ЛАБОРАТОРНАЯ РАБОТА №8

КЛАССЫ

ЦЕЛЬ РАБОТЫ: изучить принципы написания программ, использующих объектно-ориентированный подход на языке программирования C++.

ПРИМЕРЫ ПРОГРАММ

Пример 8.1. Поиск в простой базе (массив объектов)

В текстовом файле хранится база отдела кадров предприятия. На предприятии 100 сотрудников. Каждая строка файла содержит запись об одном сотруднике. Формат записи: фамилия и инициалы (30 поз., фамилия должна начинаться с первой позиции), год рождения (5 поз.), оклад (10 поз.). Написать программу, которая по заданной фамилии выводит на экран сведения о сотруднике, подсчитывая средний оклад всех запрошенных сотрудников.

//Man.h

```
const int l_name = 30;
const int l_year = 5;
const int l_pay = 10;
const int l_buf = l_name + l_year + l_pay;
class Man {
public:
  Man(int lName = 30);
  ~Man();
  bool CompareName(const char*) const;
  int GetBirthYear() const { return birth_year; }
  float GetPay() const { return pay; }
  char* GetName() const { return pName; }
  void Print() const;
  void SetBirthYear(const char*);
  void SetName(const char*);
  void SetPay(const char*);
private:
  char* pName;
  int birth_year;
  float pay;
};
```

//Man.cpp

```
#include <iostream>
#include <cstring>
```

```
#include "Man.h"
using namespace std;
Man::Man(int lName) {
  cout << "Constructor is working... ";
  pName = new char[lName + 1];
}
Man::~Man() {
  cout << "Destructor is working...";
  delete[] pName;
}
void Man::SetName(const char* fromBuf) {
  strncpy(pName, fromBuf, l_name);
  pName[l_name] = 0;
}
void Man::SetBirthYear(const char* fromBuf) {
  birth_year = atoi(fromBuf + l_name);
}
void Man::SetPay(const char* fromBuf) {
  pay = atof(fromBuf + l_name + l_year);
}
bool Man::CompareName(const char* name) const {
  if ((strstr(pName, name)) && (pName[strlen(name)] == ' ')) return true;
  else return false;
}
void Man::Print() const {
  cout << pName << birth_year << ' ' << pay << endl;
}
```

//Main.cpp

```
#include <windows.h> // содержит прототип OemToChar
#include <iostream>
#include <fstream>
#include "Man.h"
using namespace std;
const char filename[] = "dbase.txt";
int main() {
  const int maxn_record = 10;
  Man man[maxn_record];
  char buf[l_buf + 1];
  char name[l_name + 1];
  setlocale(LC_ALL, "Russian");
  ifstream fin(filename);
  if (!fin) { cout << "Нет файла " << filename << endl; return 1; }
  int i = 0;
  while (fin.getline(buf, l_buf)) {
```

```
    if (i >= maxn_record) { cout << "Слишком длинный файл"; return 1; }
    man[i].SetName(buf);
    man[i].SetBirthYear(buf);
    man[i].SetPay(buf);
    i++;
  }
  int n_record = i, n_man = 0;
  float mean_pay = 0;
  while (true) {
    cout << "Введите фамилию или слово end: ";
    cin >> name;
    OemToChar(name, name); // для ввода кириллицы
    if (0 == strcmp(name, "end")) break;
    bool not_found = true;
    for (int i = 0; i < n_record; ++i) {
      if (man[i].CompareName(name)) {
        man[i].Print();
        n_man++; mean_pay += man[i].GetPay();
        not_found = false;
        break;
      }
    }
    if (not_found) cout << "Такого сотрудника нет" << endl;
  }
  if (n_man) cout << " Средний оклад: " << mean_pay / n_man << endl;
}
```

