А.А.Быков

Сборник задач по программированию с решениями, часть 2

Общие положения

Экзамен по курсу «Программирование на языке C++» проводится в терминальном классе с установленным программным обеспечением, Microsoft Visual Studio или Borland C++ Builder по выбору экзаменуемого. Экзаменационный билет содержит три задания.

1) [1] Простое задание, для выполнения которого достаточно знакомства с основными конструкциями языка C++, в том числе иметь понятие об основных типах объектов, знать арифметические, логические и условные операторы if, операторы цикла for, одномерные и двумерные массивы с постоянной размерностью. Время выполнения: 30 минут.
2) [2] Задание средней сложности, для выполнения которого достаточно знания всех конструкций языка C++, включая одномерные и двумерные массивы с переменными границами (операторы new и delete), логические операторы, оператор switch, операторы цикла while и do, умение работать с текстовыми переменными и константами, знать основные стандартные функции для работы со строками, уметь выводить информацию в файл и считывать информацию из файла с помощью операторов << и >>. Время выполнения: 60 минут.
3) [3] Сложное задание, для выполнения которого требуется свободное владение всеми конструкциями языка в рамках программы первого семестра обучения, а также умение составить простейшую модель объекта. Время выполнения: 90 минут.

Общая продолжительность экзамена составляет не более 180 минут в терминальном классе. Студент, выполнивший первое задание, получает оценку не ниже 3. Студент, выполнивший второе задание, получает оценку не ниже 4. Студент, выполнивший третье задание, получает оценку 5. Во время выполнения заданий студент может использовать любые пособия, а также свои записи лекций и практических занятий. Можно использовать также любые файлы (в том числе C++ файлы), которые заранее должны быть сохранены на локальном диске в рабочей директории студента. В случае необходимости студент может обратиться за помощью к преподавателю. Преподаватель помогает студенту исправить ошибки в коде. В этом случае оценка 5 не ставится. Если курсант обращается за помощью несколько раз, не ставится оценка выше 3.

Список типовых задач объявляется заранее не позже чем за месяц до экзамена. Типовые задания разбираются на лекциях и отрабатываются на практических занятиях. Первое (простое) задание на экзамене может незначительно отличаться от типового задания из данного списка. Второе (средней сложности) задание также выбирается из данного списка и видоизменяется так, чтобы ход его решения ненамного отличался от типового. Например, если в типовом задании сказано, что оценка каждого курсанта потока за каждую неделю обучения в течение семестра

хранится в двумерном массиве и требуется найти среднюю оценку каждого курсанта за весть семестр, то на экзамене может быть поставлена задача: найти среднюю оценку всего потока за каждую неделю обучения.

Третье задание (сложное) будет похоже на одно из отработанных на лекциях или ПЗ заданий, но будет содержать новые элементы, которые студент должен разработать самостоятельно.

Задания из этого списка следует выполнять в среде Microsoft Visual Studio или Borland C++ Builder. При создании проекта указывайте опции “Win32 application” и “Console”. Вывод в файл осуществляйте с помощью оператора myofs<<, ввод из файла – с помощью оператора myifs>>, где myofs – объект типа ofstream, связанный с файлом с помощью оператора myofs.open(“имя файла”), myifs – объект типа ifstream, связанный с файлом с помощью оператора myifs.open(“имя файла”). После завершения работы с файлом не забудьте его закрыть, например, myofs.close(). Ввод с клавиатуры (при необходимости) осуществляйте с помощью оператора cin>>…, а вывод на экран cout<<…

В тех задачах, в которых форма отчета не определена явно, предполагается, что результаты направляются в файл protocol.txt.

Следует иметь в виду, что разделение задач по темам является условным, так как для решения некоторых задач придется обращаться к разным разделам курса программирования.

Задачи с решениями помечены символом +, например [+2].

Исходные тексты можно найти по адресу boombook@narod.ru

Сентябрь 2008 - январь 2009

Часть 2, C++ с классами

1. Классы

Задачи

1. [1] ***Простой класс.*** Создайте класс для хранения информации об автомобиле. Все поля данных объявите как public. Включите поле данных в виде массива постоянной размерности.
2. [1] ***Более сложный класс для хранения данных.*** Создайте класс для хранения информации об автомобиле. Все поля данных объявите как public. Создайте статический и динамический массивы объектов класса.
3. [1] ***Простой класс.*** Создайте класс для хранения времени (часы, минуты, секунды), включающий также два текстовых поля для названия и комментария. В учебных целях определите некоторые поля данных как private, остальные как public.
4. [2] ***Класс со стандартным уровнем функциональности.*** Создайте класс для хранения времени (часы, минуты, секунды), включающий также два текстовых поля для названия и комментария. Оснастите класс перегруженными операциями сложения, сравнения, сортировки и т.д.

Решения

Класс для хранения данных

5. [1] ***Простой класс.*** Создайте класс для хранения информации об автомобиле. Все поля данных объявите как public. Включите поле данных в виде массива постоянной размерности.

Решение.

```
// file MyStream.h:
#pragma once
#include <iostream>
#include <fstream>
#include <iomanip>
using std::setw;
using std::cout;
using std::cin;
using std::endl;
using std::ofstream;
using std::ifstream;
using namespace std;
extern ofstream myofs;
```

```
extern ifstream myifs;
using namespace std;

// file MyStream.cpp:
#include "MyStream.h"
ofstream myofs;
ifstream myifs;
// my.cpp : Defines the entry point for the
console application.
#include "MyStream.h"
class CMyCar1
{
public:
  int pos;
  char name[16];
  int move(int dist);
};
class CMyCar2{
public:
  int pos;
  char name[16];
  int move(int dist){
    pos += dist;
    return pos;
  }
  char* rename(char* ni){
    strcpy(name, ni);
    return name;
  }
  CMyCar2(){
    pos=0;
    strcpy(name, "Maserati");
  }
  CMyCar2(int pi, char* ni){
    pos=pi;
    strcpy(name, ni);
  }
  friend ofstream& operator<<(ofstream& ofs, CMyCar2&
cmc){
    ofs << "pos=" << cmc.pos << "; name=" << cmc.name;
    return ofs;
  }
```



```
};

int main()
{
  myofs.open("proto5n21m.txt");
  myofs << "Team 123, Иванов Иван Иванович. 18 февраля
2008 г." << endl;
  myofs << "Задание 5-21m. Простой класс пользователя"
<< endl << endl;

  myofs << "Упражнение 1. CMyCar1" << endl;
  CMyCar1 car1;
  myofs << "pos=" << car1.pos << "; name=" << car1.name
<< endl << endl;

  myofs << "Упражнение 2. CMyCar2, конструктор без
аргументов." << endl;
  CMyCar2 car2;
  myofs << "pos=" << car2.pos << "; name=" << car2.name
<< endl << endl;

  myofs << "Упражнение 3. CMyCar2, move(7)." << endl;
  car2.move(7);
  myofs << "pos=" << car2.pos << "; name=" << car2.name
<< endl << endl;

  myofs << "Упражнение 4. CMyCar2, rename(...)." <<
endl;
  car2.rename("Lamborghini");
  myofs << "pos=" << car2.pos << "; name=" << car2.name
<< endl << endl;

  myofs << "Упражнение 5. CMyCar2, конструктор с
аргументами." << endl;
  CMyCar2 car2m(17, "Maybach");
  myofs << "pos=" << car2m.pos << "; name=" <<
car2m.name << endl << endl;

  myofs << "Упражнение 6. CMyCar2, оператор <<." <<
endl;
  myofs << car2m << endl << endl;
```

```
  myofs << "Упражнение 7. CMyCar2, статический массив
обьектов." << endl;
  const int M=3;
  CMyCar2 cars[M];
  cars[1].pos=41; strcpy(cars[1].name, "BMW");
  cars[2].pos=73; strcpy(cars[2].name, "Mers");
  for(int m=0; m<M; m++){
    myofs << "cars[" << m << "] "; myofs << cars[m] <<
endl; }

  myofs << "Упражнение 8. CMyCar2, динамический массив
обьектов." << endl;
  int N=4;
  CMyCar2* avto = new CMyCar2[N];
  avto[1].pos=319; strcpy(avto[1].name, "GMC");
  avto[2].pos=538; strcpy(avto[2].name, "Yaguar");
  avto[3].pos=752; strcpy(avto[3].name, "Daimler");
  for(int n=0; n<N; n++){
    myofs << "avto[" << n << "] "; myofs << avto[n] <<
endl; }

  myofs << "Упражнение 9. CMyCar2, доступ через
указатель." << endl;
  avto->pos=1319; strcpy(avto->name, "Volga");
  (avto+1)->pos=9386; strcpy((avto+1)->name, "Lada");
  for(int n=0; n<N; n++){
    myofs << "avto[" << n << "] "; myofs << *(avto+n)
<< endl; }

  myofs << "Упражнение 8. CMyCar2, разрушение
динамического массива." << endl;
  delete[] avto;

  myofs.close();
  return 0;
}
```

