```
uriel@uriel:~/Documents/CUCEI/estructuras de

Menú - Administración de cheques
1. Registrar Cheque - (insertar)
2. Procesar Cheque - (eliminar)
3. Registrar Cheque Rechazado - (insertar)
4. Procesar Cheque Rechazado - (eliminar)
5. Salir
Elige tu opción:
```

```
Elige tu opción: 2

(!) Pila vacía, no hay cheques
```

```
Elige tu opción: 4

(!) Pila vacía, no hay cheques rechazados
```

```
Menú - Administración de cheques
1. Registrar Cheque - (insertar)
2. Procesar Cheque - (eliminar)
3. Registrar Cheque Rechazado - (insertar)
4. Procesar Cheque Rechazado - (eliminar)
5. Salir
Elige tu opción: 1

Número de cheque: 11
Nombre de banco: Banamex
Cuenta a depositar: 1111
Monto: 5000.50

Menú - Administración de cheques
1. Registrar Cheque - (insertar)
2. Procesar Cheque - (eliminar)
3. Registrar Cheque Rechazado - (insertar)
4. Procesar Cheque Rechazado - (eliminar)
5. Salir
Elige tu opción: 1

Número de cheque: 22
Nombre de banco: Bancomer
Cuenta a depositar: 2222
Monto: 98.24

Menú - Administración de cheques
1. Registrar Cheque - (insertar)
2. Procesar Cheque - (eliminar)
3. Registrar Cheque Rechazado - (insertar)
4. Procesar Cheque Rechazado - (eliminar)
5. Salir
Elige tu opción: 1

Número de cheque: 33
Nombre de banco: Santander
Cuenta a depositar: 3333
Monto: 45000.67

Menú - Administración de cheques
1. Registrar Cheque - (insertar)
2. Procesar Cheque - (eliminar)
3. Registrar Cheque Rechazado - (insertar)
4. Procesar Cheque Rechazado - (eliminar)
5. Salir
Elige tu opción: 1

(!) Pila de cheques llena
```

```
Elige tu opción: 2

Número de cheque        Nombre de banco             Cuenta a depositar                  Monto