Пример 8.2. Реализация класса треугольников

Для некоторого множества заданных координатами своих вершин треугольников найти треугольник максимальной площади (если максимальную площадь имеют несколько треугольников, найти первый из них). Предусмотреть возможность перемещения треугольников и проверки включения одного треугольника в другой. Программа должна содержать меню, позволяющее выполнить проверку всех методов класса. Для реализации этой задачи составить описание класса треугольников на плоскости. Предусмотреть возможность объявления в клиентской программе (main) экземпляра треугольника с заданными координатами вершин. Предусмотреть наличие в классе методов, обеспечивающих: 1) перемещение треугольников на плоскости; 2) определение отношения > для пары заданных треугольников (мера сравнения – площадь треугольников); 3) определение отношения включения типа: «Треугольник 1 входит (не входит) в Треугольник 2».

```
// Point.h
#ifndef POINT_H
#define POINT_H
```

```
class Point {
public:
  // Конструктор
  Point(double _x = 0, double _y = 0) : x(_x), y(_y) {}
  // Другие методы
  void Show() const;
  double DistanceTo(Point) const; // расстояние до другой точки
  void operator +=(Point&);
  double x, y;
};
#endif /* POINT_H */
```

// Point.cpp

```
#include <iostream>
#include <math.h>
#include "Point.h"
using namespace std;
void Point::Show() const {
  cout << "  (" << x << ", " << y << ")";
}
void Point::operator +=(Point& p) {
  x += p.x;    y += p.y;
}

double Point::DistanceTo(Point sp) const {
  return sqrt((x - sp.x) * (x - sp.x) + (y - sp.y) * (y - sp.y));
}
```

// Triangle.h

```
#ifndef TRIANGLE_H
#define TRIANGLE_H
#include "Point.h"
class Triangle {
public:
  Triangle(Point, Point, Point, const char*);  // конструктор
  Triangle(const char*); // конструктор пустого (нулевого) треугольника
  Triangle(const Triangle&);  // конструктор копирования
  ~Triangle();           // деструктор
  Triangle& operator =(const Triangle&);
  static int count;      // количество созданных объектов
  char* GetName() const { return name; } // Получить имя объекта
  Point Get_v1() const { return v1; } // Получить значение v1
  Point Get_v2() const { return v2; } // Получить значение v2
  Point Get_v3() const { return v3; } // Получить значение v3
  double Area() const;       // площадь объекта
```

```
  bool operator >(const Triangle&) const;
  double Area(Point, Point, Point) const;  // площать треугольника,
                    // образованного заданными точками
  void  Show() const;              // Показать объект
  void Move(Point);
  bool Contains(Point) const;  // определяет,
                    // находится ли точка внутри треугольника
private:
  char* objID;          // идентификатор объекта
  char* name;           // наименование треугольника
  Point v1, v2, v3;      // вершины
  double a;             // сторона, соединяющая v1 и v2
  double b;             // сторона, соединяющая v2 и v3
  double c;             // сторона, соединяющая v1 и v3
  friend bool TriaInTria(Triangle, Triangle); // Определить,
                // входит ли один треугольник во второй
};
#endif /* TRIANGLE_H */
```