Результат в файле протокола:

```
Team 123, Иванов Иван Иванович. 18 февраля 2008 г.
Задание 5-21m. Простой класс пользователя

Упражнение 1. CMyCar1 .// Плохой конструктор!!!
pos=-858993460; name=MMMMMMMMMMMMMMMMMMMM<•ЉдЁя□
```



```
Упражнение 2. CMyCar2, конструктор без аргументов.
pos=0; name=Maserati

Упражнение 3. CMyCar2, move(7).
pos=7; name=Maserati

Упражнение 4. CMyCar2, rename(...).
pos=7; name=Lamborghini

Упражнение 5. CMyCar2, конструктор с аргументами.
pos=17; name=Maybach

Упражнение 6. CMyCar2, оператор <<.
pos=17; name=Maybach

Упражнение 7. CMyCar2, статический массив обьектов.
cars[0] pos=0; name=Maserati
cars[1] pos=41; name=BMW
cars[2] pos=73; name=Mers
Упражнение 8. CMyCar2, динамический массив обьектов.
avto[0] pos=0; name=Maserati
avto[1] pos=319; name=GMC
avto[2] pos=538; name=Yaguar
avto[3] pos=752; name=Daimler
Упражнение 9. CMyCar2, доступ через указатель.
avto[0] pos=1319; name=Volga
avto[1] pos=9386; name=Lada
avto[2] pos=538; name=Yaguar
avto[3] pos=752; name=Daimler
Упражнение 8. CMyCar2, разрушение динамического массива.
```

Более сложный класс для хранения данных

[1] Создайте класс для хранения информации об автомобиле. Все поля данных объявите как public. Создайте статический и динамический массивы объектов класса.

Решение.

```
// MyCar.h : main header file for the CMyCar
class
class CMyCar
{
public:
  int hp;
  char* name;
```

```
private:
  int cyls, year;
  char* comment;
public:
  CMyCar(int  hi=325,  int  ci=6,  int  yi=2006,  char*
ni="Maserati", char* co="Adelaida");
  CMyCar(char* txt);
  int Setcomment(char* txt);
  int Setcyls(int m);
  int Setyear(int s);
  ~CMyCar();
  friend  ofstream&  operator<<(ofstream&  ofs,  CMyCar&
cmc);
};
const int BUFLEN = 256;
```

// MyCar.cpp : implementation file

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "MyStream.h"
#include "MyCar.h"
CMyCar::CMyCar(int hi, int mi, int si, char* ni,
char* ci)
{
  hp = hi;
  cyls = mi;
  year = si;
  name = new char[BUFLEN];
  strcpy(name, ni);
  comment = new char[BUFLEN];
  strcpy(comment, ci);
  myofs << " CMyCar: параметрический конструктор
      с выдачей в файл, ";
  myofs << *this << endl;
 }
CMyCar::CMyCar(char* txt){
  hp = 160;
  cyls = 4;
  year = 2004;
  name = new char[BUFLEN];
  strcpy(name, txt);
  comment = new char[BUFLEN];
  strcpy(comment, "CMyCar: Параметрический конструктор
```



```
      без выдачи в файл.");
 }

CMyCar::~CMyCar(){
  myofs << "CMyCar: in destructor, this="
      << this << ", *this=";
  myofs << *this << endl;
  delete name;
  delete comment;
}
int CMyCar::Setcomment(char* txt){
  strcpy(comment, txt);
  return false;
}
int CMyCar::Setcyls(int m){
  int prevmin=cyls;
  cyls = m;
  return prevmin;
}
int CMyCar::Setyear(int s){
  int prevsec=year;
  year = s;
  return prevsec;
}

ofstream& operator<<(ofstream& ofs, CMyCar& cmt){
  myofs << "(" << cmt.hp << ", " << cmt.cyls <<
      ", " << cmt.year;
  myofs << ") <" << cmt.name << "> <"
      << cmt.comment << ">";
  return ofs;
}
```

// my.cpp : Defines the entry point for the console application.

```
//
#include "MyStream.h"
#include "MyCar.h"
#include "math.h"
int main()
{
  myofs.open("proto5n22.txt");
  myofs << "Team 123, Иванов Иван Иванович.
```

```
      18 февраля 2008 г." << endl;
  myofs << "Задание 5-22. Простой класс пользователя"
      << endl << endl;
  {
  myofs << "Упражнение 1. Конструируем автоматический
      объект." << endl;
  myofs << " Конструктор по умолчанию:" << endl;
  CMyCar car1;
  myofs << endl << " Посмотрим на значение car1:" <<
endl;
  myofs << car1 << endl;
  myofs << " Посмотрим на значение &car1:" << endl;
  myofs << &car1 << endl;
  }
  myofs << endl;
  {
  myofs << "Упражнение 2. Конструируем автоматический
объект." << endl;
  myofs << " Конструктор с одним аргументом типа
char*:" << endl;
  CMyCar car2("Maybach");
  myofs << "   Посмотрим на значение car2:" << endl;
  myofs << car2 << endl;
  myofs << "   Посмотрим на значение &car2:" << endl;
  myofs << &car2 << endl;
  }
  myofs << endl;
  {
  myofs << "Упражнение 3. Конструируем автоматический
объект." << endl;
  myofs << " Конструктор с полным списком аргументов:"
<< endl;
  CMyCar car3(520, 12, 25, "Lamborghini", "Diabolo");
  myofs << "   Посмотрим на значение car3:" << endl;
  myofs << car3 << endl;
  myofs << "   Посмотрим на значение &car2:" << endl;
  myofs << &car3 << endl;
  }
  myofs << endl;
  {
  myofs << "Упражнение 3. Конструирование динамического
объекта с использованием оператора new." << endl;
  myofs << " Конструктор по умолчанию:" << endl;
```



```
  CMyCar *pcar1 = new CMyCar;
  myofs << "   Посмотрим на значение pcar1:" << endl;
  myofs << pcar1 << endl;
  myofs << "   Посмотрим на значение обьекта *pcar1:"
<< endl;
  myofs << *pcar1 << endl << endl;

  myofs << "Упражнение 4. Конструирование динамического
объекта с использованием оператора new." << endl;
  myofs << " Конструктор с полным списком аргументов:"
<< endl;
  CMyCar *pcar2 = new CMyCar(120, 4, 1998, "Lada",
"Kalina");
  myofs << "   Посмотрим на значение *pcar2:" << endl;
  myofs << *pcar2 << endl;
  myofs << "   Посмотрим на значение pcar2:" << endl;
  myofs << pcar2 << endl << endl;

  myofs << "Упражнение 5. Доступ к динамическому
обьекту через указатель." << endl;
  myofs << "   Значение pcar1->hp:" << endl;
  myofs << pcar1->hp << endl;
  myofs << "   Значение pcar1->cyls: доступа к этому
полю данных у нас нет." << endl;
///  myofs << pcar1->cyls << endl;
  myofs << "   Значение pcar1->name:" << endl;
  myofs << pcar1->name << endl << endl;

  myofs << "Упражнение 6. Разрушение динамического
обьекта." << endl;
  myofs << " Разрушаем обьект, на который указывает
pcar1:" << endl;
  delete pcar1;
  myofs << " Разрушаем обьект, на который указывает
pcar2:" << endl;
  delete pcar2;
  }
  myofs << endl;
  {
  myofs << "Упражнение 7. Статический массив (с
постоянными границами)." << endl;
  CMyCar cars[3];
```