33                      Santander                   3333                                4500.67
```

```
Elige tu opción: 3

Número de cheque: 11
Nombre de banco: HSBC
Cuenta a depositar: 1111
Monto: 500.25
Cargo: 500.25

Menú - Administración de cheques
1. Registrar Cheque - (insertar)
2. Procesar Cheque - (eliminar)
3. Registrar Cheque Rechazado - (insertar)
4. Procesar Cheque Rechazado - (eliminar)
5. Salir
Elige tu opción: 3

Número de cheque: 22
Nombre de banco: Banamex
Cuenta a depositar: 2222
Monto: 5
Cargo: 4

Menú - Administración de cheques
1. Registrar Cheque - (insertar)
2. Procesar Cheque - (eliminar)
3. Registrar Cheque Rechazado - (insertar)
4. Procesar Cheque Rechazado - (eliminar)
5. Salir
Elige tu opción: 3

Número de cheque: 33
Nombre de banco: Banorte
Cuenta a depositar: 3333
Monto: 500.75
Cargo: 0
```

```
Elige tu opción: 3

(!) Pila de cheques rechazados llena
```

```
Elige tu opción: 2

Número de cheque        Nombre de banco         Cuenta a depositar      Monto
33                      Santander               3333                    4500.67
```

Paso 5: Elegir opción 2 (debe mostrar en pantalla el ch

Paso 6: Elegir opción 4 (debe mostrar en pantalla el ch

```
Elige tu opción: 4

Número de cheque        Nombre de banco         Cuenta a depositar      Monto                   cargo
33                      Banorte                 3333                    500.75                  0
```

```
Elige tu opción: 5
```

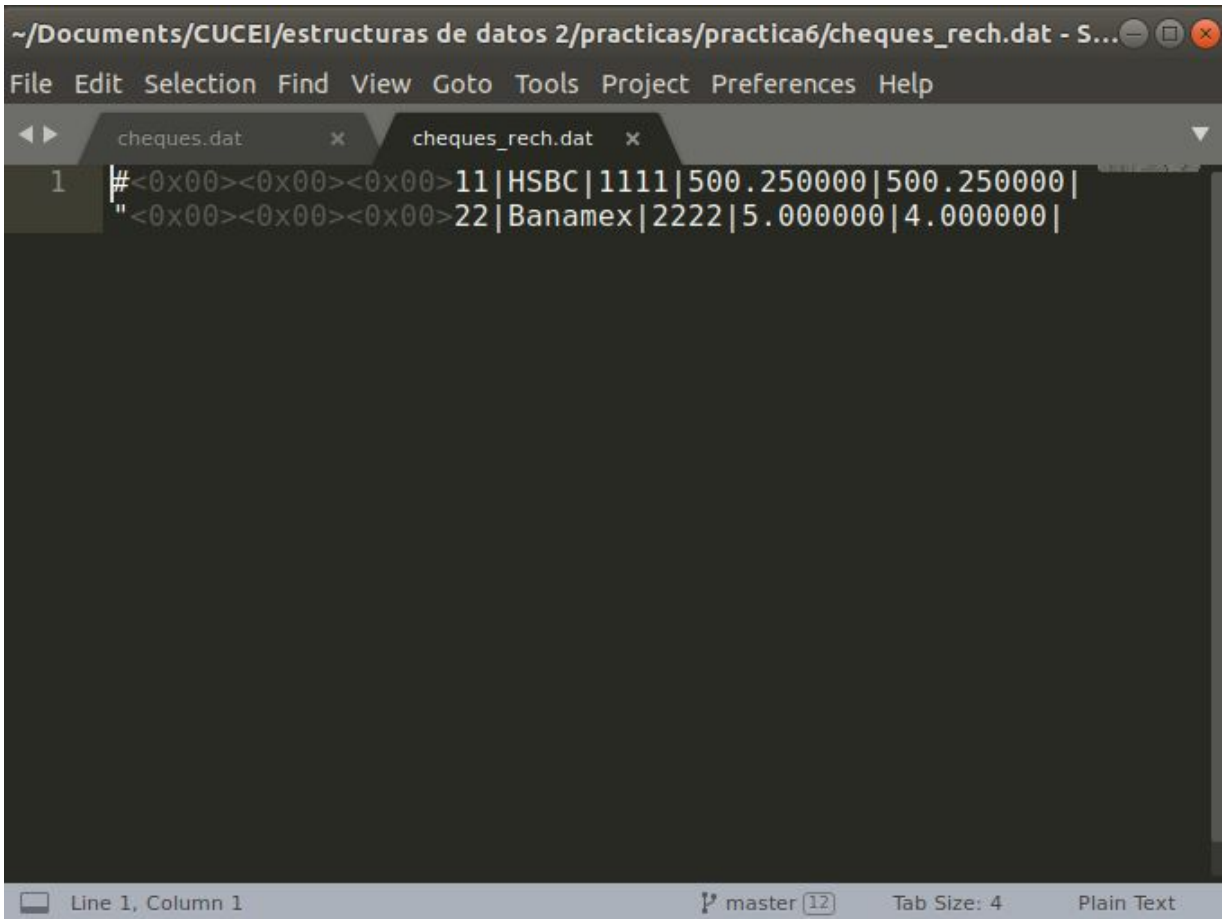
~/Documents/CUCEI/estructuras de datos 2/practicas/practica6/cheques.dat - Sublim...

