Лабораторная работа №4. Стандартные потоки

Потоковые классы

Программа на C++ представляет ввод и вывод как поток байтов. При вводе она читает байты из потока ввода, при выводе вставляет байты в поток вывода. Понятие потока позволяет абстрагироваться от того, с каким устройством ввода-вывода идет работа. Например, байты потока ввода могут поступать либо с клавиатуры, либо из файла на диске, либо из другой программы. Аналогично, байты потока вывода могут выводиться на экран, в файл на диске или на вход другой программы.

Обычно ввод-вывод осуществляется через *буфер* — область оперативной памяти, накапливающую какое-то количество байтов, прежде чем они пересылаются по назначению. При обмене, например, с диском это значительно повышает скорость передачи информации. При вводе с клавиатуры буферизация обеспечивает другое удобство для пользователя — возможность исправления ошибок набора символов, пока они не отправлены из буфера в программу.

Буфер ввода обычно очищается при нажатии клавиши Enter. *Буфер вывода* на экран очищается либо при появлении в выходном потоке символа новой строки `'\n'`, либо при выполнении программой очередной операции ввода с клавиатуры.

Язык C++ предоставляет возможности ввода-вывода на низком уровне (*неформатированный* ввод-вывод) и на высоком уровне (*форматированный* ввод-вывод). В первом случае передача информации выполняется блоками байтов без какого-либо преобразования данных. Во втором случае байты группируются для представления таких элементов данных, как целые и вещественные числа, строки символов и так далее.

Для поддержки потоков библиотека C++ содержит иерархию классов, построенную на основе двух базовых классов — `ios` и `streambuf`. Класс `ios` содержит базовые средства управления потоками, являясь родительским для других классов `вводавывода`. Класс `streambuf` обеспечивает общие средства управления буферами потоков и их взаимодействие с физическими устройствами, являясь родительским для других буферных классов. Связь между этими двумя классами реализует поле `bp` в классе `ios`, являющееся указателем на `streambuf`.

Классы стандартных потоков

Потоки, связанные с консольным вводом-выводом (клавиатура–экран), называются *стандартными*. Стандартному потоку ввода соответствует класс

istream, стандартному потоку вывода — класс ostream. Оба класса являются потомками класса ios. Следующим в иерархии является класс iostream, наследующий классы istream и ostream и обеспечивающий общие средства потокового ввода-вывода.

Заголовочные файлы библиотеки ввода-вывода C++

Заголовочные файлы стандартной библиотеки C++ по стандарту указываются без расширения, например <iostream>. Все имена стандартной библиотеки принадлежат пространству имен std. Для упрощения доступа к именам в программах часто используется директива using namespace std;.

Объекты и методы стандартных потоков ввода-вывода

По директиве #include <iostream> становятся доступными объекты:

- cin — объект класса istream, соответствующий стандартному потоку ввода;
- cout — объект класса ostream, соответствующий стандартному потоку вывода.

Оба потока являются буферизованными. Благодаря объектам cin и cout становятся доступными и методы соответствующих классов.

Форматированный ввод-вывод реализуется через две перегруженные операции: операция сдвига влево '<<' перегружена для вывода в поток и называется *операцией вставки* (*вывода*) в поток; операция сдвига вправо '>>' перегружена для ввода из потока и называется *операцией извлечения* (*ввода*) из потока. Например, в классе ostream определены операции

```
ostream& operator<<( short );
ostream& operator<<( int );
ostream& operator<<( double );          // и так далее
```

Рассмотрим, что произойдет, если в программе встретится оператор

```
cout << x;                          // ранее в программе: int x = 5;
```

Сначала компилятор определит, что левый аргумент имеет тип ostream, а правый — тип int. Далее он найдет в заголовочном файле ostream прототип метода ostream& operator<<(int). В конечном итоге будет вызван метод cout.operator<<(x), в результате выполнения которого мы увидим на экране 5.

Заметим, что в случае вывода значений встроенных типов, таких как int или double, класс ostream обеспечивает их преобразование из внутреннего

двоичного представления в строку символов. О форматировании можно не заботиться, так как класс `ostream` поддерживает форматирование вывода по умолчанию, задаваемое соответствующими полями базового класса `ios`.

