Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий

Кафедра «МОП ЭВМ»

Курсовая работа

по дисциплине «?»

Вариант 12

Студент группы 6ИСб-1 И. Нозимзода

Преподаватель С.Ю. Александров

2018

Содержание

[Содержание 2](#_Toc9287451)

[Задания 4](#_Toc9287452)

[Введение 5](#_Toc9287453)

[1 Классы 6](#_Toc9287454)

[1.1 Описание программы 6](#_Toc9287455)

[1.2 Текст программы 6](#_Toc9287456)

[1.3 Тестирование программы 10](#_Toc9287457)

[2 Наследование 12](#_Toc9287458)

[2.1 Описание программы 12](#_Toc9287459)

[2.2 Текст программы 12](#_Toc9287460)

[2.3 Тестирование программы 21](#_Toc9287461)

[3 Шаблоны классов. Обработка исключительных ситуаций 23](#_Toc9287462)

[3.1 Описание программы 23](#_Toc9287463)

[3.2 Текст программы 23](#_Toc9287464)

[3.3 Тестирование программы 25](#_Toc9287465)

[4 Стандартные потоки 27](#_Toc9287466)

[4.1 Описание программы 27](#_Toc9287467)

[4.2 Текст программы 27](#_Toc9287468)

[4.3 Тестирование программы 29](#_Toc9287469)

[5 Файловые и строковые потоки. Строки класса string. 31](#_Toc9287470)

[5.1 Описание программы 31](#_Toc9287471)

[5.2 Текст программы 31](#_Toc9287472)

[5.3 Тестирование программы 32](#_Toc9287473)

[6 Стандартная библиотека шаблонов 33](#_Toc9287474)

[6.1 Описание программы 33](#_Toc9287475)

[6.2 Текст программы 33](#_Toc9287476)

[6.3 Тестирование программы 37](#_Toc9287477)

[Заключение 39](#_Toc9287478)

[Список использованных источников 40](#_Toc9287479)

Задания

1. Классы.
2. Наследование.
3. Шаблоны классов. Обработка исключительных ситуаций
4. Стандартные потоки.
5. Файловые и строковые потоки. Строки класса string.
6. Стандартная библиотека шаблонов.

Введение

Язык С++ как средство обучения программированию обладает рядом несомненных достоинств. Он хорошо организован, строг, большинство его конструкций логичны и удобны. Развитые средства диагностики и редактирования кода делают процесс программирования приятным и эффективным.

Немаловажно, что С++ является не учебным, а профессиональным языком, предназначенным для решения широкого спектра задач, и в первую очередь - в быстро развивающейся области создания распределенных приложений.

# Классы

Классы и объекты в С++ являются основными концепциями объектно-ориентированного программирования — ООП. Объектно-ориентированное программирование — расширение структурного программирования, в котором основными концепциями являются понятия классов и объектов.

## Описание программы

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.

## Текст программы

Проект состоит из одного файла исходного кода, который приведен в листине 1.1.

