Федеральное государственное бюджетное образовательное учреждение высшего образования
«Саратовский государственный технический университет имени Гагарина Ю.А.»

Институт прикладных информационных технологий и коммуникаций

Кафедра Информационная безопасность автоматизированных систем

Направление подготовки 10.03.01 Информационная безопасность

Расчётно-графическая работа по дисциплине «Языки программирования»

«Бухгалтерская программа»

Выполнил: студент 1 курса
учебной группы б-ИФБС11
очной формы обучения
Блинова Ксения Игоревна
Проверил: ассистент каф. ИБС
Романчук С. П.

Саратов 2020

Аннотация

Разработать приложение, эмулирующий функционал бухгалтерской программы. Программа должна обладать следующим возможностями:

- создание расходного и приходного кассовых ордеров,
- просмотр баланса на определенную дату,
- просмотр истории операций и поиск операций по различным характеристикам,
- загрузка и сохранение истории операций в файл.

При разработке приложения обязательно должен быть реализованы классы, моделирующие приходный кассовый ордер, расходный кассовый ордер и контрагент.

Содержание

Введение

В данной расчётно-графической работе предлагается разработать бухгалтерскую программу. Цель работы – разработать удобный интерфейс для ведения учёта бухгалтерского дела с широким диапазоном действий. Задача работы – разбить создание на несколько этапов, для которых продумать соответствующий алгоритм. В работе применим все ранее полученные навыки по предмету «Языки программирования» в ходе лекционных и лабораторных занятий, а также в ходе индивидуального изучения материала.

Теоретическая часть

C++ – компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование, обобщенное программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. Синтаксис C++ унаследован от языка C. Одним из принципов разработки было сохранение совместимости с C. Тем не менее, C++ не является в строгом смысле надмножеством C. Являясь одним из самых популярных языков программирования, C++ широко используется для разработки программного обеспечения. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр).

Классы называется составной тип данных, элементами которого являются свойства и методы. Свойства – это любые данные, которыми можно характеризовать объект класса. Методы – это функции, которые могут выполнять какие-либо действия над данными(свойствами) класса. Все свойства и методы классов имеют права доступа, для этого используются модификаторы доступа public, protected и private.

Созданные объекты на основе одного класса называются экземплярами этого класса. Эти объекты могут иметь различное поведение, свойства, но все равно будут являться объектами одного класса. В ООП существует три основных принципа построения классов:

1. Инкапсуляция— это свойство, позволяющее объединить в классе и данные, и методы, работающие с ними и скрыть детали реализации от пользователя.
2. Наследование— это свойство, позволяющее создать новый класс-потомок на основе уже существующего, при этом все характеристики класса родителя присваиваются классу-потомку.
3. Полиморфизм— свойство классов, позволяющее использовать объекты классов с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Абстрактный класс – это класс, экземпляр которого нельзя создать. Абстрактные классы нужны для предоставления базового функционала для классов-наследников. Производный класс должен переопределить и реализовать все абстрактные методы, которые имеются в базовом абстрактном классе.

Конструктор класса – это специальная функция, которая выполняет начальную инициализацию элементов данных. Он не имеет типа возвращаемого значения и должен называться также, как класс, в котором он находится. Деструктор класса вызывается при уничтожении объекта. Имя деструктора аналогично имени конструктора, только в начале ставится знак тильды ~. Деструктор не имеет входных параметров.

Линейный односвязный список — это структура данных, состоящая из элементов одного типа, связанных между собой последовательно посредством указателей. Каждый элемент списка имеет указатель на следующий элемент. Последний элемент списка указывает на NULL. Элемент, на который нет указателя, является первым (головным) элементом списка. Здесь ссылка в каждом узле указывает на следующий узел в списке. В односвязном списке можно передвигаться только в сторону конца списка. Узнать адрес предыдущего элемента, опираясь на содержимое текущего узла, невозможно. Любой односвязный список базируется на специальной структуре или классе, которая описывает один узел. Действия с самим списком реализуются в дружественном классе.

В языке программирования C++ термин «Стандартная библиотека» означает коллекцию классов и функций, написанных на базовом языке. Стандартная Библиотека поддерживает несколько основных контейнеров, функций для работы с этими контейнерами, объектов-функции, основных типов строк и потоков (включая интерактивный и файловый ввод-вывод) и т.д. Функциональные особенности Стандартной Библиотеки объявляются внутри пространства имен *std*.