Аналогично происходит работа со стандартным потоком ввода через объект `cin`, но направление движения информации противоположное: из потока — в программу.

Если форматирование по умолчанию вас не устраивает, можно воспользоваться либо соответствующими *методами* класса `ios`, либо *манипуляторами* ввода-вывода,представляющими собой функции, которые можно включать прямо в поток. Они определены частично в файлах `istream` и `ostream` и частично в файле `iomanip`. В табл. 4.1 приведены наиболее часто употребляемые манипуляторы[1].

Таблица 4.1. Основные манипуляторы и методы управления форматом

Манипулятор	Метод класса ios	Ввод	Вывод	Описание
endl	—		+	Включить в поток символ новой строки и выгрузить буфер
dec	flags(ios::dec)	+	+	Перейти в десятичную систему счисления
hex	flags(ios::hex)	+	+	Перейти в шестнадцатеричную систему счисления
oct	flags(ios::oct)	+	+	Перейти в восьмеричную систему счисления
setprecision(n)	precision(n)		+	Установить количество отображаемых знаков
setw(n)	width(n)		+	Установить ширину поля вывода
setfill(c)	fill(c)		+	Установить символ заполнения с

Пример вывода целой переменной `x` в шестнадцатеричной форме:

```
cout.flags( ios::hex ); cout << x; // способ 1: использование метода
cout << hex << x; // способ 2: использование манипулятора
```

Остановимся немного подробней на манипуляторе `setprecision(n)` и соответствующем методе `precision(n)`, поскольку их воздействие на формат вывода подчиняется довольно сложным правилам. И тот, и другой изменяют значение поля `ios::x_precision` (по умолчанию это значение равно шести). Данное поле управляет точностью вывода вещественных чисел, причем его интерпретация зависит от значений других полей, управляющих

[1] Полный список манипуляторов и методов форматирования см. в Учебнике, с. 271.

форматом вывода: ‰ `ios::scientific` _ «научный» формат (с плавающей точкой); ‰ `ios::fixed` _ формат вывода с фиксированной точкой.

Если установлен хотя бы один из этих форматов (например, с помощью метода `flags`), то `x_precision` задает количество цифр *после десятичной точки*. Если не установлен ни тот, ни другой и вывод идет в так называемом *автоматическом формате*, то значение `x_precision` задает *общее количество значащих цифр*.

Учтите, что большинство параметров форматирования сохраняют свое состояние вплоть до следующего вызова функции, изменяющей это состояние. Исключение — манипулятор `setw(n)` и соответствующий ему метод `width(n)`: их действие распространяется только на ближайшую операцию вывода, после чего восстанавливается ширина поля вывода по умолчанию.

По признаку наличия аргумента манипуляторы подразделяются на *простые* (без аргумента) и *параметризованные* (с аргументом). Для использования последних необходимо подключить заголовочный файл `iomanip`.

С точки зрения *реализации* манипуляторы можно разделить на три группы:

1) манипуляторы без аргумента, выводящие в поток управляющий символ (`endl`, `ends`) или очищающие буфер потока (`flush`);

2) манипуляторы без аргумента, изменяющие значения полей базового класса `ios`, задающих текущую систему счисления (`dec`, `hex`, `oct`);

3) манипуляторы с аргументом.

Кроме рассмотренных, класс `ios` предоставляет и другие методы, обеспечивающие неформатированный ввод-вывод, а также связь с буфером потока. Наиболее часто употребляемые методы ввода-вывода приведены в табл. 4.2.