Листинг . – Текст файла Lab10.cpp

|  |
| --- |
| #include <iostream>  using namespace std;  class Matrix // целочисленная матрица  {  private:  int\*\* a; // для хранения информации  int w, h;// кол-вл строк и столбцов  public:  Matrix(int H, int W);  Matrix();  ~Matrix();  void SetXY(int Y, int X, int data); // для заполнения ячейки строки y и столбца x  int GetXY(int Y, int X); // возвращает значение элемента строки y и столбца x  void Show();  void Show(int Hbeg, int Wbeg, int Hend, int Wend); // вывод начиная со строки HBeg и заканчивая Hend, и столбца с WBeg до Wend  void SetH(int H);  void SetW(int W);  int GetH() { return h; }  int GetW() { return w; }  };  Matrix::Matrix()  {  h = w = 0;  }  Matrix::Matrix(int H, int W)  {  h = H;  w = W;  a = new int\* [H];// выделение памяти под строки  for (int i = 0; i < H; i++)  a[i] = new int[W];// выделение памяти под каждую строку  }  Matrix::~Matrix()  {  for (int i = 0; i < h; i++)  delete[] a[i];  delete[] a;  }  void Matrix::SetXY(int Y, int X, int data)  {  if (Y < h && Y >= 0 && X < w && X >= 0)  a[Y][X] = data;  }  int Matrix::GetXY(int Y, int X)  {  if (Y < h && Y >= 0 && X < w && X >= 0)  return a[Y][X];  else  return -1;  }  void Matrix::Show()  {  for (int i = 0; i < h; i++, cout << endl)  for (int j = 0; j < w; j++)  cout << a[i][j] << "\t";  }  void Matrix::Show(int Hbeg, int Hend, int Wbeg, int Wend)  {  if (Hend < h && Hbeg >= 0 && Wend < w && Wbeg >= 0)  {  for (int i = Hbeg; i <= Hend; i++, cout << endl)  for (int j = Wbeg; j <= Wend; j++)  cout << a[i][j] << " ";  }  }  void Matrix::SetH(int H)  {  int\*\* b;// новый участок памяти  b = new int\* [H];// выделение памяти под строки  for (int i = 0; i < H; i++)  {  b[i] = new int[w];// выделение памяти под каждую строку  for (int j = 0; j < w; j++)  b[i][j] = 0; // сразу обнуляем строки  }  for (int i = 0; i < H && i < h; i++)  for (int j = 0; j < w; j++)  b[i][j] = a[i][j]; // копируем информацию  // устанавливаем новое кол-во строк  for (int i = 0; i < h; i++)  delete[] a[i]; // очистка памяти из под старой информации  delete[] a;  h = H;  a = b;  }  void Matrix::SetW(int W)  {  int\*\* b;  b = new int\* [h];// выделение памяти под строки  for (int i = 0; i < h; i++)  {  b[i] = new int[W];// выделение памяти под каждую строку  for (int j = 0; j < W; j++)  b[i][j] = 0; // сразу обнуляем строки  }  for (int i = 0; i < h; i++)  for (int j = 0; j < w && j < W; j++)  b[i][j] = a[i][j]; // копируем информацию  for (int i = 0; i < h; i++) // очистка памяти из под старой информации  delete[] a[i];  delete[] a;  w = W; // устанавливаем новое кол-во столбцов  a = b;  }  int main()  {  setlocale(LC\_ALL, "rus");  cout << "Класс Matrix" << endl;  cout << "Начальная инициализация размера матрицы" << endl;  cout << "Высота (количество строк): ";  int height;  cin >> height;  cout << "Ширина (количество столбцов): ";  int width;  cin >> width;  Matrix matrix(height, width);  for (int i = 0; i < height; i++) {  for (int j = 0; j < width; j++) {  int value = i \* 5 + j;  matrix.SetXY(i, j, value);  }  }  cout << "Матрица создана и заполнена произвольными значениями" << endl;  system("pause");  system("cls");  while (true)  {  cout << endl;  cout << "Класс Matrix (" << matrix.GetH() << "x" << matrix.GetW() << ")" << endl;  cout << "1.Заполнение матрицы новыми значениями" << endl;  cout << "2.Вывод матрицы" << endl;  cout << "3.Установка высоты (количества строк) матрицы" << endl;  cout << "4.Установка ширины (количества столбцов) матрицы" << endl;  cout << "5.Вывод подматрицы" << endl;  cout << "6.Очистка консоли" << endl;  cout << ">";  int select;  cin >> select;  switch (select)  {  case 1:  {  for (int i = 0; i < height; i++) {  for (int j = 0; j < width; j++) {  cout << "Элемент [" << i+1 << "]" << "[" << j + 1 << "]: ";  int value;  cin >> value;  matrix.SetXY(i, j, value);  }  }  }  break;  case 2:  {  cout << endl << "Матрица:" << endl;  matrix.Show();  }  break;  case 3:  {  cout << "Введите новую высоту: ";  int newH;;  cin >> newH;  matrix.SetH(newH);  }  break;  case 4:  {  cout << "Введите новую ширину: ";  int newW;;  cin >> newW;  matrix.SetH(newW);  }  break;  case 5:  {  cout << "Введите начальную строку: ";  int Hbeg;;  cin >> Hbeg;  cout << "Введите конечную строку: ";  int Hend;  cin >> Hend;  cout << "Введите начальный столбец: ";  int Wbeg;;  cin >> Wbeg;  cout << "Введите конечный столбец: ";  int Wend;  cin >> Wend;  cout << endl << "Подматрица:" << endl;  matrix.Show(Hbeg, Hend, Wbeg, Wend);  }  break;  case 6:  system("cls");  break;  default:  system("cls");  break;}  }  return 0;  } |

## Тестирование программы

Результаты работы программы приведены на рисунках 1.1,1.2.

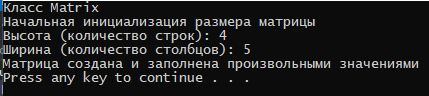


Рисунок . – Результат работы программы

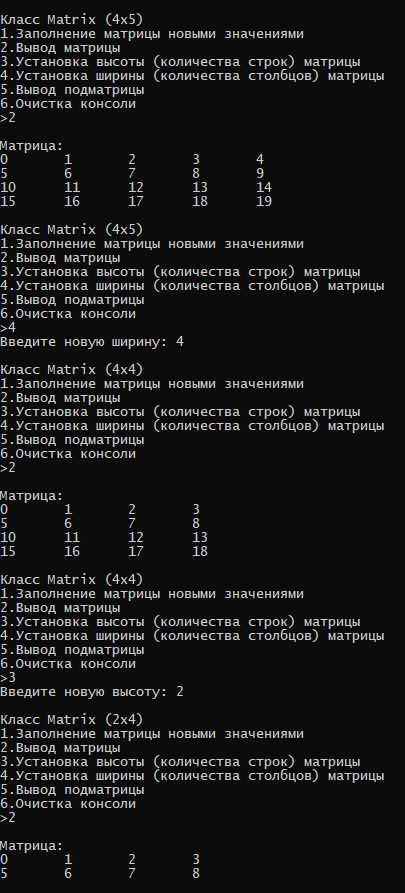


Рисунок .2 – Результат работы программы

# Наследование

Классы в объектно-ориентированных программах используются для моделирования концепций реального и программного мира. Концепции или сущности предметной области находятся в различных взаимоотношениях. Одно из таких взаимоотношений ‑ отношение наследования (именуемое также отношением родитель потомок или отношением обобщение–специализация).