Стандартная библиотека шаблонов (STL) — подмножество стандартной библиотеки C++ и содержит контейнеры, алгоритмы, итераторы, объекты-функции и т. д.

Практическая часть

Класс Order – абстрактный базовый класс с виртуальной функцией print и методами get, set для каждого поля. Далее реализованы два класса-наследника от абстрактного класса Order с методами get, set для каждого поля и реализацией функции print для печати готового ордеров: InOrder, реализующий функционал для приходного кассового ордера, и OutOrder, реализующий функционал для расходного кассового ордера.

Класс Contractor является узлом для линейного односвязного списка. Класс List необходим для формирования самого односвязного списка и работы с ним. В данном классе объявляются следующие поля и методы:

- указатель head(корень списка) и переменная count типа int для подсчёта элементов списка;
- конструктор по умолчанию;
- деструктор;
- метод search, который реализует поиск в списке по желаемому параметру;
- метод clear, который полностью очищает список;
- метод show, который выводит содержимое всего списка на экран;
- методы savefile и uploadfile, которые реализуют работу с файлами, а именно сохранение списка в файл и загрузка из него;
- метод add, который помещает элемент в начало списка;
- метод empty, который проверяет наличие элементов вписке;
- методы get для каждого поля, которые возвращают соответствующие значения.

Программа включает следующие библиотеки:

- iostream – заголовочный файл с классами, функциями и переменными для организации ввода-вывода, использующий объекты cin, cout, cerr и clog для передачи информации в и из стандартных потоков ввода, вывода, ошибок (без буферизации) и ошибок (с буферизацией) соответственно. Он включён в стандартную библиотеку. Название образовано от input/outputStream («поток ввода-вывода»).
- fstream (сокращение от *«FileStream»*) — заголовочный файл из стандартной библиотеки C++, включающий набор классов, методов и функций, которые предоставляют интерфейс для чтения/записи данных из/в файл. Для манипуляции с данными файлов используются объекты, называемые потоками («stream»). Функции, включенные в данный файл, позволяют производить чтение из файлов как побайтово, так и блоками, и записывать так же. В комплект включены все необходимые функции для управления последовательностью доступа к данным файлов, а также множество вспомогательных функций.

- sstream — заголовочный файл с классами, функциями и переменными для организации работы со строками, через интерфейс потоков, в языке программирования C++. Он включён в стандартную библиотеку C++. Название образовано от сокращения имени строчного типа данных (англ. *string*) и англ. *stream* (поток). Также является частью стандартной библиотеки C++.
- stdlib.h — заголовочный файл стандартной библиотеки языка Си, который содержит в себе функции, занимающиеся выделением памяти, контроль процесса выполнения программы, преобразования типов и другие. Заголовок вполне совместим с C++. Название «stdlib» расшифровывается как «standard library» (стандартная библиотека).
- string — класс с методами и переменными для организации работы со строками в языке программирования C++. Он включён в стандартную библиотеку C++. Название образовано от имени строчного типа данных (англ. *string*; с англ. — «строка»).

В функции main реализовано меню с выбором действий через оператора switch и цикл do while, которое выполняются пока пользователь не захочет выйти из приложения. В качестве истории операции используется список, содержащий последовательность информации.

Заключение

Цель и задачи расчетно-графической работы выполнены. В процессе выполнения расчетно-графической работы был создан интерфейс бухгалтерской программы с различным спектром действий. В ходе работы я глубже ознакомилась с теоретической составляющей языка C++ и на практике отработала навык составления логики программы и нахождения разных способов реализации алгоритма.