Таблица 4.2. Некоторые методы ввода-вывода класса ios

Метод	Описание
get()	Возвращает код извлеченного из потока символа или EOF, если достигнут конец файла
get(c)	Присваивает код извлеченного из потока символа аргументу c
get(buf, num, lim='\n')	Читает из потока символы, пока не встретится символ lim (по умолчанию '\n') или пока не будет прочитано num–1 символов. Извлеченные символы размещаются в символьный массив buf, к ним добавляется нулевой байт. Символ lim остается в потоке
getline(buf, num, lim='\n')	Выполняется аналогично предыдущему методу, но символ lim удаляется из потока (в массив buf он также не записывается)

peek()	Возвращает код следующего (готового для извлечения) символа в потоке, но не извлекает его; или EOF, если достигнут конец файла
read(buf, num)	Считывает num символов из потока в символьный массив buf
gcount()	Возвращает количество символов, прочитанных последним вызовом функции неформатированного ввода
rdbuf()	Возвращает указатель на буфер типа streambuf, связанный с данным потоком
put(c)	Выводит в поток символ c
write(buf, num)	Выводит в поток num символов из массива buf

Обработка ошибок потоков

Библиотека ввода-вывода C++ обеспечивает гораздо более надежный ввод-вывод, чем старые функции библиотеки C. Это достигается перегрузкой операций извлечения и вставки для всех встроенных типов, что исключает путаницу с типами, которая была возможна при использовании функций `scanf` и `printf` и которая приносила массу неприятностей программистам, поскольку компилятор никак не реагировал на ошибки в спецификации формата.

И все же проблема ошибок практически не волнует нас только при выводе информации в поток `cout`. При вводе никто не может запретить пользователю ввести вместо ожидаемого программой целого числа произвольную строку символов, и если не принять специальных мер, программа «сломается». Для отслеживания ошибок потоков в базовом классе `ios` определено поле `state`, отдельные биты (флаги) которого отображают состояние потока, как показано в табл. 4.3.

Таблица 4.3. Флаги состояния потока

Имя флага	Интерпретация
ios::goodbit	Нет ошибок
ios::eofbit	Достигнут конец файла
ios::failbit	Ошибка форматирования или преобразования
ios::badbit	Серьезная ошибка, после которой пользоваться потоком невозможно

Получить текущее состояние потока можно с помощью метода `rdstate`, который возвращает значение типа `int`. Есть и другие, более удобные для анализа методы:

- `int eof()` — возвращает ненулевое значение, если установлен флаг `eofbit;` % `int fail()` — возвращает ненулевое значение, если случилась

любая ошибка ввода-вывода (но не конец файла). Этому условию соответствует установка либо флага `badbit`, либо флага `failbit`;

- `int bad()` — возвращает ненулевое значение, если случилась серьезная ошибка ввода-вывода (то есть установлен флаг `badbit`);
- `int good()` — возвращает ненулевое значение, если сброшены все флаги ошибок. Если произошла ошибка ввода, в результате которой установлен только флаг `failbit`, то после этого можно продолжать работу с потоком, но сначала нужно сбросить все флаги ошибок, для чего предназначен метод `void clear(int = 0)`. Для сброса флагов достаточно сделать вызов `clear()`, так как аргумент по умолчанию равен нулю. Этот же метод можно использовать для установки соответствующих флагов поля `state`, если передать ему ненулевой аргумент (обычно это комбинация флагов из табл. 4.3, объединенных операцией `|`).

Есть и другие приемы диагностирования ошибочных ситуаций. Мы используем их при решении задачи 4.1.

Перегрузка операций извлечения и вставки для типов, определенных программистом

Одно из удобств C++ — возможность перегрузить операции извлечения и вставки для любого класса `MyClass`, созданного программистом. После этого для любого объекта obj класса `MyClass` можно записывать операторы вида `cin >> obj; cout << obj`. Удобно, не правда ли?

Однако операции >> и << не могут быть элементами класса `MyClass`. Причина в том, что у любой функции-операции класса левым операндом подразумевается объект этого класса, вызывающий данную функцию. Но для операций извлечения (вставки) левый операнд должен быть потоком ввода-вывода. Поэтому эти функции всегда объявляется *дружественными* классу, для которого они создаются.