//Triangle.cpp

```
// Реализация класса Triangle
#include <cmath>
#include <iostream>
#include <iomanip>
#include <string>
#include "Triangle.h"
using namespace std;
Triangle::Triangle(Point _v1, Point _v2, Point _v3, const char* ident)
: v1(_v1), v2(_v2), v3(_v3) {
  char buf[16];
  objID = new char[strlen(ident) + 1];
  strcpy(objID, ident);
  count++;
  setlocale( LC_ALL, "Russian" );
  sprintf(buf, "Треугольник %d", count);
  name = new char[strlen(buf) + 1];
  strcpy(name, buf);
  a = v1.DistanceTo(v2);
  b = v2.DistanceTo(v3);
  c = v1.DistanceTo(v3);
  // отладочный вывод
  cout << "Constructor_1 for: " << objID << " (" << name << ")" << endl;
}
Triangle::Triangle(const char* ident) {
  char buf[16];
```

```
  objID = new char[strlen(ident) + 1];
  strcpy(objID, ident);
  count++;
  sprintf(buf, "Треугольник %d", count);
  name = new char[strlen(buf) + 1];
  strcpy(name, buf);
  a = b = c = 0;
  // отладочный вывод
  cout << "Constructor_2 for: " << objID << " (" << name << ")" << endl;
}
Triangle::~Triangle() {
  cout << "Destructor for: " << objID << endl;
  delete [] objID;
  delete [] name;
}
Triangle::Triangle(const Triangle& tria) : v1(tria.v1), v2(tria.v2), v3(tria.v3) {
  cout << "Copy constructor for: " << tria.objID << endl; // отладочный вывод
  objID = new char[strlen(tria.objID) + strlen("(копия)") + 1];
  strcpy(objID, tria.objID);
  strcat(objID, "(копия)");
  name = new char[strlen(tria.name) + 1];
  strcpy(name, tria.name);
  v1 = tria.v1;  v2 = tria.v2;  v3 = tria.v3;
  a = tria.a;    b = tria.b;    c = tria.c;
}
Triangle& Triangle::operator =(const Triangle& tria) {
  // отладочный вывод
  cout << "Assign operator: " << objID << " = " << tria.objID << endl;
  if (&tria == this) return *this;
  delete [] name;
  name = new char[strlen(tria.name) + 1];
  strcpy(name, tria.name);
  v1 = tria.v1;  v2 = tria.v2;  v3 = tria.v3;
  a = tria.a;  b = tria.b;  c = tria.c;
  return *this;
}
void Triangle::Show() const {
  cout << name << ":";
  v1.Show();   v2.Show();   v3.Show();
  cout << endl;
}
double Triangle::Area() const {
  double p = (a + b + c) / 2;
  return sqrt(p * (p - a) * (p - b) * (p - c));
}
```

```
bool Triangle::operator >(const Triangle& tria) const {
  if (Area() > tria.Area()) return true;
  else  return false;
}
double Triangle::Area(Point p1, Point p2, Point p3) const {
  double a1 = p1.DistanceTo(p2);
  double b1 = p2.DistanceTo(p3);
  double c1 = p1.DistanceTo(p3);
  double p = (a1 + b1 + c1) / 2;
  return  sqrt(p * (p - a1) * (p - b1) * (p - c1));
}
void Triangle::Move(Point dp) {
  v1 += dp;    v2 += dp;    v3 += dp;
}
bool Triangle::Contains(Point pt) const {
  const double calc_error = 0.001;
  double area1 = Area(pt, v1, v2);
  double area2 = Area(pt, v2, v3);
  double area3 = Area(pt, v3, v1);
  if (fabs(area1 + area2 + area3 - Area()) < calc_error)
    return true;
  else  return false;
}
bool TriaInTria(Triangle tria1, Triangle tria2) {
  Point v1 = tria1.Get_v1();
  Point v2 = tria1.Get_v2();
  Point v3 = tria1.Get_v3();
  return (tria2.Contains(v1) && tria2.Contains(v2) && tria2.Contains(v3));
}
```