Приложения

Order.h

```
#include <iostream>
#include <string>
using namespace std;
class Order
{
   string name, base;
   int docnumber, money;
   int day, month, year;
protected:
   Order(const string n, const string b, const int d, const int m, const int y, const int dc, const int s);
public:
   Order();
   void setName(const string n);
   void setBase(const string b);
   bool setDay(const int d);
   bool setMonth(const int m);
   bool setYear(const int y);
   bool setDocnumber(const int dc);
   bool setMoney(const int s);
   string getName() const;
   string getBase() const;
   int getDay() const;
   int getMonth() const;
   int getYear() const;
   int getDocnumber() const;
   int getMoney() const;
   virtual void print (string const o) = 0;
};
```

Order.cpp

```
#include "Order.h"
Order::Order(){};
Order::Order(const string n, const string b, const int d, const int m, const int y, const int dc, const int
s)
{
   setName(n);
   setBase(b);
   setDay(d);
   setMonth(m);
   setYear(y);
   setDocnumber(dc);
   setMoney(s);
}
void Order::setName(const string n) { this->name = n; }
void Order::setBase(const string b) { this->base = b; }
bool Order::setDay(const int d)
{
   if((d > 31)&&(d <= 0))
   {
      return false;
   }
   this->day = d;
```

```
    return true;
}
bool Order::setMonth(const int m)
{
    if((m > 12)&&(m <= 0))
    {
        return false;
    }
    this->month = m;
    return true;
}
bool Order::setYear(const int y)
{
    if((y > 2020)&&(y < 1980))
    {
        return false;
    }
    this->year = y;
    return true;
}
bool Order::setDocnumber(const int dc)
{
    if(dc <= 0)
    {
        return false;
    }
    this->docnumber = dc;
    return true;
}
bool Order::setMoney(const int s)
{
    if(s <= 0)
    {
        return false;
    }
    this->money = s;
    return true;
}
string Order::getName() const { return name; }
string Order::getBase() const { return base; }
int Order::getDay() const { return day; }
int Order::getMonth() const { return month; }
int Order::getYear() const { return year; }
int Order::getDocnumber() const { return docnumber; }
int Order::getMoney() const { return money; }
```

InOrder.h

```
#include "Order.h"
class InOrder : public Order
{
    float vat;
public:
    InOrder();
```

```
    InOrder(const string n, const string b, const int d, const int m, const int y, const int dc, const int s, const float v);
    bool setVat(const float v);
    float getVat() const;
    void print (string const o) override;
};
```

InOrder.cpp

```
#include "InOrder.h"
InOrder::InOrder(){};
InOrder::InOrder(const string n, const string b, const int d, const int m, const int y, const int dc, const int s, const float v) : Order(n,b,d,m,y,dc,s)
{
    setVat(v);
}
bool InOrder::setVat(const float v)
{
    if(v < 0)
    {
        return false;
    }
    this->vat = v;
    return true;
}
float InOrder::getVat() const { return vat; }
void InOrder::print (string const o)
{
    cout << endl;
    cout << "_____" << endl;
    cout << "          INCOMING CASH WARRANT" << endl;
    cout << "Organization: " << o << endl;
    cout << "|" << getDocnumber() << "|" << getDay() << "." << getMonth() << "." << getYear() << endl;
    cout << "Received from " << getName() << endl;
    cout << "Base: " << getBase() << endl;
    cout << "Including ";
    if (getVat() == 0)
    {
        cout << "without VAT" << endl;
    }
    else cout << getVat() << endl;
    cout << "Sum: " << getMoney() << endl;
    cout << "_____" << endl;
    cout << endl;
}
```

OutOrder.h

```
#include "Order.h"
class OutOrder : public Order
{
    string info;
public:
    OutOrder();
    OutOrder(const string n, const string b, const int d, const int m, const int y, const int dc, const int s, const string i);
```

```
    void setInfo(const string i);
    string getInfo() const;
    void print() override;
};
```

OutOrder.cpp

```
#include "OutOrder.h"
OutOrder::OutOrder(){};
OutOrder::OutOrder(const string n, const string b, const int d, const int m, const int y, const int dc, const int s, const string i) : Order(n,b,d,m,y,dc,s)
{
    setInfo(i);
}
void OutOrder::setInfo(const string i) { this->info = i; }
string OutOrder::getInfo() const { return info; }
void OutOrder::print()
{
    cout << endl;
    cout << "_____" << endl;
    cout << "         OUTCOMING CASH WARRANT" << endl;
    cout << "Organization: " << o << endl;
    cout << "|" << getDocnumber() << "|" << getDay() << "." << getMonth() << "." << getYear() << endl;
    cout << "Received from " << getName() << endl;
    cout << "Base: " << getBase() << endl;
    cout << "Passport details: " << getInfo() << endl;
    cout << "Sum: " << getMoney() << endl;
    cout << "_____" << endl;
    cout << endl;
}
```