Общая форма пользовательской *функции извлечения*: `istream& operator >>( istream& is, MyClass& obj )` где `is` _ ссылка на входной поток, `obj` — ссылка на объект, принимающий ввод. В теле функции последний оператор должен возвращать объект `is`. Общая форма пользовательской *функции вставки*: `ostream& operator <<( ostream& os, MyClass& obj )`

где `os` — ссылка на выходной поток, а `obj` — ссылка на объект, к которому применяется операция. Последний оператор функции должен

возвращать объект os. В принципе, возможна и другая сигнатура операции вставки с передачей второго аргумента по значению (MyClass obj). Но в этом случае, во-первых, при вызове функции в стеке будет создаваться копия объекта obj, а во-вторых, в классе MyClass должен быть предусмотрен конструктор копирования.

Рассмотрим теперь задачу, в которой используются стандартные средства вводавывода для потоков и, кроме этого, перегружаются операции извлечения и вставки для пользовательского типа данных.

Задача 4.1. Первичный ввод и поиск информации в базе данных

Написать программу, которая обеспечивает первичный ввод информации в базу данных отдела кадров предприятия численностью до 100 человек (без записи в файл) и поиск информации по заданному критерию. Каждая запись базы данных содержит следующие сведения о сотруднике: фамилия и инициалы; год поступления на работу; оклад. Критерий поиска: сотрудники с окладом, превышающим некоторую заданную величину.

Решение задачи начнем с выявления понятий (классов) и их фундаментальных взаимосвязей.

В данном случае первым понятием является «*база данных*», и, следовательно, для моделирования этого понятия понадобится класс, который мы назовем DBase. Объект типа DBase должен содержать некоторую совокупность, или коллекцию, других объектов, соответствующих записям базы данных. Для моделирования понятия «*запись базы данных*» введем класс Man. Очевидно, что взаимоотношение между указанными классами относится к типу «DBase *has a* Man».

На втором этапе необходимо уточнить классы, определив основные поля и набор операций над ними.

Начнем с класса DBase. Первый вопрос: какую структуру данных целесообразно использовать для хранения коллекции записей. Поскольку объем базы данных небольшой, выберем самое простое решение — массив объектов типа Man. Очевидно, что в конструкторе класса необходимо выполнить выделение памяти для требуемого количества объектов типа Man, а в деструкторе — освобождение этой памяти. Адрес начала массива объектов представим полем Man* pMan.

Ясно также, что в классе необходимо иметь метод InitInput для первичного ввода информации в базу данных и метод SearchPayNotLess для поиска сотрудников с окладом, превышающим некоторую заданную

величину. Для контроля правильности ввода исходных данных нам пригодится еще один метод — Show , выполняющий вывод на экран содержимого базы данных.

Теперь разберемся с классом Man. Для хранения информации об одном сотруднике потребуются следующие поля:

- char* pName — адрес строки, содержащей фамилию и инициалы;
- int take_job_year — год поступления на работу; ‰ double pay — величина оклада.

Конструктор класса должен выделять память для хранения указанной строки, а деструктор — освобождать эту память. Для решения второй подзадачи (поиск информации) добавим в класс метод доступа GetPay. И наконец, для класса Man нужно предусмотреть перегрузку операции извлечения, чтобы обеспечить первичный ввод информации с клавиатуры в методе InitInput класса DBase, и операцию вставки, которая будет использована в методе Show класса DBase. Обе операции должны быть реализованы как внешние дружественные функции.

Иногда при решении задачи удобно использовать внешние функции, не являющиеся элементами классов. Обычно они выполняют какую-то рутинную работу и могут быть вызваны как из методов классов, так и из основной функции.

Типичный пример — ввод значений из стандартного потока cin с защитой от непреднамеренных ошибок пользователя. Начнем с «наивной» реализации перегруженной операции >> для класса Man:

```
istream& operator >> ( istream& in, Man& obj ) {
// . . . . . .
    in >> obj.take_job_year;     // 1
    in >> obj.pay;
return in;
}
```

Если при выполнении *оператора 1* пользователь введет вместо целого числа строку символов, программа «сломается». Вашему заказчику наверняка это не понравится. Такая же ситуация возможна и при вводе вещественного числа в следующем операторе. Для решения этих проблем в программе будут использованы функции GetInt и GetDouble, обеспечивающие надежный ввод целых и вещественных чисел соответственно. Так как эти функции

универсальные и внеклассовые, их код целесообразно разместить в отдельном модуле.