```
  cars[1].Setcomment("Поменяем комментарий в данном
элементе");
  cars[1].Setcyls(8); // Прямого доступа к этому полю
нет.
  strcpy(cars[2].name, "Поменяем имя этого элемента");
  cars[2].Setyear(2002); // Прямого доступа к этому
полю нет.
  cars[2].hp = 11;    // Прямой доступ к этому полю
есть.
  myofs << "   Посмотрим на значение cars[0]:" << endl;
  myofs << cars[0] << endl;
  myofs << "   Посмотрим на значение cars[1]:" << endl;
  myofs << cars[1] << endl;
  myofs << "   Посмотрим на значение cars[2]:" << endl;
  myofs << cars[2] << endl << endl;
  }
  myofs << endl;
  {
  myofs << "Упражнение 7. Динамический массив (создаем
new, разрушаем delete)." << endl;
  CMyCar* pcars = new CMyCar[3];
  pcars[1].hp = 13;
  strcpy(pcars[1].name, "А я-колобок");
  pcars[2].hp = 21;
  strcpy(pcars[2].name, "А я-бармалей");
//    strcpy(pcars[1].comment, "Меняем значение
элемента массива [1]");
  myofs << "   Посмотрим на значение  pcars: ";
  myofs << pcars << endl;
  myofs << "   Посмотрим на значение *pcars: ";
  myofs << *pcars << endl;
  myofs << "   Посмотрим на значение pcars[0]: ";
  myofs << pcars[0] << endl;
  myofs << "   Посмотрим на значение pcars[1]: ";
  myofs << pcars[1] << endl;
  myofs << "   Посмотрим на значение *(pcars+1): ";
  myofs << *(pcars+1) << endl;
  myofs << "   Посмотрим на значение pcars+1: ";
  myofs << pcars+1 << endl;
  myofs << "   Посмотрим на значение &pcars[1]: ";
  myofs << &pcars[1] << endl;
  myofs << "   Посмотрим на значение pcars[2]: ";
  myofs << pcars[2] << endl;
```



```
  myofs << " А теперь разрушим динамический массив." <<
endl;
  myofs << " Посмотрите в протоколе, в каком порядке
разрушаются элементы динамического массива:" << endl;
  delete[] pcars;
  myofs << endl;

  myofs << "Упражнение 8. Разрушение обьекта
пользователя." << endl;
  CMyCar car3;
  myofs << "   Посмотрим на значение car3:" << endl;
  myofs << car3 << endl;
  myofs << "   Посмотрим на значение &car3:" << endl;
  myofs << &car3 << endl << endl;
  myofs << " Обьект car3 разрушается автоматически в
момент завершения приложения." << endl;
  myofs << " И сейчас мы это увидим в протоколе:" <<
endl << endl;
  }
  myofs.close();
  return 0;
}
```

Результат в файле протокола:

```
Team 123, Иванов Иван Иванович. 18 февраля 2008 г.
Задание 5-22. Простой класс пользователя

Упражнение 1. Конструируем автоматический объект.
 Конструктор по умолчанию:
 CMyCar: параметрический конструктор с выдачей в файл, (325,
6, 2006) <Maserati> <Adelaida>

 Посмотрим на значение car1:
(325, 6, 2006) <Maserati> <Adelaida>
 Посмотрим на значение &car1:
0012FF48
CMyCar: in destructor, this=0012FF48, *this=(325, 6, 2006)
<Maserati> <Adelaida>

Упражнение 2. Конструируем автоматический объект.
 Конструктор с одним аргументом типа char*:
   Посмотрим на значение car2:
(160, 4, 2004) <Maybach> <CMyCar: Параметрический конструктор
без выдачи в файл.>
   Посмотрим на значение &car2:
```

```
0012FF2C
CMyCar: in destructor, this=0012FF2C, *this=(160, 4, 2004)
<Maybach> <CMyCar: Параметрический конструктор без выдачи в
файл.>

Упражнение 3. Конструируем автоматический объект.
 Конструктор с полным списком аргументов:
 CMyCar: параметрический конструктор с выдачей в файл, (520,
12, 25) <Lamborghini> <Diabolo>
   Посмотрим на значение car3:
(520, 12, 25) <Lamborghini> <Diabolo>
   Посмотрим на значение &car2:
0012FF10
CMyCar: in destructor, this=0012FF10, *this=(520, 12, 25)
<Lamborghini> <Diabolo>

Упражнение 3. Конструирование динамического объекта с
использованием оператора new.
 Конструктор по умолчанию:
 CMyCar: параметрический конструктор с выдачей в файл, (325,
6, 2006) <Maserati> <Adelaida>
   Посмотрим на значение pcar1:
00336558
   Посмотрим на значение обьекта *pcar1:
(325, 6, 2006) <Maserati> <Adelaida>

Упражнение 4. Конструирование динамического объекта с
использованием оператора new.
 Конструктор с полным списком аргументов:
 CMyCar: параметрический конструктор с выдачей в файл, (120,
4, 1998) <Lada> <Kalina>
   Посмотрим на значение *pcar2:
(120, 4, 1998) <Lada> <Kalina>
   Посмотрим на значение pcar2:
003365A8

Упражнение 5. Доступ к динамическому обьекту через указатель.
   Значение pcar1->hp:
325
   Значение pcar1->cyls: доступа к этому полю данных у нас
нет.
   Значение pcar1->name:
Maserati

Упражнение 6. Разрушение динамического обьекта.
 Разрушаем обьект, на который указывает pcar1:
CMyCar: in destructor, this=00336558, *this=(325, 6, 2006)
<Maserati> <Adelaida>
```



```
 Разрушаем обьект, на который указывает pcar2:
CMyCar: in destructor, this=003365A8, *this=(120, 4, 1998)
<Lada> <Kalina>

Упражнение 7. Статический массив (с постоянными границами).
 CMyCar: параметрический конструктор с выдачей в файл, (325,
6, 2006) <Maserati> <Adelaida>
 CMyCar: параметрический конструктор с выдачей в файл, (325,
6, 2006) <Maserati> <Adelaida>
 CMyCar: параметрический конструктор с выдачей в файл, (325,
6, 2006) <Maserati> <Adelaida>
   Посмотрим на значение cars[0]:
(325, 6, 2006) <Maserati> <Adelaida>
   Посмотрим на значение cars[1]:
(325, 8, 2006) <Maserati> <Поменяем комментарий в данном
элементе>
   Посмотрим на значение cars[2]:
(11, 6, 2002) <Поменяем имя этого элемента> <Adelaida>

CMyCar: in destructor, this=0012FEDC, *this=(11, 6, 2002)
<Поменяем имя этого элемента> <Adelaida>
CMyCar: in destructor, this=0012FEC8, *this=(325, 8, 2006)
<Maserati> <Поменяем комментарий в данном элементе>
CMyCar: in destructor, this=0012FEB4, *this=(325, 6, 2006)
<Maserati> <Adelaida>

Упражнение 7. Динамический массив (создаем new, разрушаем
delete).
 CMyCar: параметрический конструктор с выдачей в файл, (325,
6, 2006) <Maserati> <Adelaida>
 CMyCar: параметрический конструктор с выдачей в файл, (325,
6, 2006) <Maserati> <Adelaida>
 CMyCar: параметрический конструктор с выдачей в файл, (325,
6, 2006) <Maserati> <Adelaida>
   Посмотрим на значение  pcars: 0033655C
   Посмотрим на значение *pcars: (325, 6, 2006) <Maserati>
<Adelaida>
   Посмотрим на значение pcars[0]: (325, 6, 2006) <Maserati>
<Adelaida>
   Посмотрим на значение pcars[1]: (13, 6, 2006) <А я-колобок>
<Adelaida>
   Посмотрим на значение *(pcars+1): (13, 6, 2006) <А я-
колобок> <Adelaida>
   Посмотрим на значение pcars+1: 00336570
   Посмотрим на значение &pcars[1]: 00336570
   Посмотрим на значение pcars[2]: (21, 6, 2006) <А я-
бармалей> <Adelaida>
 А теперь разрушим динамический массив.
```

```
 Посмотрите в протоколе, в каком порядке разрушаются элементы
динамического массива:
CMyCar: in destructor, this=00336584, *this=(21, 6, 2006) <A
я-бармалей> <Adelaida>
CMyCar: in destructor, this=00336570, *this=(13, 6, 2006) <A
я-колобок> <Adelaida>
CMyCar: in destructor, this=0033655C, *this=(325, 6, 2006)
<Maserati> <Adelaida>

Упражнение 8. Разрушение обьекта пользователя.
 CMyCar: параметрический конструктор с выдачей в файл, (325,
6, 2006) <Maserati> <Adelaida>
   Посмотрим на значение car3:
(325, 6, 2006) <Maserati> <Adelaida>
   Посмотрим на значение &car3:
0012FE8C

 Обьект car3 разрушается автоматически в момент завершения
приложения.
 И сейчас мы это увидим в протоколе:

CMyCar: in destructor, this=0012FE8C, *this=(325, 6, 2006)
<Maserati> <Adelaida>
```

Класс для хранения момента времени

[1] ***Простой класс.*** Создайте класс для хранения времени (часы, минуты, секунды), включающий также два текстовых поля для названия и комментария. В учебных целях определите некоторые поля данных как private, остальные как public.

Решение.