List.h

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <stdlib.h>
using namespace std;
class Contractor
{
    string name;
    int day, month, year, money;
    Contractor *next;
    friend class List;
};
class List
{
    Contractor *head;
    int count = 0;
public:
    List();
    ~List();
    void search(int num);
    void clear();
    void show();
```

```
    void savefile();
    void uploadfile();
    Contractor* add(string n, int d, int m, int y, int s);
    bool empty();
    string getName(Contractor *temp) const;
    int getDay(Contractor *temp) const;
    int getMonth(Contractor *temp) const;
    int getYear(Contractor *temp) const;
    int getMoney(Contractor *temp) const;
};
```

List.cpp

```
#include "List.h"
List::List()
{
    head = 0;
    count = 0;
}
List::~List()
{
    clear();
}
void List::search(int num)
{
    Contractor *temp = head;
    int i = 0;
    string line;
    cout << "Search: ";
    getline(cin, line);
    while (temp != 0)
    {
        if (num == 1)
        {
            if (temp->month == atoi(line.c_str()))
            {
                cout << getDay(temp) << "." << getMonth(temp) << "." << getYear(temp) << " '" << getName(temp) << "' " << getMoney(temp) << endl;
                i++;
            }
        }
        if (num == 2)
        {
            if (temp->year == atoi(line.c_str()))
            {
                cout << getDay(temp) << "." << getMonth(temp) << "." << getYear(temp) << " '" << getName(temp) << "' " << getMoney(temp) << endl;
                i++;
            }
        }
        if (num == 3)
        {
            if (temp->name == line)
            {
```

```
                cout << getDay(temp) << "." << getMonth(temp) << "." << getYear(temp) << " '" << getName(temp) << "' " << getMoney(temp) << endl;
                i++;
            }
        }
        if (num == 3)
        {
            if (temp->money == atoi(line.c_str()))
            {
                cout << getDay(temp) << "." << getMonth(temp) << "." << getYear(temp) << " '" << getName(temp) << "' " << getMoney(temp) << endl;
                i++;
            }
        }
        temp = temp->next;
    }
    if (i > 0)
    {
        cout << "Total found: " << i << endl;
    }
    else cout << "<No such parameter found>" << endl;
}
void List::clear()
{
    if (empty())
    {
        cout << "The list is empty" << endl;
        return;
    }
    Contractor *temp = head;
    Contractor *p = 0;
    while (temp != 0)
    {
        p = temp;
        temp = temp->next;
        delete p;
    }
    count = 0;
    head = 0;
}
void List::show()
{
    if (empty())
    {
        cout << "The list is empty" << endl;
        return;
    }
    Contractor *temp = head;
    cout << "<LIST>" << endl;
    while (temp != 0)
    {
        cout << getDay(temp) << "." << getMonth(temp) << "." << getYear(temp) << " '" << getName(temp) << "' " << getMoney(temp) << endl;
```

```
            temp = temp->next;
        }
        cout << "Total items: " << count << endl;
        cout << endl;
    }
    void List::savefile()
    {
        fstream file;
        file.open("file.txt", ios_base::out);
        if (!file.is_open())
        {
            cout << "ERROR: file is not open" << endl;
            return;
        }
        Contractor *temp = head;
        while (temp != 0)
        {
            file << getName(temp) << "\t" << getDay(temp) << "\t" << getMonth(temp) << "\t" << getYear(temp) << "\t"  << getMoney(temp) << endl;
            temp = temp->next;
        }
        cout << "<File saved>" << endl;
        file.close();
    }
    void List::uploadfile()
    {
        ifstream file;
        file.open("f.txt");
        if (!file.is_open())
        {
            cout << "ERROR: file is not open" << endl;
            return;
        }
        string line;
        while (getline(file, line))
        {
            string n, token;
            int d, m ,y, s, i = 0;
            istringstream flow(line);
            while (getline(flow, token, '\t'))
            {
                if (i == 0) n = token;
                if (i == 1) d = atoi(token.c_str());
                if (i == 2) m = atoi(token.c_str());
                if (i == 3) y = atoi(token.c_str());
                if (i == 4) s = atoi(token.c_str());
                i++;
            }
            add(n, d, m, y, s);
        }
    }
    Contractor* List::add(string n, int d, int m, int y, int s)
    {
```