Решение задачи, в котором реализованы рассмотренные концепции, представляет собой многофайловый проект, содержащий файлы `DBase.h`, `DBase.cpp`, `Man.h`, `Man.cpp`, `GetFunc.h`, `GetFunc.cpp` и `Main.cpp` (листинг 4.1).

Листинг 4.1. Многофайловый проект

```
//////////////////////////////////////////////// DBase.h
#pragma once
class DBase {
public:
    DBase(int);
    ~DBase();
    void InitInput();
    void Show();
    void SearchPayNotLess(double);
private:
    Man* pMan;
    int  nRecords;
};
//////////////////////////////////////////////// DBase.cpp
#include "Man.h"
#include "DBase.h"
DBase::DBase(int nRec) : nRecords(nRec),
pMan(new Man[nRec]) {}
DBase::~DBase() { if (pMan) delete [] pMan; }
void DBase::InitInput() {
    for ( int i = 0; i < nRecords; i++ ) cin >> *( pMan + i );  // 1
}
void DBase::Show() {
    cout << "================================" << endl;
    cout << "Содержимое базы данных:" << endl;
    for (int i = 0; i < nRecords; i++) cout << *( pMan + i );   // 2
}
void DBase::SearchPayNotLess(double anyPay) {
    bool not_found = true;
    for (int i = 0; i < nRecords; i++)
        if (( pMan + i )->GetPay() >= anyPay) {
            cout << *(pMan + i);
            not_found = false;
        }
    if (not_found) cout << "Таких сотрудников нет." << endl;
}
//////////////////////////////////////////////////////////// Man.h
#pragma once
#include <iostream>
```

```
#include <iomanip>
using namespace std;
const int l_name = 30;
class Man {
public:
    Man(int lName = 30);
    ~Man();
    double GetPay() const;
    friend istream& operator >>(istream&, Man&); // Операция
извлечения (ввода)

friend ostream& operator <<(ostream&, Man&); // Операция вставки
(вывода)
private:
    char*  pName;
    int    take_job_year;
    double pay;
};
////////////////////////////////////////////////////// Man.cpp
#include "windows.h"      // для работы в среде Windows с кириллицей
#include "Man.h"
#include "GetFunc.h"
Man::Man(int lName) { pName = new char[lName + 1]; }
Man::~Man() { if (pName) delete [] pName; }
double Man::GetPay() const { return pay; }
istream& operator >> (istream& in, Man& ob) { //Операция извлечения
(ввода)
    cout << "\nВведите данные в формате" << endl;
    cout << "Фамилия И.О. <Enter> Год поступления <Enter>";
    cout << " Оклад <Enter>:" << endl;
    in.getline(ob.pName, l_name);
    OemToChar(ob.pName, ob.pName); // для работы в среде Windows с
кириллицей
    ob.take_job_year = GetInt(in);                              // 3
    ob.pay = GetDouble(in);                                     // 4
    return in;
}
ostream& operator << (ostream& out, Man& ob) { //Операция вставки
(вывода)
    out << setw( 30 ) << setiosflags( ios::left );
    out << ob.pName << "  ";
    out << ob.take_job_year << "  ";
    out << ob.pay << endl;
    return out;
}
////////////////////////////////////////////////////// GetFunc.h
#pragma once
int GetInt(istream&);           // Ввод целого числа
```

```
double GetDouble(istream&);            // Ввод вещественного числа
////////////////////////////////////////////////////// GetFunc.cpp
#include "Man.h"
#include "GetFunc.h"
int GetInt(istream& in) { // ---------------------- ввод целого числа
    int value;
    while ( true ) {
        in >> value;                                        // 5
        if (in.peek() == '\n') {                            // 6
            in.get();                                       // 7
            break; }
        else {
            cout << "Повторите ввод (ожидается целое число):" << endl;
// 8
            in.clear();                                      // 9
            while ( in.get() != '\n' ) {};                   // 10
        }
    }
    return value;
}
double GetDouble(istream& in) { // ------- ввод вещественного числа
    double value;
    while ( true ) {
        in >> value;
        if (in.peek() == '\n') { in.get(); break; }
        else {
            cout << "Повторите ввод (ожидается вещественное число):"
<< endl;
            in.clear();
            while ( in.get() != '\n' ) {};
        }
    }
    return value;
}
////////////////////////////////////////////////////// Main.cpp
#include "Man.cpp"
#include "GetFunc.cpp"
#include "DBase.cpp"
int main() {
    const int nRecord = 10;      // количество записей в базе данных
    double any_pay;
    setlocale(LC_ALL, "Russian"); // только для работы в среде Windows
    DBase dBase( nRecord );
    dBase.InitInput();
    dBase.Show();
    cout << "Ввод данных завершен." << endl;
    cout << "========================================" << endl;
```

```
    cout << "Поиск сотрудников, чей оклад не меньше заданной
величины." << endl;
    cout << "Поиск завершается при вводе -1." << endl;
    while ( true ) {
        cout << "\nВведите величину оклада или -1: ";
        any_pay = GetDouble(cin);
        if (any_pay == -1)   break;
        dBase.SearchPayNotLess(any_pay);
    }
}
```