// **Main.cpp**

```
#include <iostream>
#include "Triangle.h"
using namespace std;
int Menu();
int GetNumber(int, int);
void ExitBack();
void Show(Triangle* [], int);
void Move(Triangle* [], int);
void FindMax(Triangle* [], int);
void IsIncluded(Triangle* [], int);
double GetDouble();

// Инициализация глобальных переменных
int Triangle::count = 0;
```

```
//главная функция
int main() {
  // Определения точек
  Point p1(0, 0);    Point p2(0.5, 1);
  Point p3(1, 0);    Point p4(0, 4.5);
  Point p5(2, 1);    Point p6(2, 0);
  Point p7(2, 2);    Point p8(3, 0);
  // Определения треугольников
  Triangle triaA(p1, p2, p3, "triaA");
  Triangle triaB(p1, p4, p8, "triaB");
  Triangle triaC(p1, p5, p6, "triaC");
  Triangle triaD(p1, p7, p8, "triaD");
  // Определение массива указателей на треугольники
  Triangle* pTria[] = { &triaA, &triaB, &triaC, &triaD };
  int n = sizeof (pTria) / sizeof (pTria[0]);
  setlocale( LC_ALL, "Russian" );
  // Главный цикл
  bool done = false;
  while (!done) {
    switch (Menu()) {
    case 1: Show(pTria, n);
      break;
    case 2: Move(pTria, n);
      break;
    case 3: FindMax(pTria, n);
      break;
    case 4: IsIncluded(pTria, n);
      break;
    case 5: cout << "Конец работы." << endl;
      done = true;
      break;
    }
  }
  return 0;
}

//вывод меню
int Menu() {
  cout << "\n============= Г л а в н о е   м е н ю =============" << endl;
  cout << "1 - вывести все объекты\t 3 - найти максимальный" << endl;
  cout << "2 - переместить\t\t 4 - определить отношение включения" << endl;
  cout << "\t\t    5 - выход" << endl;
  return GetNumber(1, 5);
}
```

```
//ввод целого числа в заданном диапазоне
int GetNumber(int min, int max) {
  int number = min - 1;
  while (true) {
    cin >> number;
    if ((number >= min) && (number <= max) && (cin.peek() == '\n'))
      break;
    else {
      cout << "Повторите ввод (ожидается число от " << min
        << " до " << max << "):" << endl;
      cin.clear();
      while (cin.get() != '\n') {};
    }
  }
  return number;
}

// возврат в функцию с основным меню
void ExitBack() {
  cout << "Нажмите Enter." << endl;
  cin.get();  cin.get();
}

// вывод всех треугольников
void Show(Triangle* p_tria[], int k) {
  cout << "======= Перечень треугольников ========" << endl;
  for (int i = 0; i < k; ++i) p_tria[i]->Show();
  ExitBack();
}

// перемещение
void Move(Triangle* p_tria[], int k) {
  cout << "============= Перемещение ============" << endl;
  cout << "Введите номер треугольника (от 1 до " << k << "): ";
  int i = GetNumber(1, k) - 1;
  p_tria[i]->Show();
  Point dp;
  cout << "Введите смещение по x: ";
  dp.x = GetDouble();
  cout << "Введите смещение по y: ";
  dp.y = GetDouble();
  p_tria[i]->Move(dp);
  cout << "Новое положение треугольника:" << endl;
  p_tria[i]->Show();
  ExitBack();
```

```
}

// поиск максимального треугольника
void FindMax(Triangle* p_tria[], int k) {
  cout << "=== Поиск максимального треугольника ==" << endl;
  // Создаем объект triaMax, который по завершению поиска будет идентичен
    //максимальному объекту.
  // Инициализируем его значением 1-го объекта из массива объектов.
  Triangle triaMax("triaMax");
  triaMax = *p_tria[0];
  // Поиск
  for (int i = 1; i < 4; ++i)
    if (*p_tria[i] > triaMax)
      triaMax = *p_tria[i];
  cout << "Максимальный треугольник: " << triaMax.GetName() << endl;
  ExitBack();
}

// определение отношения включения
void IsIncluded(Triangle* p_tria[], int k) {
  cout << "======== Отношение включения ==========" << endl;
  cout << "Введите номер 1-го треугольника (от 1 до " << k << "): ";
  int i1 = GetNumber(1, k) - 1;
  cout << "Введите номер 2-го треугольника (от 1 до " << k << "): ";
  int i2 = GetNumber(1, k) - 1;
  if (TriaInTria(*p_tria[i1], *p_tria[i2]))
    cout << p_tria[i1]->GetName() << " - входит в - "<< p_tria[i2]->GetName()
      <<endl;
  else cout << p_tria[i1]->GetName() << " - не входит в - "
    << p_tria[i2]->GetName() << endl;
  ExitBack();
}

// ввод вещественного числа
double GetDouble() {
  double value;
  while (true) {
    cin >> value;
    if (cin.peek() == '\n') break;
    else {
      cout << "Повторите ввод (ожидается вещественное число):" << endl;
      cin.clear();
      while (cin.get() != '\n') {};
    }
  }
```

```
  return value;
}
```

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Разработать на языке C++ в соответствии со своим вариантом программу, использующую объектно-ориентированный подход.
2. Номер варианта определяется по последними двум цифрам зачётной книжки согласно табл. 8.1.

Таблица 8.1 – Формулы расчета варианта исходных данных

Последние две цифры зачетной книжки, Nзач	Номер варианта, Nвар
Nзач=от 1 до 20	Nвар=Nзач
Nзач=от 21 до 40	Nвар=Nзач-20
Nзач=от 41 до 60	Nвар=Nзач-40
Nзач=от 61 до 80	Nвар=Nзач-60
Nзач=от 81 до 99	Nвар=Nзач-80

3. В коде должны присутствовать комментарии, поясняющие алгоритм работы программы.
4. Выполненное задание предоставить в виде файла с расширением .cpp. В начале файла написать в виде комментариев: номер и вид работы, тему работы, Ф.И.О., группу, вариант задания, задание.
5. Подготовиться к защите и защитить работу у преподавателя.

ВАРИАНТЫ ИСХОДНЫХ ДАННЫХ

Вариант 1

Описать класс, реализующий стек. Написать программу, использующую этот класс для моделирования Т-образного сортировочного узла на железной дороге. Программа должна разделять на два направления состав, состоящий из вагонов двух типов (на каждое направление формируется состав из вагонов одного типа). Предусмотреть возможность формирования состава из файла и с клавиатуры.

Вариант 2

Описать класс, реализующий бинарное дерево, обладающее возможностью добавления новых элементов, удаления существующих, поиска элемента по ключу, а также последовательного доступа ко всем элементам.

Написать программу, использующую этот класс для представления англо-русского словаря. Программа должна содержать меню, позволяющее выполнить проверку всех методов класса. Предусмотреть возможность создания словаря из файла и с клавиатуры.

Вариант 3

Построить систему классов для описания плоских геометрических фигур: круг, квадрат, прямоугольник. Предусмотреть методы для создания объектов, перемещения на плоскости, изменения размеров и вращения на заданный угол.

Написать программу, демонстрирующую работу с этими классами. Программа должна содержать меню, позволяющее осуществить проверку всех методов классов.

Вариант 4

Построить описание класса, содержащего информацию о почтовом адресе организации. Предусмотреть возможность раздельного изменения составных частей адреса, создания и уничтожения объектов этого класса.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 5

Составить описание класса для представления комплексных чисел. Обеспечить выполнение операций сложения, вычитания и умножения комплексных чисел.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 6

Составить описание класса для объектов-векторов, задаваемых координатами концов в трехмерном пространстве. Обеспечить операции сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления скалярного произведения двух векторов, длины вектора, косинуса угла между векторами.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 7

Составить описание класса прямоугольников со сторонами, параллельными осям координат. Предусмотреть возможность перемещения прямоугольников на плоскости, изменение размеров, построение наименьшего прямоугольника, содержащего два заданных прямоугольника, и прямоугольника, являющегося общей частью (пересечением) двух прямоугольников.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 8

Составить описание класса для определения одномерных массивов целых чисел (векторов). Предусмотреть возможность обращения к отдельному элементу массива с контролем выхода за пределы массива, возможность задания произвольных границ индексов при создании объекта и выполнения операций поэлементного сложения и вычитания массивов с одинаковыми границами индексов, умножения и деления всех элементов массива на скаляр, вывода на экран элемента массива по заданному индексу и всего массива.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 9

Составить описание класса для определения одномерных массивов строк фиксированной длины. Предусмотреть возможность обращения к отдельным строкам массива по индексам, контроль выхода за пределы массива, выполнения операций поэлементного сцепления двух массивов с образованием нового массива, слияния двух массивов с исключением повторяющихся элементов, вывода на экран элемента массива по заданному индексу и всего массива.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 10