```
// MyTime.h : main header file for the CMyTime class
const int BUFLEN=256;
class CMyTime
{
public:
  int hou;
  char* name;
private:
  int min, sec;
  char* comment;
public:
```



```
  CMyTime(int hi=12, int mi=0, int si=0,
          char* ni="Default name",
          char* ci="Default comment");
  CMyTime(char* txt);
  int SetComment(char* txt);
  int Setmin(int m);
  int Setsec(int s);
  ~CMyTime();
  friend ofstream& operator<<(ofstream& ofs,
                              CMyTime& cmt);
};

// MyTime.cpp : implementation file
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "MyStream.h"
#include "MyTime.h"
CMyTime::CMyTime(int hi, int mi, int si,
                 char* ni, char* ci)
{
  // Параметрический конструктор:
  hou = hi;
  min = mi;
  sec = si;
  name = new char[BUFLEN];
  strcpy(name, ni);
  comment = new char[BUFLEN];
  strcpy(comment, ci);
  myofs << " CMyTime: параметрический конструктор с
выдачей в файл, ";
  myofs << *this << endl;
 }
CMyTime::CMyTime(char* txt)
{
  //  Конструктор по умолчанию:
  hou = 17;
  min = 18;
  sec = 19;
  name = new char[BUFLEN];
  strcpy(name, txt);
  comment = new char[BUFLEN];
```

```
  strcpy(comment, "CMyTime: Параметрический конструктор
без выдачи в файл.");
 }

CMyTime::~CMyTime()
{
  // Деструктор:
  myofs << "CMyTime: in destructor, this="
        << this << ", *this=";
  myofs << *this << endl;
  delete name;
  delete comment;
}
int CMyTime::SetComment(char* txt)
{
  strcpy(comment, txt);
  return false;
}
int CMyTime::Setmin(int m)
{
  int prevmin=min;
  min = m;
  return prevmin;
}
int CMyTime::Setsec(int s)
{
  int prevsec=sec;
  sec = s;
  return prevsec;
}

ofstream& operator<<(ofstream& ofs, CMyTime& cmt)
{
  myofs << "(" << setw(2) << cmt.hou << "."
        << setw(2) << cmt.min << "."
        << setw(2) << cmt.sec;
  myofs << ") <" << cmt.name
        << "> <" << cmt.comment << ">";
  return ofs;
}
// my.cpp : Defines the entry point for the
console application.
```



```
//
#include "MyStream.h"
#include "MyTime.h"
#include "math.h"

// Раскомментируйте следующую строку
// и обьясните последствия
// (будет ошибка исполнения, Run Time Error).
// Ошибку ищите в деструкторе.
//   CMyTime tim01("Это глобальный обьект (с глобальным
временем жизни)");
//   CMyTime tim02(5, 23, 17, "Kangaroo", "Australia
jumper");

int main()
{
  myofs.open("proto5n21.txt");
  myofs << "Team 123, Иванов Иван Иванович.
           18 февраля 2008 г." << endl;
  myofs << "Практическое задание 5-21.
           Простой класс пользователя" << endl << endl;
  CMyTime tim0("Это авто-обьект,
           конструируемый и разрушаемый
           вместе с main()");
  myofs << "Упражнение 1. Конструирование
           автоматических объектов." << endl;
  myofs << " &tim0="; myofs << &tim0 << endl;
  myofs << " tim0="; myofs << tim0 << endl << endl;
  myofs << " Параметрический конструктор
           с аргументами по умолчанию:" << endl;
  CMyTime tim1;
  myofs << " Параметрический конструктор:" << endl;
  CMyTime tim2(9, 15, 25, "Shark", "Sea beast");
  myofs << endl;
  myofs << "Упражнение 2. Конструирование объектов
           с использованием оператора new." << endl;
  myofs << " Конструируем обьект,
      используем оператор new (параметрический
      конструктор с аргументами по умолчанию):" << endl;
  CMyTime *ptim1 = new CMyTime;
  myofs << " Конструируем обьект,
      используем оператор new
      (параметрический конструктор с полным списком
```

```
      аргументов):" << endl;
  CMyTime *ptim2 = new CMyTime(14, 24, 34, "Kenguru",
"Africa animal");
  myofs << endl << "Упражнение 3. Доступ к обьекту
пользователя." << endl;
  myofs << endl << "   Посмотрим на значение обьекта
tim1:" << endl;
  myofs << tim1 << endl << endl;
  myofs << "   Посмотрим на значение &tim1:" << endl;
  myofs << &tim1 << endl << endl;
  myofs << "   Посмотрим на значение обьекта tim2:" <<
endl;
  myofs << tim2 << endl;
  myofs << endl << "   Посмотрим на значение &tim2:" <<
endl;
  myofs << " &tim2=" << &tim2 << endl << endl;
  myofs << "Упражнение 4. Доступ к обьекту пользователя
через указатель." << endl;
  myofs << "   Значение ptim1->hou:" << endl;
  myofs << ptim1->hou << endl;
  myofs << "   Значение ptim1->min: доступа к этому
полю данных у нас нет." << endl;
//    myofs << ptim1->min << endl;
  myofs << "   Значение ptim1->name:" << endl;
  myofs << ptim1->name << endl << endl;
  myofs << "   Посмотрим на значение обьекта *ptim1:"
<< endl;
  myofs << *ptim1 << endl << endl;
  myofs << "   Посмотрим на значение ptim1:" << endl;
  myofs << ptim1 << endl << endl;
  myofs << "   Посмотрим на значение *ptim2:" << endl;
  myofs << *ptim2 << endl << endl;
  myofs << "   Посмотрим на значение ptim2:" << endl;
  myofs << ptim2 << endl << endl;
  myofs << "Упражнение 5. Статический массив (с
постоянными границами)." << endl;
  CMyTime tims[3];
  tims[1].SetComment("Поменяем комментарий в данном
элементе");
  tims[1].Setmin(47); // Прямого доступа к этому полю
нет.
  strcpy(tims[2].name, "Поненяем имя этого элемента");
```



```
  tims[2].Setsec(33); // Прямого доступа к этому полю
нет.
  tims[2].hou = 11;   // Прямой доступ к этому полю
есть.
  myofs << "   Посмотрим на значение tims[0]:" << endl;
  myofs << tims[0] << endl;
  myofs << "   Посмотрим на значение tims[1]:" << endl;
  myofs << tims[1] << endl;
  myofs << "   Посмотрим на значение tims[2]:" << endl;
  myofs << tims[2] << endl << endl;
  myofs << "Упражнение 6. Динамический массив (создаем
new, разрушаем delete)." << endl;
  CMyTime* ptims = new CMyTime[3];
  ptims[1].hou = 13;
  strcpy(ptims[1].name, "я колобок");
  ptims[2].hou = 21;
  strcpy(ptims[2].name, "а я бармалей");
//    strcpy(ptims[1].comment, "Меняем значение
элемента массива [1]");
  myofs << "   Посмотрим на значение  ptims: ";
  myofs << ptims << endl;
  myofs << "   Посмотрим на значение *ptims: ";
  myofs << *ptims << endl;
  myofs << "   Посмотрим на значение *ptims[1]: ";
  myofs << ptims[1] << endl;
  myofs << "   Посмотрим на значение *(ptims+1): ";
  myofs << *(ptims+1) << endl;
  myofs << " А теперь разрушим динамический массив." <<
endl;
  myofs << " Посмотрите в протоколе, в каком порядке
разрушаются элементы динамического массива:" << endl;
  delete[] ptims;
  myofs << endl;
  myofs << "Упражнение 7. Разрушение обьекта
пользователя." << endl;
  myofs << " Разрушаем обьект, на который указывает
ptim1:" << endl;
  delete ptim1;
  myofs << " Разрушаем обьект, на который указывает
ptim2:" << endl << endl;
  delete ptim2;
  myofs << " Обьекты tim1 и tim2 разрушаются
автоматически в момент завершения приложения." << endl;
```