Обратите внимание на следующие моменты.

В реализации метода `InitInput` перегруженная для класса `Man` операция извлечения применяется к объекту, задаваемому выражением `*(pMan + i)`, то есть к объекту, адрес которого есть `pMan + i` (*оператор 1*).

Аналогичная адресация объекта для операции вставки использована в методе `Show` (*оператор 2*).

В реализации перегруженной операции извлечения (файл `Man.cpp`) обратите внимание на *операторы 3 и 4*, в которых вызываются функции `GetInt` и `GetDouble`, предназначенные для ввода из стандартного потока целых и вещественных чисел соответственно.

Реализация функций `GetInt` и `GetDouble` находится в файле `GetFunc.cpp`. Рассмотрим подробно первую из них. Ввод целого числа организован внутри бесконечного цикла `while` следующим образом.

Информация читается из входного потока *оператором 5*. Если в буфере типа `streambuf`, связанном с потоком `in`, находится изображение целого числа, завершаемое символом перевода строки `'\n'`, то по завершении операции извлечения в буфере останется только символ `'\n'`. *Оператор 6* проверяет это условие, используя метод `peek`. Если проверка завершилась успешно, *оператор 7* очищает входной буфер от символа `'\n'`, после чего происходит выход из цикла `while` с последующим возвратом из функции значения `value`. Если же введенная информация не является корректным изображением целого числа, то выполняются три действия:

- *оператор 8* выводит сообщение об этом, предлагая повторить ввод;
- сбрасываются флаги ошибок для потока `in` (*оператор 9*);
- с помощью внутреннего цикла `while` из входного буфера извлекаются все символы, вплоть до символа `'\n'` (*оператор 10*). Внешний цикл `while` повторяется сначала.

Функция `GetDouble` работает аналогично.

Функция main (файл `Main.cpp`) особых пояснений не требует. Алгоритм прозрачный и воспринимается легко благодаря выразительным именам методов.

В заключение отметим, что рассмотренная задача очень похожа на задачу 1.1. Мы советуем читателю сравнить решение этих двух задач, чтобы увидеть преимущества технологии ООП. Особенно впечатляет сравнение кодов функции `main`: положите рядом тексты листингов 1.1 и 4.1, и вы будете поражены совершенством, лаконичностью и красотой второго решения по сравнению с первым.

Итоги

1. Ввод и вывод представляются в программе как поток байтов и обычно выполняются через буфер. Потоки поддерживаются в библиотеке C++ с помощью иерархии классов, которая реализует безопасный ввод-вывод как стандартных, так и пользовательских типов данных.
2. Ввод-вывод бывает форматированный и неформатированный. Для форматированного ввода-вывода используются перегруженные операции << и >>, для неформатированного — методы стандартных классов.
3. Управление форматированием выполняется с помощью манипуляторов и методов стандартных классов.
4. Для вывода объектов пользовательских типов данных следует с помощью дружественных функций перегрузить операции чтения и извлечения.
5. Для обеспечения безопасного ввода необходимо диагностировать возможные ошибки, используя флаги состояния потока и (или) методы `peek`, `get` и `clear`.