File Edit Selection Find View Goto Tools Project Preferences Help

cheques.dat × | cheques_rech.dat ×

```
1  <0x1c><0x00><0x00><0x00>11|Banamex|1111|5000.500000|
   <0x1b><0x00><0x00><0x00>22|Bancomer|2222|98.240000|
```

Line 1, Column 1 | master 12 | Tab Size: 4 | Plain Text




```
Elige tu opción: 2

Número de cheque        Nombre de banco                 Cuenta a depositar                  Monto
22                      Bancomer                        2222                                98.24

Menú - Administración de cheques
1. Registrar Cheque - (insertar)
2. Procesar Cheque - (eliminar)
3. Registrar Cheque Rechazado - (insertar)
4. Procesar Cheque Rechazado - (eliminar)
5. Salir
Elige tu opción: 2

Número de cheque        Nombre de banco                 Cuenta a depositar                  Monto
11                      Banamex                 1111                                5000.5

Menú - Administración de cheques
1. Registrar Cheque - (insertar)
2. Procesar Cheque - (eliminar)
3. Registrar Cheque Rechazado - (insertar)
4. Procesar Cheque Rechazado - (eliminar)
5. Salir
Elige tu opción: 2

(!) Pila vacía, no hay cheques
```

Paso 7: Elegir opción 5 (en ese momento se guarda