```
  myofs << " И сейчас мы это увидим в протоколе:" <<
endl << endl;
  myofs.close();
  return 0;
}
```

Результат в файле протокола:

```
Team 123, Иванов Иван Иванович. 18 февраля 2008 г.
Практическое задание 5-21. Простой класс пользователя

Упражнение 1. Конструирование автоматических объектов.
 &tim0=0012FF48
 tim0=(17.18.19)
<Это авто-обьект, конструируемый и разрушаемый
вместе с main()>
<CMyTime: Параметрический конструктор без выдачи в файл.>

Параметрический конструктор с аргументами по умолчанию:
 CMyTime: параметрический конструктор с выдачей в файл,
 (12. 0. 0) <Default name> <Default comment>
 Параметрический конструктор:
 CMyTime: параметрический конструктор с выдачей в файл,
 ( 9.15.25) <Shark> <Sea beast>

Упражнение 2. Конструирование объектов
              с использованием оператора new.
 Конструируем обьект, используем оператор new (параметрический
конструктор с аргументами по умолчанию):
 CMyTime: параметрический конструктор с выдачей в файл,
    (12. 0. 0) <Default name> <Default comment>
 Конструируем обьект, используем оператор new (параметрический
конструктор с полным списком аргументов):
 CMyTime: параметрический конструктор с выдачей в файл,
    (14.24.34) <Kenguru> <Africa animal>

Упражнение 3. Доступ к обьекту пользователя.

   Посмотрим на значение обьекта tim1:
(12. 0. 0) <Default name> <Default comment>

   Посмотрим на значение &tim1:
0012FF2C

   Посмотрим на значение обьекта tim2:
( 9.15.25) <Shark> <Sea beast>

   Посмотрим на значение &tim2:
```



```
 &tim2=0012FF10

Упражнение 4. Доступ к обьекту пользователя через указатель.
   Значение ptim1->hou:
12
   Значение ptim1->min: доступа к этому полю данных у нас нет.
   Значение ptim1->name:
Default name

   Посмотрим на значение обьекта *ptim1:
(12. 0. 0) <Default name> <Default comment>

   Посмотрим на значение ptim1:
00336770

   Посмотрим на значение *ptim2:
(14.24.34) <Kenguru> <Africa animal>

   Посмотрим на значение ptim2:
003367C0

Упражнение 5. Статический массив (с постоянными границами).
 CMyTime: параметрический конструктор с выдачей в файл,
    (12. 0. 0) <Default name> <Default comment>
 CMyTime: параметрический конструктор с выдачей в файл,
    (12. 0. 0) <Default name> <Default comment>
 CMyTime: параметрический конструктор с выдачей в файл,
    (12. 0. 0) <Default name> <Default comment>
   Посмотрим на значение tims[0]:
(12. 0. 0) <Default name> <Default comment>
   Посмотрим на значение tims[1]:
(12.47. 0) <Default name> <Поменяем комментарий
в данном элементе>
   Посмотрим на значение tims[2]:
(11. 0.33) <Поненяем имя этого элемента> <Default comment>

Упражнение 6. Динамический массив (создаем new,
разрушаем delete).
 CMyTime: параметрический конструктор с выдачей в файл,
    (12. 0. 0) <Default name> <Default comment>
 CMyTime: параметрический конструктор с выдачей в файл,
    (12. 0. 0) <Default name> <Default comment>
 CMyTime: параметрический конструктор с выдачей в файл,
    (12. 0. 0) <Default name> <Default comment>
   Посмотрим на значение  ptims: 00337B64
   Посмотрим на значение *ptims:
(12. 0. 0) <Default name> <Default comment>
   Посмотрим на значение *ptims[1]:
```

```
(13. 0. 0) <я колобок> <Default comment>
   Посмотрим на значение *(ptims+1):
(13. 0. 0) <я колобок> <Default comment>
 А теперь разрушим динамический массив.
 Посмотрите в протоколе, в каком порядке разрушаются элементы
динамического массива:
CMyTime: in destructor, this=00337B8C, *this=(21. 0. 0)
<а я бармалей> <Default comment>
CMyTime: in destructor, this=00337B78, *this=(13. 0. 0)
<я олобок> <Default comment>
CMyTime: in destructor, this=00337B64, *this=(12. 0. 0)
<Default name> <Default comment>

Упражнение 7. Разрушение обьекта пользователя.
 Разрушаем обьект, на который указывает ptim1:
CMyTime: in destructor, this=00336770, *this=(12. 0. 0)
<Default name> <Default comment>
 Разрушаем обьект, на который указывает ptim2:

CMyTime: in destructor, this=003367C0, *this=(14.24.34)
<Kenguru> <Africa animal>
 Обьекты tim1 и tim2 разрушаются автоматически
в момент завершения приложения.
 И сейчас мы это увидим в протоколе:
```

Класс со стандартным уровнем функциональности.

[2] ***Класс со стандартным уровнем функциональности.*** Создайте класс для хранения времени (часы, минуты, секунды), включающий также два текстовых поля для названия и комментария. Оснастите класс перегруженными операциями сложения, сравнения, сортировки и т.д.

Решение.

```
// file MyStream.h:
#pragma once
#include <iostream>
#include <fstream>
#include <iomanip>
using std::setw;
using std::cout;
using std::cin;
using std::endl;
using std::ofstream;
```



```
using std::ifstream;
extern ofstream myofs;
extern ifstream myifs;
// MyTime.h : main header file for the CMyTime
class
//
const int BUFLEN = 256;
class CMyTime{
private:
  int hou, min, sec;
  char* name;
  char* comment;
  int validate(int, int, int);
public:
  CMyTime(int ho, int mi, int se, char* nam, char*
com);
  CMyTime(const CMyTime& cmt);
  CMyTime();
  ~CMyTime();
  int seth(int hou_new){int hou_old = hou; hou =
validate(hou_new, 0, 23); return hou_old;}
  int setm(int min_new){int min_old = min; min =
validate(min_new, 0, 59); return min_old;}
  int sets(int sec_new){int sec_old = sec; sec =
validate(sec_new, 0, 59); return sec_old;}
  int setn(char* name_new);
  int setc(char* comment_new);
  int sethms(int hou_new, int min_new, int
sec_new){seth(hou_new); setm(min_new); sets(sec_new);
return 1;}
  int sethms(){hou = 12; min = 0; sec = 0; return 1;}
  int geth(){return hou;}
  int getm(){return min;}
  int gets(){return sec;}
  char* getn(){return name;}
  char* getc(){return comment;}
      int MyPrint(char*, char*);
      CMyTime& operator=(CMyTime& cmt);
  operator int()const; //see
      friend CMyTime operator+(CMyTime& cmt1, CMyTime&
cmt2);
      friend CMyTime operator+(int sec1, CMyTime& cmt2);
```

```
      friend CMyTime operator+(CMyTime& cmt1, int sec2);
      friend CMyTime operator-(CMyTime& cmt1, int sec2);
      friend ofstream& operator<<(ofstream& ofs,
CMyTime& cmt);
      friend ifstream& operator>>(ifstream& ofs,
CMyTime& cmt);
      bool operator<(const CMyTime& cmt);
      bool operator>(const CMyTime& cmt);
      bool operator==(const CMyTime& cmt);
      bool operator!=(const CMyTime& cmt);
      CMyTime& operator!();
  int swap(CMyTime *pt2);
};

int incr(int, int);
int decr(int, int);
int MonotIncrease(CMyTime *tims, int N);
int swap(CMyTime *pt1, CMyTime *pt2);
```

// file MyStream.cpp:

```
#include "MyStream.h"
ofstream myofs;
ifstream myifs;
```

// MyTime.cpp : implementation file

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "MyStream.h"
#include "MyTime.h"

int CMyTime::validate(int m, int mi, int ma){
  if(m<mi)m=mi;
  if(m>ma)m=ma;
  return m;
}
int CMyTime::setn(char* name_new){
  if(name != name_new){
    delete name;
    name = new char[strlen(name_new)+1];
    strcpy(name, name_new);
  }
  return 1;
}
```



```
int CMyTime::setc(char* comment_new){
  if(comment != comment_new){
    delete comment;
    comment = new char[strlen(comment_new)+1];
    strcpy(comment, comment_new);
  }
  return 1;
}
CMyTime::CMyTime(int hi, int mi=0, int si=0, char*
ni="name", char* ci="comment"){
  hou = validate(hi, 0, 23);
  min = validate(mi, 0, 59);
  sec = validate(si, 0, 59);
  name = new char[2];
  strcpy(name, "n");
  setn(ni);
  comment = new char[2];
  strcpy(comment, "c");
  setc(ci);
  myofs << "CMyTime: in parametric constructor=" <<
endl;
  myofs << *this << endl;
}
CMyTime::CMyTime(){
  hou = validate(12, 0, 23);
  min = validate(0, 0, 59);
  sec = validate(0, 0, 59);
  name = new char[2];
  strcpy(name, "n");
  setn("Полдень");
  comment = new char[2];
  strcpy(comment, "c");
  setc("Конструктор без аргументов класса CMyTime");
  myofs << "CMyTime: in default constructor=" << endl;
  myofs << *this << endl;
}
CMyTime::CMyTime(const CMyTime& cmt){
  hou = cmt.hou;
  min = cmt.min;
  sec = cmt.sec;
  name = new char[2];
  strcpy(name, "n");
  setn(cmt.name);
```

```
  comment = new char[2];
  strcpy(comment, "c");
  setc(cmt.comment);
  myofs << " CMyTime: in copy constructor" << endl;
  myofs << *this << endl;
}
CMyTime::~CMyTime()
{
  myofs << "CMyTime: in destructor" << endl;
  myofs << *this << endl;
  delete name;
  name=NULL;
  delete comment;
  comment=NULL;
}
CMyTime& CMyTime::operator=(CMyTime& cmt){
  if(this != &cmt){
        hou = cmt.hou;
        min = cmt.min;
        sec = cmt.sec;
    setn(cmt.name);
    setc(cmt.comment);
  }
      return *this;
}
bool CMyTime::operator<(const CMyTime& cmt){
      if(hou<cmt.hou) return true;
      if(hou>cmt.hou) return false;
      if(min<cmt.min) return true;
      if(min>cmt.min) return false;
  if(sec<cmt.sec) return true;
  return false;
}
bool CMyTime::operator>(const CMyTime& cmt){
      if(hou>cmt.hou) return true;
      if(hou<cmt.hou) return false;
      if(min>cmt.min) return true;
      if(min<cmt.min) return false;
  if(sec>cmt.sec) return true;
  return false;
}
bool CMyTime::operator==(const CMyTime& cmt){
      if(hou != cmt.hou) return false;
```



```
      if(min != cmt.min) return false;
  if(sec != cmt.sec) return false;
  return true;
}
CMyTime::operator int()const{
  return sec + 60*(min + 60*hou);
}
bool CMyTime::operator!=(const CMyTime& cmt){
      if(hou != cmt.hou) return true;
      if(min != cmt.min) return true;
  if(sec != cmt.sec) return true;
  return false;
}
CMyTime& CMyTime::operator!(){
  char buf[BUFLEN]; int k, m;
  strcpy(buf, name);
  k = strlen(buf);
  for(m=0; m<k; m++)
    name[m] = buf[k-m-1];
  strcpy(buf, comment);
  k = strlen(buf);
  for(m=0; m<k; m++)
    comment[m] = buf[k-m-1];
      if(min==0 && sec==0 && hou==0)
    return *this;
  if(min==0 && sec==0){
    hou = 24-hou;
    return *this;
  }
  if(sec==0){
    min = 60-min;
    hou = 23-hou;
    return *this;
  }
  sec = 60-sec;
      min = 59-min;
  hou = 23-hou;
  return *this;
}
CMyTime operator+(CMyTime& cmt1, CMyTime& cmt2){
  CMyTime cmt;
  int m, n, k;
  cmt.setn(cmt1.name);
```

```
  cmt.setc(cmt2.comment);
  k = cmt1.sec + cmt2.sec;
  m = 0;
  if (k > 59){
    k -= 60;
    m = 1;
  }
  cmt.sets(k);
  k = cmt1.min + cmt2.min + m;
  n = 0;
  if (k > 59){
    k -= 60;
    n = 1;
  }
  cmt.setm(k);
  k = cmt1.hou + cmt2.hou + n;
  cmt.seth(k%24);
  return cmt;
}
CMyTime operator+(int sec1, CMyTime& cmt2){
  CMyTime cmt;
  int m, n, k;
  cmt.setn(cmt2.name);
  cmt.setc(cmt2.comment);
  k = sec1 + cmt2.sec;
  m = 0;
  while(k > 59){
    k -= 60;
    m += 1;
  }
  cmt.sets(k);
  k = cmt2.min + m;
  n = 0;
  while(k > 59){
    k -= 60;
    n += 1;
  }
  cmt.setm(k);
  k = cmt2.hou + n;
  cmt.seth(k%24);
  return cmt;
}
CMyTime operator-(CMyTime& cmt1, int sec2){
```



```
  CMyTime cmt;
  int m, n, k;
  cmt.setn(cmt1.name);
  cmt.setc(cmt1.comment);
  k = cmt1.sec - sec2;
  m = 0;
  while(k < 0){
    k += 60;
    m -= 1;
  }
  cmt.sets(k);
  k = cmt1.min + m;
  n = 0;
  while(k < 0){
    k += 60;
    n -= 1;
  }
  cmt.setm(k);
  k = cmt1.hou + n;
  cmt.seth(k%24);
  return cmt;
}
CMyTime operator+(CMyTime& cmt1, int sec2){
  CMyTime cmt;
  cmt = sec2 + cmt1;
  return cmt;
}
int CMyTime::MyPrint(char* title, char* buf){
      char buf1[BUFLEN];
      sprintf(buf1, "%3d hour, %3d min, %3d sec ", hou,
min, sec);
      strcpy(buf, title);
      strcat(buf, buf1);
      strcat(buf, " <");
      strcat(buf, name);
  strcat(buf, " ");
  strcat(buf, comment);
      return 1;
}
ofstream& operator<<(ofstream& ofs, CMyTime& cmt){
  myofs << "(" << setw(2) << cmt.hou << "." << setw(2)
<< cmt.min << "." << setw(2) << cmt.sec << ") ";
```

```
  myofs << " " << setw(10) << cmt.name << ";  " <<
cmt.comment;
  return ofs;
}
ifstream& operator>>(ifstream& ifs, CMyTime& cmt){
  ifs >> cmt.hou >> cmt.min >> cmt.sec;
  char buf[256];
  myifs >> buf;
  delete cmt.name;
  cmt.name = new char[strlen(buf)+1];
  strcpy(cmt.name, buf);
  myifs.getline(buf, 200);
  delete cmt.comment;
  cmt.comment = new char[strlen(buf)+1];
  strcpy(cmt.comment, buf);
  return ifs;
}
int swap(CMyTime* ptim1, CMyTime* ptim2){
  CMyTime* ptim = new CMyTime;
  *ptim = *ptim1;
  *ptim1 = *ptim2;
  *ptim2 = *ptim;
  delete ptim;
  return 1;
}
int CMyTime::swap(CMyTime* ptim2){
  CMyTime* ptim = new CMyTime;
  *ptim = *this;
  *this = *ptim2;
  *ptim2 = *this;
  delete ptim;
  return 1;
}

int decr(int ncur, int N){ncur--; if(ncur<0) ncur=0;
return ncur;}
int incr(int ncur, int N){ncur++; if(ncur>=N) ncur=N-1;
return ncur;}

int MonotIncrease(CMyTime *tims, int N){
  int m, n, k;
  CMyTime* pcmt = new CMyTime;