Задания

Вариант 1

1. Определить класс с именем STUDENT, содержащий следующие поля: фамилия и инициалы; номер группы; успеваемость (массив из пяти элементов).

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа STUDENT.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти объектов типа STUDENT; записи должны быть упорядочены по возрастанию номера группы;
- вывод на дисплей фамилий и номеров групп для всех студентов, включенных в массив, если средний балл студента больше 4.0;
- если таких студентов нет, вывести соответствующее сообщение.

Вариант 2

1. Определить класс с именем STUDENT, содержащий следующие поля: фамилия и инициалы; номер группы; успеваемость (массив из пяти элементов).

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа STUDENT.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти объектов типа STUDENT; записи должны быть упорядочены по возрастанию среднего балла;
- вывод на дисплей фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5;
- если таких студентов нет, вывести соответствующее сообщение.

Вариант 3

1. Определить класс с именем STUDENT, содержащий следующие поля: фамилия и инициалы; номер группы; успеваемость (массив из пяти элементов).

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа STUDENT.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти объектов типа STUDENT; записи должны быть упорядочены по алфавиту;
- вывод на дисплей фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку 2;
- если таких студентов нет, вывести соответствующее сообщение.

Вариант 4

1. Определить класс с именем AEROFLOT, содержащий следующие поля: название пункта назначения рейса; номер рейса; тип самолета.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа AEROFLOT.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из семи объектов типа AEROFLOT; записи должны быть упорядочены по возрастанию номера рейса;
- вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры; если таких рейсов нет, выдать на дисплей соответствующее сообщение.

Вариант 5

1. Определить класс с именем AEROFLOT, содержащий следующие поля: название пункта назначения рейса; номер рейса; тип самолета.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа AEROFLOT.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из семи объектов типа AEROFLOT; записи должны быть размещены в алфавитном порядке по названиям пунктов назначения;
- вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры;
- если таких рейсов нет, выдать на дисплей соответствующее сообщение.

Вариант 6

1. Определить класс с именем WORKER, содержащий следующие поля:

- фамилия и инициалы работника; название занимаемой должности; год поступления на работу.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа WORKER.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти объектов типа WORKER; записи должны быть размещены по алфавиту;

- вывод на дисплей фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры;
- если таких работников нет, вывести на дисплей соответствующее сообщение.

Вариант 7

1. Определить класс с именем TRAIN, содержащий следующие поля: название пунк та назначения; номер поезда; время отправления.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа TRAIN.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа TRAIN; записи должны быть размещены в алфавитном порядке по названиям пунктов назначения;
- вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени;
- если таких поездов нет, выдать на дисплей соответствующее сообщение.

Вариант 8

1. Определить класс с именем TRAIN, содержащий следующие поля: название пунк та назначения; номер поезда; время отправления.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа TRAIN.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из шести объектов типа TRAIN; записи должны быть упорядочены по времени отправления поезда;
- вывод на экран информации о поездах, направляющихся в пункт, название которого введено с клавиатуры;
- если таких поездов нет, выдать на дисплей соответствующее сообщение.

Вариант 9

1. Определить класс с именем TRAIN, содержащий следующие поля: название пунк та назначения; номер поезда; время отправления.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа TRAIN.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа TRAIN; записи должны быть упорядочены по номерам поездов;
- вывод на экран информации о поезде, номер которого введен с клавиатуры; если таких поездов нет, выдать на дисплей соответствующее сообщение.

Вариант 10

1. Определить класс с именем MARSH, содержащий следующие поля:

- название начального пункта маршрута; название конечного пункта маршрута; номер маршрута.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа MARSH.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа MARSH; записи должны быть упорядочены по номерам маршрутов;
- вывод на экран информации о маршруте, номер которого введен с клавиатуры; если таких маршрутов нет, выдать на дисплей соответствующее сообщение.