Paso 8: Abrir los archivos para verificar que se haya
Paso 9: Volver a correr el programa (el programa ab
objetos a la pila que le corresponde)
Paso 10: Elegir opción 2 (dos veces seguidas para de
Paso 11: Elegir opción 4 (dos veces seguidas para de
Paso 12: Elegir opción 5 (en ese momento se guarda
OJO -> LAS PILAS ESTÁN VACÍAS
Paso 13: Abrir los archivos para verificar que se haya
OJO -> LOS ARCHIVOS DEBEN ESTAR LIMPIOS (SIN N

++
MARCO TEÓRICO
- Tomar lo que el equipo decida de la Actividad 2 de

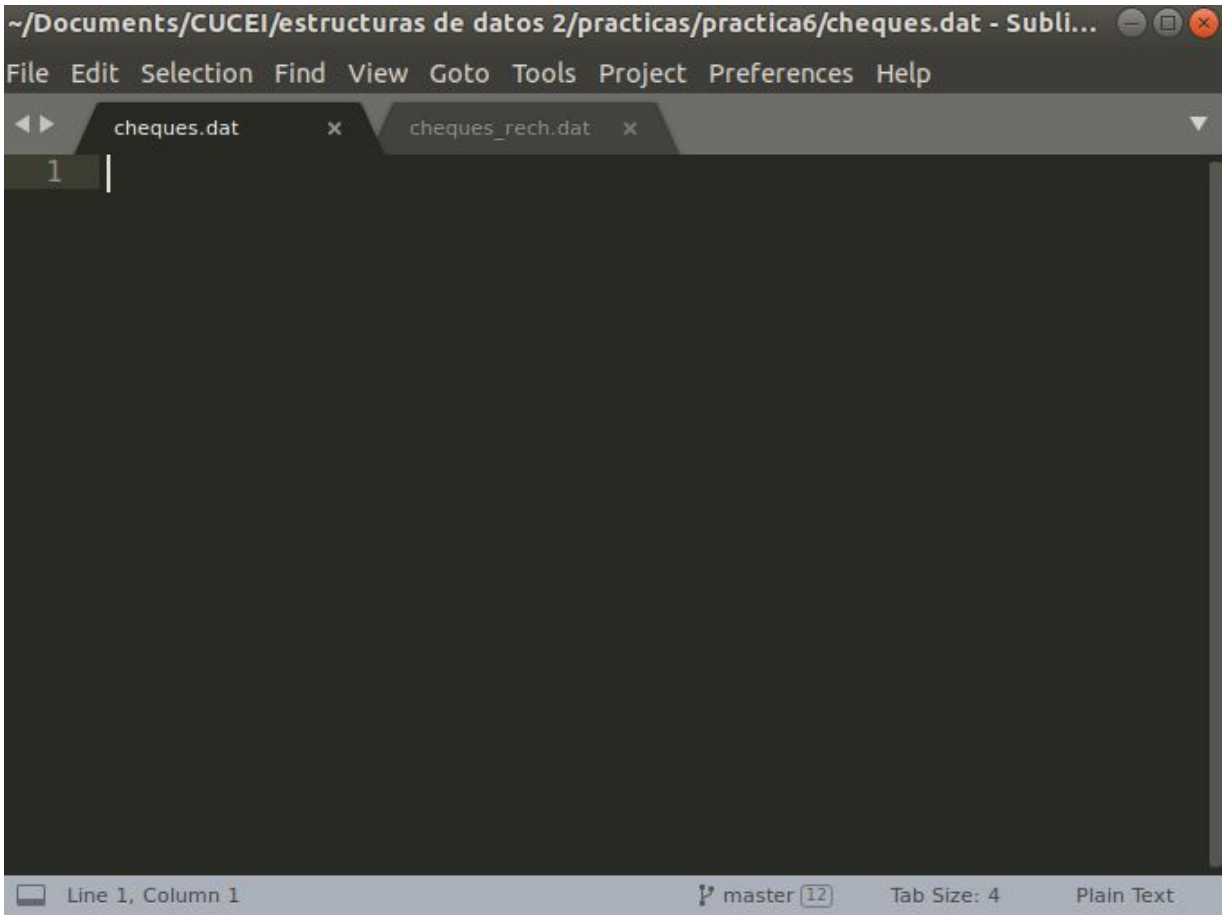
```
Elige tu opción: 4

Número de cheque        Nombre de banco
22                      Banamex

Menú - Administración de cheques
1. Registrar Cheque - (insertar)
2. Procesar Cheque - (eliminar)
3. Registrar Cheque Rechazado - (insertar)
4. Procesar Cheque Rechazado - (eliminar)
5. Salir
Elige tu opción: 4

Número de cheque        Nombre de banco
11                      HSBC

Menú - Administración de cheques
1. Registrar Cheque - (insertar)
2. Procesar Cheque - (eliminar)
3. Registrar Cheque Rechazado - (insertar)
4. Procesar Cheque Rechazado - (eliminar)
5. Salir
Elige tu opción: 4

(!) Pila vacía, no hay cheques rechazados
```

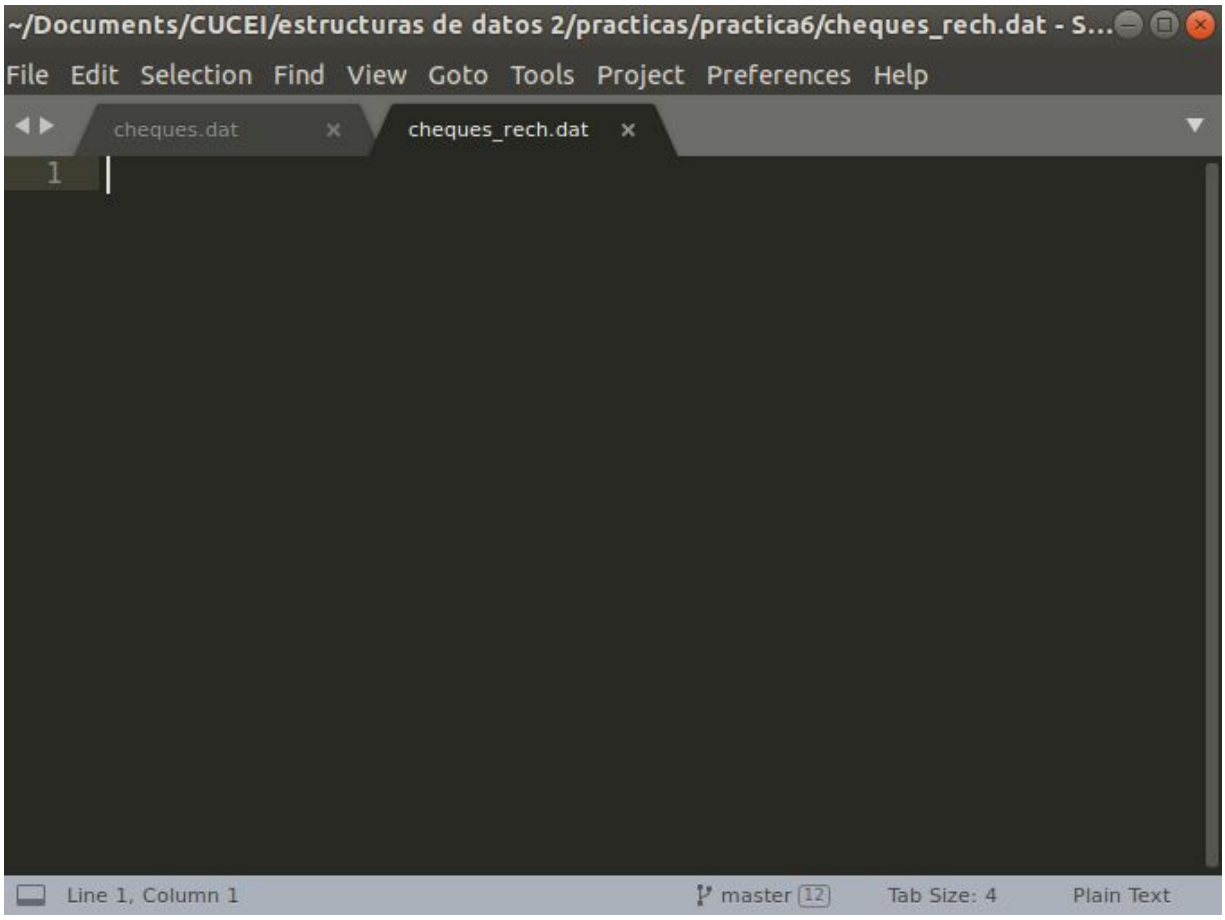
```
Cuenta a depositar      Monto                   cargo
2222                    5                       4

Cuenta a depositar      Monto                   cargo
1111                    500.25                  500.25
```

Paso 7: Elegir opción 5 (en ese momento se guarda la información (objetos) de cada pila en cad
Paso 8: Abrir los archivos para verificar que se hayan almacenado los objetos y hacer captura d
Paso 9: Volver a correr el programa (el programa abre los archivos y lee la información de cada
objetos a la pila que le corresponde)
Paso 10: Elegir opción 2 (dos veces seguidas para dejar la pila de cheques vacía)
Paso 11: Elegir opción 4 (dos veces seguidas para dejar la pila de cheques rechazados vacía)
Paso 12: Elegir opción 5 (en ese momento se guarda la información (objetos) de cada pila en ca
OJO -> LAS PILAS ESTÁN VACÍAS
Paso 13: Abrir los archivos para verificar que se hayan almacenado los objetos y hacer captura
OJO -> LOS ARCHIVOS DEBEN ESTAR LIMPIOS (SIN NINGUNA INFORMACIÓN)

++
MARCO TEÓRICO
- Tomar lo que el equipo decida de la Actividad 2 del cuaderno y de los ejercicios de pilas (progr

```
Elige tu opción: 5
```




~/Documents/CUCEI/estructuras de datos 2/practicas/practica6/cheques_rech.dat - S...