  for(m=0; m<N-1; m++){
```



```
    k = m;
    for(n=m+1; n<N; n++){
      if(tims[k]>tims[n]) k=n;
    }
    if(m<k){
      *pcmt = tims[m];
      tims[m] = tims[k];
      tims[k] = *pcmt;
    }
  }
  delete pcmt;
  return 1;
}
// my.cpp : Defines the entry point for the
console application.
//
#include "MyStream.h"
#include "MyTime.h"
#include "math.h"
using namespace std;

int main(){
{
  myofs.open("proto5n23.txt");
  myofs << "Упражнение 1. Сконструируем обьект класса
CMyTime и инициализируем его:" << endl;
  CMyTime cmtx(21, 15, 47, "Caterpillar", "Длинная
зеленая гусеница, сидящая на листе капусты");
  myofs.close();
  myofs.open("mydata.dat");
  myofs << cmtx.geth() << " " << cmtx.getm() << " " <<
cmtx.gets() << " " << cmtx.getn() << " " << cmtx.getc()
<< endl;
  myofs.close();
  myofs.open("proto5n23.txt", ios_base::app);
  {
    myofs << "Team 123, Иванов Иван Иванович. 18
февраля 2008 г." << endl;
    myofs << "Практическое задание 5-23. Класс
пользователя" << endl << endl;
  }
```

```
  myofs << "Упражнение 1. Сконструируем обьект класса
CMyTime и инициализируем его:" << endl;
  CMyTime cmt(19, 33, 51, "Cat", "Млекопитающее отряда
кошачих, домашнее животное");
  myofs << cmt << endl;
  myofs << " Теперь причитаем из файла данные в обьект
класса CMyTime:" << endl;
  myifs.open("mydata.dat");
  myifs >> cmt;
  myifs.close();
  myofs << cmt << endl;

  myofs << "Упражнение 3. Сконструируем обьект класса
CMyTime с помощью оператора new:" << endl;
  CMyTime *ptim1 = new CMyTime(14, 24, 34, "Kenguru",
"Africa animal");
  CMyTime *ptim2 = new CMyTime(21, 31, 41, "Pinguin",
"Sea bird");
  CMyTime *ptim3 = new CMyTime(3, 4, 5, "Shark",
"Underwater beast");
  CMyTime *ptim4 = new CMyTime(*ptim3);
  CMyTime *ptim5 = new CMyTime(7, 12, 53, "Dolphin",
"Water animal");
  CMyTime *ptim6 = new CMyTime(16, 42, 17, "Medved",
"Chi-huana boy");

  myofs << "Упражнение 3. Сконструируем массив обьектов
класса CMyTime:" << endl;
  const int N = 8;
  CMyTime *tims = new CMyTime[N];
  char *Suits[] = {"Norka", "Medved", "Lyra", "GavGav",
"Labrador", "Chi-HuaHua", "Taxa", "Murka"};
  char buf[BUFLEN];
  int ncur = 0;
  for(int n=0; n<N; n++){
    tims[n].sethms(rand()%24, rand()%60, rand()%60);
    tims[n].setn(Suits[n]);
    int len = 3+rand()%8;
    for(int k=0; k<len; k++){
      strcpy(buf, Suits[rand()%8]);
      tims[n].setc(buf);
    }
  }
```



```
  myofs << endl << " tims after random initialization:"
<< endl;
  for(int n=0; n<N; n++){myofs << tims[n] << endl;}
myofs << endl;

  myofs << "Упражнение 3. Сортируем массив обьектов
класса CMyTime:" << endl;
  MonotIncrease(tims, N);
  myofs << endl << " tims after sorting:" << endl;
  for(int n=0; n<N; n++){myofs << tims[n] << endl;}
myofs << endl << endl;
  char buf1[BUFLEN];
{
  CMyTime *pt1 = new CMyTime(14, 24, 34, "Kenguru",
"Africa animal");
  CMyTime *pt2 = new CMyTime(21, 31, 41, "Pinguin",
"Sea bird");
  myofs << "Упражнение 3. Операция swap" << endl;
  myofs << " Два обьекта до операции swap" << endl;
  myofs << " *pt1=:"; myofs << *pt1 << endl;
  myofs << " *pt2=:"; myofs << *pt2 << endl << endl;
  swap(pt1, pt2);
  myofs << endl << " Два обьекта после операции swap"
<< endl;
  myofs << " *pt1=:"; myofs << *pt1 << endl;
  myofs << " *pt2=:"; myofs << *pt2 << endl << endl;
}
{
  CMyTime *pt1 = new CMyTime(14, 24, 34, "Kenguru",
"Africa animal");
  CMyTime *pt2 = new CMyTime(21, 31, 41, "Pinguin",
"Sea bird");
  myofs << "Упражнение 3. Операция !=" << endl;
  myofs << " До операции !=" << endl;
  myofs << " *pt1=:"; myofs << *pt1 << endl;
  myofs << " *pt2=:"; myofs << *pt2 << endl << endl;
  *pt2 = !*pt1;
  !*pt1;
  myofs << " После операции !=" << endl;
  myofs << " *pt1=:"; myofs << *pt1 << endl;
  myofs << " *pt2=:"; myofs << *pt2 << endl << endl;
}
  myofs << "Упражнение 3. Операция =" << endl;
```

```
  myofs << " Два обьекта до операции =" << endl;
  myofs << " *ptim3=:"; myofs << *ptim3 << endl;
  myofs << " *ptim4=:"; myofs << *ptim4 << endl <<
endl;
  *ptim3 = *ptim4;
  myofs << " Два обьекта после операции !=" << endl;
  myofs << " *ptim3=:"; myofs << *ptim3 << endl;
  myofs << " *ptim4=:"; myofs << *ptim4 << endl <<
endl;

  ptim1->MyPrint("ptim1=", buf);
  myofs << buf << endl;
  ptim2->MyPrint("ptim2=", buf);
  myofs << buf << endl;
  ptim3->MyPrint("ptim3=", buf);
  myofs << buf << endl;
  ptim4->MyPrint("ptim4=", buf);
  myofs << buf << endl;
  *(ptim6) = *ptim1 + *ptim2;
  int m = *(ptim6); //see
  sprintf(buf1, "ptim1+ptim2= (%6d) ", m);
  ptim6->MyPrint(buf1, buf);
  myofs << buf << endl;

//void CmyDlg::OnBnClickedButtonRight(){
  // TODO: Add your control notification handler code
here
  myofs << " Button RIGHT pressed." << endl;
  int n = incr(ncur, N);
  int n1 = n+1; if(n1 >= N) n1 = n;
  tims[n].MyPrint("", buf1);
  strcpy(buf, buf1);
  if(tims[n] < tims[n1]) strcat(buf, "    <    ");
  else
  if(tims[n] > tims[n1]) strcat(buf, "    >    ");
  else
    strcat(buf, "    =    ");
  tims[n1].MyPrint("", buf1);
  strcat(buf, buf1);
  myofs << buf << endl;
  *(ptim6) = tims[n] + tims[n1];
  m = *(ptim6);
  sprintf(buf1, " cur+next= (%6d) ", m);
```



```
  ptim6->MyPrint(buf1, buf);
  myofs << buf << " Button RIGHT pressed." << endl;
// }
// void CmyDlg::OnBnClickedButtonLeft(){
  // TODO: Add your control notification handler code
here
  myofs << " Button LEFT pressed." << endl;
  n = decr(ncur, N);
  n1 = n+1; if(n1 >= N) n1 = n;
  tims[n].MyPrint("", buf1);
  strcpy_s(buf, BUFLEN, buf1);
  if(tims[n] < tims[n1]) strcat_s(buf, BUFLEN, "    <
");
  else
  if(tims[n] > tims[n1]) strcat_s(buf, BUFLEN, "    >
");
  else
    strcat_s(buf, BUFLEN, "    =    ");
  tims[n1].MyPrint("", buf1);
  strcat_s(buf, BUFLEN, buf1);
  myofs << buf << endl;
  *(ptim6) = tims[n] + tims[n1];
  m = *(ptim6);
  sprintf(buf1, " cur+next= (%6d) ", m);
  ptim6->MyPrint(buf1, buf);
  myofs << buf << " Button LEFT pressed." << endl;
// }
  delete ptim1;
  delete ptim2;
  delete ptim3;
  delete ptim4;
  delete ptim5;
  delete ptim6;
  delete[] tims;
  }
  myofs.close();
  return 0;
}
```

