usamos salto de línea.

```
1.  Persona* Persona::leer(istream& entrada)
2.  {
3.    string _nombre, _cedula, _edad = "";// creamos las varibales necesarias para almacenar los
    atributos de la persona
4.    getline(entrada, _nombre, '\t');// leemos el nombre de la persona, se usa el tabulador
    para separar los datos
5.    getline(entrada, _cedula, '\t');// leemos la cedula de la persona, se usa el tabulador
    para separar los datos
6.    getline(entrada, _edad, '\n');// leemos la edad de la persona, se usa el salto de linea
    para separar los datos
7.    int valEdad = convertirInt(_edad);// convertimos el string a int
8.    return new Persona(_nombre, _cedula, valEdad);// retornamos un puntero a la persona creada
9.  }
```

```
10.
```

Ya vimos como crear los archivos de una clase en específico, veremos como guardar una lista de objetos y como leer una lista de objetos. Haciendo uso de los métodos guardar y leer creados anteriormente en cada objeto.

```
1.   void Lista::guardarPersonas()
2.  {
3.    ofstream salida;// salida de datos, usamos ofstream para escribir en un archivo
4.    Nodo* p = primero;// nodo auxiliar
5.
6.    salida.open("../ContenedorPersonas.txt");// abrimos el archivo, si no existe lo crea
7.    while (p != NULL)// mientras no sea nulo
8.    {
9.              if (salida.good()) {// si el archivo esta bien, es decir, si se abrio
    correctamente
10.
11.                       p->getPer()->guardar(salida);// llama el metodo guardar de la clase
    persona
12.             }
13.
14.             p = p->getsig();// pasa al siguiente hasta guardar toda la lista
15.
16.   }
17.
18.   salida.close();// siempre se debe cerrar el archivo al final del guardar
```

```
1.   void Lista::leerPersona()
2.  {
3.    ifstream entrada;// entrada de datos, usamos ifstream para leer un archivo
4.    Persona* perler = nullptr;// persona auxiliar
5.    entrada.open("../ContenedorPersonas.txt");// abrimos el archivo, debe existir y se manda
    el mismo nombre con el que fue guardado
6.
7.    if (entrada.good()) {// si el archivo esta bien, es decir, si se abrio correctamente
8.              while (!entrada.eof())// mientras no se haya llegado al final del archivo
9.              {
10.
11.             perler = Persona::leer(entrada);// llama el metodo leer de la clase persona y lo
    guarda en la persona auxiliar
12.                       if (perler->getCedula() != "") {// si la cedula no esta vacia, es
    decir, si se leyo correctamente(usamos este verificacion para ver que si se haya obtenido
    una persona del leer)
13.                                 ingresaPersona(perler);// ingresa la persona a la lista
14.                       }
15.
16.
17.             }
18.   }
19.   entrada.close();// siempre se debe cerrar el archivo al final del leer
20. }
```