Вариант 11

1. Определить класс с именем MARSH, содержащий следующие поля:

- название начального пункта маршрута; название конечного пункта маршрута; номер маршрута.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа MARSH.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа MARSH; записи должны быть упорядочены по номерам маршрутов;
- вывод на экран информации о маршрутах, которые начинаются или кончаются в пункте, название которого введено с клавиатуры;
- если таких маршрутов нет, выдать на дисплей соответствующее сообщение.

Вариант 12

1. Определить класс с именем NOTE, содержащий следующие поля: фамилия, имя; номер телефона; день рождения (массив из трех чисел).

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа NOTE.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа NOTE; записи должны быть упорядочены по датам дней рождения;
- вывод на экран информации о человеке, номер телефона которого введен с клавиатуры;
- если такого нет, выдать на дисплей соответствующее сообщение.

Вариант 13

1. Определить класс с именем NOTE, содержащий следующие поля: фамилия, имя; номер телефона; день рождения (массив из трех чисел).

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа NOTE.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа NOTE; записи должны быть размещены по алфавиту;
- вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры;
- если таких нет, выдать на дисплей соответствующее сообщение.

Вариант 14

1. Определить класс с именем NOTE, содержащий следующие поля: фамилия, имя; номер телефона; день рождения (массив из трех чисел).

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа NOTE.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа NOTE; записи должны быть упорядочены по трем первым цифрам номера телефона;
- вывод на экран информации о человеке, чья фамилия введена с клавиатуры; если такого нет, выдать на дисплей соответствующее сообщение.

Вариант 15

1. Определить класс с именем ZNAK, содержащий следующие поля: фамилия, имя; знак Зодиака; день рождения (массив из трех чисел).

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа ZNAK.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа ZNAK; записи должны быть упорядочены по датам дней рождения;
- вывод на экран информации о человеке, чья фамилия введена с клавиатуры;
- если такого нет, выдать на дисплей соответствующее сообщение.

Вариант 16

1. Определить класс с именем ZNAK, содержащий следующие поля: фамилия, имя; знак Зодиака; день рождения (массив из трех чисел).

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа ZNAK.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа ZNAK; записи должны быть упорядочены по датам дней рождения;
- вывод на экран информации о людях, родившихся под знаком, наименование которого введено с клавиатуры;
- если таких нет, выдать на дисплей соответствующее сообщение.

Вариант 17

1. Определить класс с именем ZNAK, содержащий следующие поля: фамилия, имя; знак Зодиака; день рождения (массив из трех чисел).

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа ZNAK.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа ZNAK; записи должны быть упорядочены по знакам Зодиака;
- вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры;
- если таких нет, выдать на дисплей соответствующее сообщение.

Вариант 18

1. Определить класс с именем PRICE, содержащий следующие поля:

- название товара;
- название магазина, в котором продается товар; стоимость товара в руб.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа PRICE.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа PRICE; записи должны быть размещены в алфавитном порядке по названиям товаров;
- вывод на экран информации о товаре, название которого введено с клавиатуры;
- если таких товаров нет, выдать на дисплей соответствующее сообщение.

Вариант 19

1. Определить класс с именем PRICE, содержащий следующие поля:

- название товара;
- название магазина, в котором продается товар; стоимость товара в руб.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа PRICE.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа PRICE; записи должны быть размещены в алфавитном порядке по названиям магазинов;
- вывод на экран информации о товарах, продающихся в магазине, название которого введено с клавиатуры;
- если такого магазина нет, выдать на дисплей соответствующее сообщение.

Вариант 20

1. Определить класс с именем ORDER, содержащий следующие поля: расчетный счет плательщика; расчетный счет получателя; перечисляемая сумма.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа ORDER.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа ORDER; записи должны быть размещены в алфавитном порядке по расчетным счетам плательщиков;
- вывод на экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры;
- если такого расчетного счета нет, выдать на дисплей соответствующее сообщение.