Результат в файле протокола:

```
Упражнение 1. Сконструируем обьект класса CMyTime и
инициализируем его:
CMyTime: in parametric constructor=
```

```
(21.15.47)  Caterpillar;  Длинная зеленая гусеница, сидящая на
листе капусты
Team 123, Иванов Иван Иванович. 18 февраля 2008 г.
Практическое задание 5-23. Класс пользователя

Упражнение  1.  Сконструируем  обьект  класса  CMyTime  и
инициализируем его:
CMyTime: in parametric constructor=
(19.33.51)              Cat;   Млекопитающее отряда кошачих,
домашнее животное
(19.33.51)              Cat;   Млекопитающее отряда кошачих,
домашнее животное
 Теперь причитаем из файла данные в обьект класса CMyTime:
(21.15.47)  Caterpillar;   Длинная зеленая гусеница, сидящая
на листе капусты
Упражнение 3. Сконструируем обьект класса CMyTime с помощью
оператора new:
CMyTime: in parametric constructor=
(14.24.34)     Kenguru;  Africa animal
CMyTime: in parametric constructor=
(21.31.41)     Pinguin;  Sea bird
CMyTime: in parametric constructor=
( 3. 4. 5)       Shark;  Underwater beast
 CMyTime: in copy constructor
( 3. 4. 5)       Shark;  Underwater beast
CMyTime: in parametric constructor=
( 7.12.53)     Dolphin;  Water animal
CMyTime: in parametric constructor=
(16.42.17)      Medved;  Chi-huana boy
Упражнение 3. Сконструируем массив обьектов класса CMyTime:
CMyTime: in default constructor=
(12. 0. 0)       Полдень;   Конструктор без аргументов класса
CMyTime
CMyTime: in default constructor=
(12. 0. 0)       Полдень;   Конструктор без аргументов класса
CMyTime
CMyTime: in default constructor=
(12. 0. 0)       Полдень;   Конструктор без аргументов класса
CMyTime
CMyTime: in default constructor=
(12. 0. 0)       Полдень;   Конструктор без аргументов класса
CMyTime
CMyTime: in default constructor=
(12. 0. 0)       Полдень;   Конструктор без аргументов класса
CMyTime
CMyTime: in default constructor=
(12. 0. 0)       Полдень;   Конструктор без аргументов класса
CMyTime
```



```
CMyTime: in default constructor=
(12. 0. 0)       Полдень;   Конструктор без аргументов класса
CMyTime
CMyTime: in default constructor=
(12. 0. 0)       Полдень;   Конструктор без аргументов класса
CMyTime

 tims after random initialization:
(22.47.41)       Norka;  Medved
( 3. 1. 5)      Medved;  GavGav
(12.51.36)        Lyra;  Taxa
(21.18.11)      GavGav;  GavGav
( 3.23.54)    Labrador;  Murka
(11.28.53)  Chi-HuaHua;  Medved
(11.16.58)        Taxa;  Chi-HuaHua
(10. 9.50)       Murka;  Labrador

Упражнение 3. Сортируем массив обьектов класса CMyTime:
CMyTime: in default constructor=
(12. 0. 0)       Полдень;   Конструктор без аргументов класса
CMyTime
CMyTime: in destructor
(22.47.41)       Norka;  Medved

 tims after sorting:
( 3. 1. 5)      Medved;  GavGav
( 3.23.54)    Labrador;  Murka
(10. 9.50)       Murka;  Labrador
(11.16.58)        Taxa;  Chi-HuaHua
(11.28.53)  Chi-HuaHua;  Medved
(12.51.36)        Lyra;  Taxa
(21.18.11)      GavGav;  GavGav
(22.47.41)       Norka;  Medved


CMyTime: in parametric constructor=
(14.24.34)     Kenguru;  Africa animal
CMyTime: in parametric constructor=
(21.31.41)     Pinguin;  Sea bird
Упражнение 3. Операция swap
 Два обьекта до операции swap
 *pt1=:(14.24.34)     Kenguru;  Africa animal
 *pt2=:(21.31.41)     Pinguin;  Sea bird

CMyTime: in default constructor=
(12. 0. 0)       Полдень;   Конструктор без аргументов класса
CMyTime
CMyTime: in destructor
```

```
(14.24.34)     Kenguru;  Africa animal

 Два обьекта после операции swap
 *pt1=:(21.31.41)     Pinguin;  Sea bird
 *pt2=:(14.24.34)     Kenguru;  Africa animal

CMyTime: in parametric constructor=
(14.24.34)     Kenguru;  Africa animal
CMyTime: in parametric constructor=
(21.31.41)     Pinguin;  Sea bird
Упражнение 3. Операция !=
 До операции !=
 *pt1=:(14.24.34)     Kenguru;  Africa animal
 *pt2=:(21.31.41)     Pinguin;  Sea bird

 После операции !=
 *pt1=:(14.24.34)     Kenguru;  Africa animal
 *pt2=:( 9.35.26)     urugneK;  lamina acirfA

Упражнение 3. Операция =
 Два обьекта до операции =
 *ptim3=:( 3. 4. 5)       Shark;  Underwater beast
 *ptim4=:( 3. 4. 5)       Shark;  Underwater beast

 Два обьекта после операции !=
 *ptim3=:( 3. 4. 5)       Shark;  Underwater beast
 *ptim4=:( 3. 4. 5)       Shark;  Underwater beast

ptim1= 14 hour,  24 min,  34 sec  <Kenguru Africa animal
ptim2= 21 hour,  31 min,  41 sec  <Pinguin Sea bird
ptim3=  3 hour,   4 min,   5 sec  <Shark Underwater beast
ptim4=  3 hour,   4 min,   5 sec  <Shark Underwater beast
CMyTime: in default constructor=
(12. 0. 0)       Полдень;  Конструктор без аргументов класса
CMyTime
 CMyTime: in copy constructor
(11.56.15)     Kenguru;  Sea bird
CMyTime: in destructor
(11.56.15)     Kenguru;  Sea bird
CMyTime: in destructor
(11.56.15)     Kenguru;  Sea bird
ptim1+ptim2= ( 42975)  11 hour,  56 min,  15 sec  <Kenguru Sea
bird
 Button RIGHT pressed.
  3 hour,  23 min,  54 sec  <Labrador Murka    <     10 hour,
9 min,  50 sec  <Murka Labrador
CMyTime: in default constructor=
```



```
(12. 0. 0)       Полдень;  Конструктор без аргументов класса
CMyTime
 CMyTime: in copy constructor
(13.33.44)    Labrador;  Labrador
CMyTime: in destructor
(13.33.44)    Labrador;  Labrador
CMyTime: in destructor
(13.33.44)    Labrador;  Labrador
 cur+next= ( 48824)  13 hour,  33 min,  44 sec  <Labrador
Labrador Button RIGHT pressed.
 Button LEFT pressed.
  3 hour,   1 min,   5 sec  <Medved GavGav    <      3 hour,
23 min,  54 sec  <Labrador Murka
CMyTime: in default constructor=
(12. 0. 0)       Полдень;  Конструктор без аргументов класса
CMyTime
 CMyTime: in copy constructor
( 6.24.59)      Medved;  Murka
CMyTime: in destructor
( 6.24.59)      Medved;  Murka
CMyTime: in destructor
( 6.24.59)      Medved;  Murka
 cur+next= ( 23099)   6 hour,  24 min,  59 sec  <Medved Murka
Button LEFT pressed.
CMyTime: in destructor
(14.24.34)     Kenguru;  Africa animal
CMyTime: in destructor
(21.31.41)     Pinguin;  Sea bird
CMyTime: in destructor
( 3. 4. 5)       Shark;  Underwater beast
CMyTime: in destructor
( 3. 4. 5)       Shark;  Underwater beast
CMyTime: in destructor
( 7.12.53)     Dolphin;  Water animal
CMyTime: in destructor
( 6.24.59)      Medved;  Murka
CMyTime: in destructor
(22.47.41)       Norka;  Medved
CMyTime: in destructor
(21.18.11)      GavGav;  GavGav
CMyTime: in destructor
(12.51.36)        Lyra;  Taxa
CMyTime: in destructor
(11.28.53)  Chi-HuaHua;  Medved
CMyTime: in destructor
(11.16.58)        Taxa;  Chi-HuaHua
CMyTime: in destructor
(10. 9.50)       Murka;  Labrador
```

```
CMyTime: in destructor
( 3.23.54)    Labrador;  Murka
CMyTime: in destructor
( 3. 1. 5)      Medved;  GavGav
CMyTime: in destructor
(21.15.47)  Caterpillar;    Длинная зеленая гусеница, сидящая
на листе капусты
CMyTime: in destructor
(21.15.47)  Caterpillar;  Длинная зеленая гусеница, сидящая на
листе капусты
```

Название задачи

Решение.
Результат в файле протокола:

Название задачи

Решение.
Результат в файле протокола:

Название задачи

Решение.
Результат в файле протокола: