Лабораторная работа № 10

Цель работы

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Разработка программы, в которой данные сохраняются в файле, корректируются и выводятся из файла на печать. Работа с файлом осуществляется с использованием потоковых классов

Постановка задачи

1. Создать пользовательский класс с минимальной функциональностью.
2. Написать функцию для создания объектов пользовательского класса (ввод исходной информации с клавиатуры) и сохранения их в потоке (файле).
3. Написать функцию для чтения и просмотра объектов из потока.
4. Написать функцию для удаления объектов из потока в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.
5. Написать функцию для добавления объектов в поток в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.
6. Написать функцию для изменения объектов в потоке в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.
7. Для вызова функций в основной программе предусмотреть меню.

Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа int для копеек. Дробная часть числа при выводе на экран должна быть отделена от целой части запятой. Реализовать:

- вычитание дробного числа из суммы
- операции сравнения (==, !=).

Задание:

- Удалить все записи равные заданному значению.
- Уменьшить все записи с заданным значением на 1 рубль 50 копеек. Значение интервала не должно быть меньше 0 рублей 0 копеек.
- Добавить K записей после элемента с заданным значением.

Описание класса

Класс Money представляет денежную сумму с рублями (тип long) и копейками (тип int). Основные характеристики:

- Хранение денежной суммы в двух полях: rub (рубли) и cheers (копейки)
- Конструкторы: по умолчанию, с параметрами, копирования
- Геттеры и сеттеры для полей класса
- Перегруженные операторы:
    - Присваивания (=)
    - Сравнения (==, !=)

- Вычитания (-)
- Ввода/вывода (>>, <<)

Определение компонентных функций

Основные методы класса Money:

- Конструкторы:
  - Money() - инициализация нулевыми значениями
  - Money(long r, int c) - инициализация заданными значениями
  - Money(const Money &m) - конструктор копирования
- Операторы:
  - operator= - присваивание значений другого объекта Money
  - operator== и operator!= - сравнение денежных сумм
  - operator- - вычитание денежных сумм
  - operator>> и operator<< - ввод/вывод в консоль
  - Дружественные операторы для работы с файловыми потоками

Определение глобальных функций

Функции для работы с файлами:

- make_file() - создание файла и запись в него объектов Money
- print_file() - чтение и вывод содержимого файла
- delete_record() - удаление записи по номеру
- add_record() - добавление записи после указанной позиции
- add_end() - добавление записи в конец файла
- change_record() - изменение указанной записи

Функция main()

```
int main() {
    Money m;
    int k, c;
    char filename[30];
    do {
        cout << "\n0. Exit";
        cout << "\n1. Make file";
        cout << "\n2. Print file";
        cout << "\n3. Delete record from file";
        cout << "\n4. Add record to file";
        cout << "\n5. Change value of record";
        cin >> c;
        switch (c) {
            case 0:
                return 0;
            case 1:
                cout << "\nEnter file name: ";
```

```
                cin >> filename;
                k = make_file(filename);
                break;
            case 2:
                cout << "\nEnter file name: ";
                cin >> filename;
                k = print_file(filename); // call function print_file
                if (k == 0) cout << "\nFile emply"; // if file is empty
                if (k < 0) cout << "\nCant read file"; // if cant open file
                break;
            case 3:
                cout << "\nEnter file name: ";
                cin >> filename;
                cout << "\nEnter value: ";
                cin >> k;
                k = delete_record(filename, k);
                if (k < 0) cout << "\nCant read file";
                break;
            case 4:
                cout << "\nEnter file name: ";
                cin >> filename;
                cout << "\nEnter value: ";
                cin >> k;
                cout << "\nEnter Money: ";
                cin >> m;
                k = add_record(filename, k, m);
                if (k < 0) cout << "\nCant read file";
                if(k==0) k = add_end(filename, m);
                break;
            case 5:
                cout << "\nEnter file name: ";
                cin >> filename;
                cout << "\nEnter value: ";
                cin >> k;
                cout << "\nEnter Money: ";
                cin >> m;
                k = change_record(filename, k, m);
                if (k < 0) cout << "\nCant read file";
                if (k == 0) cout << "\nRecord not found";
                break;
            default:
                cout << "\nWrong choice";
                break;
        }
    } while (c != 0);
    return 0;
}
```

Объяснение результатов работы программы

Программа предоставляет меню для работы с файлами:

1. Создание файла с записями Money
2. Просмотр содержимого файла
3. Удаление записи по номеру
4. Добавление записи после указанной позиции
5. Изменение указанной записи Все операции работают с файлами через потоковые классы C++, обеспечивая сохранение данных между запусками программы.

Ответы на контрольные вопросы

1. Что такое поток?

Ответ: Поток - это абстракция для последовательного ввода/вывода данных, представляющая собой последовательность байтов.

2. Какие типы потоков существуют?

Ответ:

- Входные (istream)
- Выходные (ostream)
- Файловые (ifstream, ofstream, fstream)
- Строковые (istringstream, ostringstream, stringstream)
- Стандартные (cin, cout, cerr, clog)

3. Какую библиотеку надо подключить при использовании стандартных потоков?

Ответ:

4. Какую библиотеку надо подключить при использовании файловых потоков?

Ответ:

5. Какую библиотеку надо подключить при использовании строковых потоков?

Ответ:

6. Какая операция используется при выводе в форматированный поток?

Ответ: Оператор << (оператор вставки)

7. Какая операция используется при вводе из форматированных потоков?

Ответ: Оператор >> (оператор извлечения)

8. Какие методы используются при выводе в форматированный поток?

Ответ:

- setw() - ширина поля
- setprecision() - точность
- setfill() - символ заполнения

- fixed - фиксированная точка
- scientific - научная нотация

9. Какие методы используется при вводе из форматированного потока?

Ответ:

- get() - чтение символа
- getline() - чтение строки
- ignore() - пропуск символов
- read() - чтение блока данных

10. Какие режимы для открытия файловых потоков существуют?

Ответ:

- ios::in - чтение
- ios::out - запись
- ios::app - добавление в конец
- ios::ate - открытие с перемещением в конец
- ios::trunc - очистка файла
- ios::binary - бинарный режим

11. Какой режим используется для добавления записей в файл?

Ответ: ios::app или ios::ate

12. Какой режим (комбинация режимов) используется в конструкторе ifstream file(“f.txt”)?

Ответ: ios::in

13. Какой режим (комбинация режимов) используется в конструкторе fstream file(“f.txt”)?

Ответ: По умолчанию ios::in | ios::out

14. Какой режим (комбинация режимов) используется в конструкторе ofstream file(“f.txt”)?

Ответ: ios::out (неявно включает ios::trunc)

15. Каким образом открывается поток в режиме ios::out|ios::app?

Ответ: Для записи с добавлением в конец файла

16. Каким образом открывается поток в режиме ios::out |ios::trunc?

Ответ: Для записи с очисткой файла

17. Каким образом открывается поток в режиме ios::out |ios::in|ios::trunk?

Ответ: Для чтения и записи с очисткой файла

18. Каким образом можно открыть файл для чтения?

Ответ:

```
ifstream file("filename.txt", ios::in);
// или просто
ifstream file("filename.txt");
```

19. Каким образом можно открыть файл для записи?

Ответ:

```
ofstream file("filename.txt", ios::out);
// или просто
ofstream file("filename.txt");
```

20. Привести примеры открытия файловых потоков в различных режимах.

Ответ:

```
// Чтение
ifstream in("input.txt");

// Запись с очисткой
ofstream out("output.txt", ios::out | ios::trunc);

// Чтение и запись
fstream io("data.txt", ios::in | ios::out);

// Добавление в конец
ofstream append("log.txt", ios::app);
```

21. Привести примеры чтения объектов из потока.

Ответ:

```
ifstream file("data.txt");
Money m;
while (file >> m) {
    cout << m << endl;
}
```

22. Привести примеры записи объектов в поток.

Ответ:

```
ofstream file("data.txt");
Money m(100, 50);
file << m;
```

23. Сформулировать алгоритм удаления записей из файла.

Ответ:

1. Открыть исходный файл для чтения
2. Создать временный файл для записи
3. Копировать записи из исходного файла во временный, пропуская удаляемые
4. Закрыть оба файла
5. Удалить исходный файл
6. Переименовать временный файл

24. Сформулировать алгоритм добавления записей в файл.

Ответ:

1. Открыть исходный файл для чтения
2. Создать временный файл для записи
3. Копировать записи до позиции вставки
4. Добавить новые записи
5. Копировать оставшиеся записи
6. Закрыть оба файла
7. Удалить исходный файл
8. Переименовать временный файл

25. Сформулировать алгоритм изменения записей в файле

Ответ:

1. Открыть исходный файл для чтения
2. Создать временный файл для записи
3. Копировать записи до изменяемой
4. Записать измененную версию записи
5. Копировать оставшиеся записи
6. Закрыть оба файла
7. Удалить исходный файл
8. Переименовать временный файл