Составить описание класса многочленов от одной переменной, задаваемых степенью многочлена и массивом коэффициентов. Предусмотреть методы для вычисления значения многочлена для заданного аргумента, операции сложения, вычитания и умножения многочленов с получением нового объекта-многочлена, вывод на экран описания многочлена.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 11

Составить описание класса одномерных массивов строк, каждая строка задается длиной и указателем на выделенную для нее память. Предусмотреть возможность обращения к отдельным строкам массива по индексам, контроль выхода за пределы массивов, выполнения операций поэлементного сцепления двух массивов с образованием нового массива, слияния двух массивов с исключением повторяющихся элементов, вывода на экран элемента массива и всего массива.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 12

Составить описание класса, обеспечивающего представление матрицы произвольного размера с возможностью изменения числа строк и столбцов, вывод на экран подматрицы любого размера и всей матрицы.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 13

Написать класс для эффективной работы со строками, позволяющий форматировать и сравнивать строки, хранить в строках числовые значения и извлекать их. Для этого необходимо реализовать:

- ❒ перегруженные операции присваивания и конкатенации;
- ❒ операции сравнения и приведения типов;
- ❒ преобразование в число любого типа;
- ❒ форматный вывод строки.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 14

Описать класс «домашняя библиотека». Предусмотреть возможность работы с произвольным числом книг, поиска книги по какому-либо признаку (например, по автору или по году издания), добавления книг в библиотеку, удаления книг из нее, сортировки книг по разным полям.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 15

Описать класс «записная книжка». Предусмотреть возможность работы с произвольным числом записей, поиска записи по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона), добавления и удаления записей, сортировки по разным полям.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 16

Описать класс «студенческая группа». Предусмотреть возможность работы с переменным числом студентов, поиска студента по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона), добавления и удаления записей, сортировки по разным полям.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 17

Описать класс, реализующий тип данных «вещественная матрица» и работу с ними. Класс должен реализовывать следующие операции над матрицами:

- ❒ сложение, вычитание, умножение, деление (+, –, *, /) (умножение и деление как на другую матрицу, так и на число);
- ❒ комбинированные операции присваивания (+=, –=, *=, /=);
- ❒ операции сравнения на равенство (неравенство);
- ❒ операции вычисления обратной и транспонированной матрицы, операцию возведения в степень;
- ❒ методы вычисления детерминанта и нормы;
- ❒ методы, реализующие проверку типа матрицы (квадратная, диагональная, нулевая, единичная, симметрическая, верхняя треугольная, нижняя треугольная);
- ❒ операции ввода-вывода в стандартные потоки.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 18

Описать класс «множество», позволяющий выполнять основные операции - добавление и удаление элемента, пересечение, объединение и разность множеств.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 19

Описать класс, реализующий стек. Написать программу, использующую этот класс для отыскания прохода по лабиринту. Лабиринт представляется в виде матрицы, состоящей из квадратов. Каждый квадрат либо открыт, либо закрыт. Вход в закрытый квадрат запрещен. Если квадрат открыт, то вход в него возможен со стороны, но не с угла. Каждый квадрат определяется его координатами в матрице. После отыскания прохода программа печатает найденный путь в виде координат квадратов.

Вариант 20

Описать класс «предметный указатель». Каждый компонент указателя содержит слово и номера страниц, на которых это слово встречается. Количество номеров страниц, относящихся к одному слову, от одного до десяти. Предусмотреть возможность формирования указателя с клавиатуры и

из файла, вывода указателя, вывода номеров страниц для заданного слова, удаления элемента из указателя.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое класс?
2. Как описывается класс?
3. Что входит в состав класса?
4. Каким образом и для чего создаются экземпляры класса?
5. Дайте определение и перечислите свойства конструкторов.
6. Для чего используются статические элементы класса?
7. Зачем применяется механизм дружественных функций и классов?