File Edit Selection Find View Goto Tools Project Preferences Help
cheques.dat
cheques_rech.dat
1
Line 1, Column 1
master 12
Tab Size: 4
Plain Text

```cpp
// Constructor por defecto
template<typename T>
Pila<T>::Pila() {
  tope = -1;
}

// Revisar si está vacía la pila
template<typename T>
bool Pila<T>::vacio() {
  if (tope == -1)
    return true;
  return false;
}

// Revisar si está llena la pila
template<typename T>
bool Pila<T>::lleno() {
  if (tope == MAX-1)
    return true;
  return false;
}

// Insertar en la pila
template<typename T>
void Pila<T>::insertar(T x){
  if (tope < MAX-1)
    a[++tope] = x;
}

// Eliminar un objeto de la pila
template<typename T>
void Pila<T>::eliminar(){
  if (!vacio())
    tope--;
}

// Obtener el elemento al tope de la pila
template<typename T>
T Pila<T>::arriba(){
  if (!vacio())
    return a[tope];
}
```

```cpp
#include <array>
#include <iostream>
using namespace std;

#define MAX 3 // máximo número de elementos que puede contener la pila

template<typename T>
class Pila {
  public:
    typedef typename array<T, MAX>::size_type size_type;

    Pila<T>(); // Constructor por defecto
    ~Pila<T>() = default; // Destructor

    void insertar(T x); // insertar en la pila
    void eliminar(); // eliminar de la pila
    bool vacio(); // revisar si la pila está vacía
    bool lleno(); // revisar si la pila está llena
    T arriba(); // obtener el elemento al tope de la pila
  private:
    array<T, MAX> a; // arreglo que contiene los elementos de la pila
    int tope; // apuntador de elementos de la pila
};
```

```cpp
// Práctica #6
// Equipo 2
//  Almazán de la Torre Rubén
//  Dueñas Becerra Mario Alejandro
//  Guzmán Mendoza Jesús Uriel
//  Martínez Yañez Astra Bernardino
//  Torres García Oscar
//  Vázquez Martínez Edgar Isaias
// Sección: D-05
// Calendario: 2020-A

#include "Menu.h"
#include <fstream>

int main() {
  Menu menu;
  Pila<Cheque> cheques; // pila de cheques
  Pila<ChequeRechazado> chequesRechazados; // pila de cheques rechazados

  fstream archivoCheques("cheques.dat", ios::in | ios::out | ios::app); // archivo de cheques
  fstream archivoChequesRechazados("cheques_rech.dat", ios::in | ios::out | ios::app); // archivo de
  // cheques rechazados

  menu.leer(archivoCheques, cheques); // leer datos de cheques
  menu.leer(archivoChequesRechazados, chequesRechazados); // leer datos de cheques rechazados

  string op;
  do {
    menu.mostrar(op);
    if (op == "1")
      menu.registrarCheque(cheques);
    else if (op == "2")
      menu.procesarCheque(cheques);
    else if (op == "3")
      menu.registrarChequeRechazado(chequesRechazados);
    else if (op == "4")
      menu.procesarChequeRechazado(chequesRechazados);
    else if (op == "5") {
      menu.guardar(archivoCheques, cheques); // guardar información sobre cheques
      menu.guardar(archivoChequesRechazados, chequesRechazados); // guardar información sobre cheques
      // rechazados
    } else
      cout << "\n(!) No existe esta opción\n";
  } while (op != "5");
}
```

```cpp
void mostrar(string &op) {
  cout << '\n';
  cout << "Menú - Administración de cheques\n";
  cout << "1. Registrar Cheque - (insertar)\n";
  cout << "2. Procesar Cheque - (eliminar)\n";
  cout << "3. Registrar Cheque Rechazado - (insertar)\n";
  cout << "4. Procesar Cheque Rechazado - (eliminar)\n";
  cout << "5. Salir\n";
  cout << "Elige tu opción: ";
  cin >> op;
  cout << "\n";
}
```

```
void registrarCheque(Pila<Cheque> &cheques) {
  if (cheques.lleno()) {
    cout << "(!) Pila de cheques llena\n";
    return;
  }
  int num_cheque;
  string nombre_banco;
  int cuenta_depositar;
  double monto;

  // leer atributos
  cout << "Número de cheque: ";
  cin >> num_cheque;

  cout << "Nombre de banco: ";
  cin >> nombre_banco;

  cout << "Cuenta a depositar: ";
  cin >> cuenta_depositar;

  cout << "Monto: ";
  cin >> monto;

  // insertar objeto cheque a la pila de cheques
  Cheque c(num_cheque, nombre_banco, cuenta_depositar, monto);
  cheques.insertar(c);
}

void procesarCheque(Pila<Cheque> &cheques) {
  // si está vacío, imprimir, si no, mandar mensaje
  if (!cheques.vacio()) {
    cheques.arriba().imprimir();
    cheques.eliminar();
  } else
    cout << "(!) Pila vacía, no hay cheques\n";
}
```

```
void registrarChequeRechazado(Pila<ChequeRechazado> &cheques_rechazados) {
  if (cheques_rechazados.lleno()) {
    cout << "(!) Pila de cheques rechazados llena\n";
    return;
  }
  int num_cheque;
  string nombre_banco;
  int cuenta_depositar;
  double monto;
  double cargo;

  // leer atributos
  cout << "Número de cheque: ";
  cin >> num_cheque;

  cout << "Nombre de banco: ";
  cin >> nombre_banco;

  cout << "Cuenta a depositar: ";
  cin >> cuenta_depositar;

  cout << "Monto: ";
  cin >> monto;

  cout << "Cargo: ";
  cin >> cargo;

  // agregar clase cheque rechazado a la pila de cheques rechazados
  ChequeRechazado c(num_cheque, nombre_banco, cuenta_depositar, monto, cargo);
  cheques_rechazados.insertar(c);
}
```

```
void procesarChequeRechazado(Pila<ChequeRechazado> &cheques_rechazados) {
  // si está vacía, imprimir mensaje, si no, imprimir cheque rechazado y eliminarlo
  if (!cheques_rechazados.vacio()) {
    cheques_rechazados.arriba().imprimir();
    cheques_rechazados.eliminar();
  } else
    cout << "(!) Pila vacía, no hay cheques rechazados\n";
}
```

```
// leer información sobre cheques del archivo físico
void leer(fstream &archivo, Pila<Cheque> &cheques) {
  int len;
  string campo, num_cheque, nombre_banco, cuenta_depositar, monto;
  while (!archivo.eof()) {
    if (archivo.peek() == -1)
      break;

    // leer registro
    archivo.read(reinterpret_cast<char *>(&len), sizeof(int));
    char *buffer = new char[len];
    archivo.read(buffer, len);
    campo = buffer;
    delete [] buffer;

    // crear un registro vacío
    stringstream ss(campo);

    // leer atributos
    getline(ss, num_cheque, '|');
    getline(ss, nombre_banco, '|');
    getline(ss, cuenta_depositar, '|');
    getline(ss, monto, '|');

    // crear un objeto cheque e insertarlo a la pila de cheques
    Cheque c(stoi(num_cheque), nombre_banco, stoi(cuenta_depositar), stod(monto));
    cheques.insertar(c);
  }
  archivo.close();
}
```

```
// leer información sobre cheques rechazados del archivo físico
void leer(fstream &archivo, Pila<ChequeRechazado> &chequesRechazados) {
  int len;
  string campo, num_cheque, nombre_banco, cuenta_depositar, monto, cargo;

  // leer mientras no esté vacío el archivo
  while (!archivo.eof()) {
    if (archivo.peek() == -1)
      break;
    // leer registro
    archivo.read(reinterpret_cast<char *>(&len), sizeof(int));
    char *buffer = new char[len];
    archivo.read(buffer, len);
    campo = buffer;
    delete [] buffer;

    // crear registro vacío
    stringstream ss(campo);

    // leer atributos
    getline(ss, num_cheque, '|');
    getline(ss, nombre_banco, '|');
    getline(ss, cuenta_depositar, '|');
    getline(ss, monto, '|');
    getline(ss, cargo, '|');

    // crear objeto de cheque rechazado y agregarlo a la pila de cheques rechazados vacíos
    ChequeRechazado c(stoi(num_cheque), nombre_banco, stoi(cuenta_depositar), stod(monto), stod(cargo));
    chequesRechazados.insertar(c);
  }
  archivo.close();
}
```

```cpp
// guardar información sobre cheques en el archivo físico
void guardar(fstream &archivo, Pila<Cheque> &cheques) {
  archivo.open("cheques.dat", ios::out | ios::trunc);
  archivo.clear(); // Restaura el estado del flujo a “bueno”
  archivo.seekg(0, ios::beg); // reposicionar puntero al principio del archivo

  // Pila auxiliar para insertar el el orden correcto a la hora de leer
  Pila<Cheque> aux;
  while (!cheques.vacio()) {
    aux.insertar(cheques.arriba());
    cheques.eliminar();
  }

  int len;

  while (!aux.vacio()) {
    Cheque c = aux.arriba();
    aux.eliminar();

    string campo;

    campo += to_string(c.obtenerNumCheque()) + '|' +
    c.obtenerNombreBanco() + '|' +
    to_string(c.obtenerCuentaDepositar()) + '|' +
    to_string(c.obtenerMonto()) + '|';

    len = campo.size();
    archivo.write(reinterpret_cast<char *>(&len), sizeof(int));
    archivo.write(campo.c_str(), len);
  }
  archivo.close();
}
```

```
  // guardar información sobre cheques rechazados en el archivo físico
  void guardar(fstream &archivo, Pila<ChequeRechazado> &cheques_rechazados) {
  archivo.open("cheques_rech.dat", ios::out | ios::trunc);
  archivo.clear(); // Restaura el estado del flujo a “bueno”
  archivo.seekg(0, ios::beg); // reposicionar puntero al principio del archivo

  int len;

  // Pila auxiliar para insertar el el orden correcto a la hora de leer
  Pila<ChequeRechazado> aux;
  while (!cheques_rechazados.vacio()) {
    aux.insertar(cheques_rechazados.arriba());
    cheques_rechazados.eliminar();
  }

  while (!aux.vacio()) {
    ChequeRechazado c = aux.arriba();
    aux.eliminar();

    string campo;

    campo += to_string(c.obtenerNumCheque()) + '|' +
    c.obtenerNombreBanco() + '|' +
    to_string(c.obtenerCuentaDepositar()) + '|' +
    to_string(c.obtenerMonto()) + '|' +
    to_string(c.obtenerCargo()) + '|';

    int len = campo.size();
    archivo.write(reinterpret_cast<char *>(&len), sizeof(int));
    archivo.write(campo.c_str(), len);
  }
  archivo.close();
}

class Cheque {
  protected: // permite que la clase heredada acceda a sus atributos
    int num_cheque;
    char nombre_banco[15];
    int cuenta_depositar;
    double monto;
  public:
    Cheque() {};
    Cheque(int num_cheque, string nombre_banco, int cuenta_depositar, double monto) {
      this->num_cheque = num_cheque;
      strcpy(this->nombre_banco, nombre_banco.c_str());
      this->cuenta_depositar = cuenta_depositar;
      this->monto = monto;
    }
```

```
// imprime un cheque
void imprimir() {
  cout << left << "Número de cheque\t"
  << setw(30) << "Nombre de banco\t"
  << setw(30) << "Cuenta a depositar\t"
  << setw(30) << "Monto\t"
  << '\n';

  cout << left
  << num_cheque << setw(17) << '\t'
  << nombre_banco << setw(15) << '\t'
  << cuenta_depositar << setw(28) << '\t'
  << monto << '\n';
}
```

```
class ChequeRechazado : public Cheque { // hereda atributos de la superclase cheque
 private:
   double cargo; // único atributo que difiere de la superclase
 public:
   double obtenerCargo() {
     return this->cargo;
   }
   ChequeRechazado() {};
   ChequeRechazado(int num_cheque, string nombre_banco, int cuenta_depositar, double monto, double cargo) {
     this->num_cheque = num_cheque;
     strcpy(this->nombre_banco, nombre_banco.c_str());
     this->cuenta_depositar = cuenta_depositar;
     this->monto = monto;
     this->cargo = cargo;
   }
   void establecerCargo(double cargo) {
     this->cargo = cargo;
   }
```

```
// clase virtual para sobreescribir la función del mismo nombre de la superclase
virtual void imprimir() {
  cout << left
  << "Número de cheque\t"
  << setw(30) << "Nombre de banco\t"
  << setw(30) << "Cuenta a depositar\t"
  << setw(30) << "Monto\t"
  << setw(30) << "cargo"
  << '\n';

  cout << left << num_cheque << setw(17) << '\t'
  << nombre_banco << setw(23) << '\t'
  << cuenta_depositar << setw(28) << '\t'
  << monto << setw(25) << '\t'
  << cargo << '\n';
}
```