```
    Contractor *temp = new Contractor;
    temp->name = n;
    temp->day = d;
    temp->month = m;
    temp->year = y;
    temp->money = s;
    if (empty())
    {
        temp->next = 0;
        head = temp;
    }
    else
    {
        temp->next = head;
        head = temp;
    }
    count++;
    return temp;
}
bool List::empty() { return head == 0; }
string List::getName(Contractor *temp) const { return temp->name; }
int List::getDay(Contractor *temp) const { return temp->day; }
int List::getMonth(Contractor *temp) const { return temp->month; }
int List::getYear(Contractor *temp) const { return temp->year; }
int List::getMoney(Contractor *temp) const { return temp->money; }
```

main.cpp

```
#include "Order.h"
#include "InOrder.h"
#include "OutOrder.h"
#include "List.h"
void menu();
int getNumber(int total);
int main()
{
    List list;
    int num;
    string n, b, i;
    int d, m, y, dc, s;
    float v;
    //Information about the organization
    string organization = "OOO Dallas";
    int balance = 500000;
    do{
        menu();
        num = getNumber(6);
        switch(num)
        {
            case 1:
                do{
                    cout << "1. INCOMING CASH WARRANT" << endl;
                    cout << "2. OUTCOMING CASH WARRANT" << endl;
                    num = getNumber(2);
                    switch(num)
```

```
        {
            case 1:{
                cout << "<Fill in the following fields>" << endl;
                cout << "Name: ";
                getline(cin, n);
                cout << "Base: ";
                getline(cin, b);
                cout << "Sum: ";
                cin >> s;
                cout << "Date(enter a space between day, month, year): ";
                cin >> d >> m >> y;
                cout << "Document Number: ";
                cin >> dc;
                cout << "VAT (if not, enter 0): ";
                cin >> v;
                InOrder cell(n, b, d, m, y, dc, s, v);
                cell.print(organization);
                list.add(n, d, m, y, s);
                balance = balance + s;
                break;
            }
            case 2:
                cout << "<Fill in the following fields>" << endl;
                cout << "Name: ";
                getline(cin, n);
                cout << "Base: ";
                getline(cin, b);
                cout << "Sum: ";
                cin >> s;
                cout << "Date(enter a space between day, month, year): ";
                cin >> d >> m >> y;
                cout << "Document Number: ";
                cin >> dc;
                cout << "Passport details: ";
                cin >> i;
                OutOrder cell(n, b, d, m, y, dc, s, i);
                cell.print(organization);
                list.add(n, d, m, y, s);
                balance = balance - s;
                break;
        }
        cout << "1. Create another cash order" << endl;
        cout << "2. Return to the menu" << endl;
        num = getNumber(2);
    } while(num != 2);
    break;
case 2:
    list.show();
    if (list.empty() == 0)
    {
        cout << "<Do you want to save the operation history to a file?>" << endl;
        cout << "1. Yes" << endl;
        cout << "2. No, return to the menu" << endl;
```

```
                num = getNumber(2);
                if (num == 1)
                {
                    list.savefile();
                }
            }
            break;
        case 3:
            cout << "Balance on today: " << balance << endl;
            break;
        case 4:
            cout << "<Search by>" << endl;
            cout << "1. month" << endl;
            cout << "2. year" << endl;
            cout << "3. name" << endl;
            cout << "4. amount of money" << endl;
            num = getNumber(4);
            list.search(num);
            break;
        case 5:
            list.uploadfile();
            break;
      }
   } while (num != 6);
   return 0;
}
void menu()
{
   cout << endl;
   cout << "<ACCOUNTING>" << endl;
   cout << "1. Create cash order" << endl;
   cout << "2. View all operations history" << endl;
   cout << "3. View balance" << endl;
   cout << "4. Search operations by parameters" << endl;
   cout << "5. Download history from file" << endl;
   cout << "6. Exit" << endl;
}
int getNumber(int total)
{
   int number;
   string str;
   cout << ">> ";
   getline(cin, str);
   while ((sscanf(str.c_str(),"%i",&number) != 1)||(number < 1)||(number > total))
   {
      cout << "ERROR, try again >> ";
      getline(cin, str);
   }
   cout << endl;
   return number;
}
```

Литература

1. https://ru.wikipedia.org
2. http://www.cplusplus.com