## Описание программы

*Написать программу, демонстрирующую работу с объектами двух типов, Quadrate и Pentagon, для чего создать систему соответствующих классов. Каждый объект должен иметь идентификатор (в виде произвольной строки символов) и одно или несколько полей для хранения состояния (текущего значения) объекта.*

*Клиенту (функции main) должны быть доступны следующие основные операции (методы): создать объект, удалить объект, показать значение объекта, Move(), IsIntersect(T&, T&). Операции по созданию и удалению объектов инкапсулировать в классе Factory.*

*Предусмотреть меню, позволяющее продемонстрировать заданные операции. При необходимости в разрабатываемые классы добавляются дополнительные методы (например, конструктор копирования, операция присваивания и т.п.) для обеспечения надлежащего функционирования этих классов.*

## Текст программы

Проект состоит из 5 файлов исходного кода, которые приведен в листингах 2.1-2.5.

Листинг . – Текст файла Lab11.cpp

|  |
| --- |
| #include "Factory.h"  int main()  {  setlocale(LC\_ALL, "rus");  cout << "Демонстрация возможностей классов Quadrate(квадрат) и Pentagon(пятиугольник)" << endl;  char Control;  while (true)  {  cout << endl << "Интерфейс программы:" << endl;  cout << "1. Объект Quadrate (квадрат)" << endl;  cout << "2. Объект Pentagon (пятиугольник)" << endl;  cout << ">";  cin >> Control;  switch (Control)  {  case '1':  {  FactoryQuadrate factoryQuadrate;  cout << "Создание квадрата с координатами по умолчанию.." << endl;  bool breakCheck = true;  while (breakCheck)  {  cout << "\nQuadrate (квадрат)" << endl;  cout << "1. Создать квадрат с новыми координатами" << endl;  cout << "2. Удалить квадрат" << endl;  cout << "3. Показать координаты квадрата" << endl;  cout << "4. Переместить квадрат" << endl;  cout << "5. Проверка на пересечение" << endl;  cout << "6. Вернуться к выбору фигуры" << endl;  cout << ">";  cin >> Control;  switch (Control)  {  case '1':  {  factoryQuadrate.addObject();  }  break;  case '2':  {  factoryQuadrate.deleteObject();  }  break;  case '3':  {  factoryQuadrate.quadrate\_.Result();  }  break;  case '4':  {  factoryQuadrate.quadrate\_.Move();  }  break;  case '5':  {  factoryQuadrate.quadrate\_.IsIntersect();  }  break;  case '6':  breakCheck = false;  break;  default: continue;  }  }  }  break;  case '2':  {  FactoryPentagon factoryPentagon;  bool breakCheck = true;  while (breakCheck)  {  cout << "\nPentagon (пятиугольник)" << endl;  cout << "1. Создать пятиугольник с новыми координатами" << endl;  cout << "2. Удалить пятиугольник" << endl;  cout << "3. Показать координаты пятиугольника" << endl;  cout << "4. Переместить пятиугольник" << endl;  cout << "5. Проверка на пересечение" << endl;  cout << "6. Вернуться к выбору фигуры" << endl;  cout << ">";  cin >> Control;  switch (Control)  {  case '1':  {  factoryPentagon.addObject();  }  break;  case '2':  {  factoryPentagon.deleteObject();  }  break;  case '3':  {  factoryPentagon.pentagon\_.Result();  }  break;  case '4':  {  factoryPentagon.pentagon\_.Move();  }  break;  case '5':  {  factoryPentagon.pentagon\_.IsIntersect();  }  break;  case '6':  breakCheck = false;  break;  default: continue;  }  }  }  break;  }  }  return 0;  } |

Листинг .2– Текст файла Factory.h

|  |
| --- |
| #include "Figures.h"  class FactoryQuadrate {  public:  FactoryQuadrate() {}  void addObject();  void deleteObject();  Quadrate quadrate\_;  };  class FactoryPentagon {  public:  FactoryPentagon() {}  void addObject();  void deleteObject();  Pentagon pentagon\_;  }; |

Листинг .3 – Текст файла Factory.cpp

|  |
| --- |
| #include "Factory.h"  void FactoryQuadrate::addObject() {  quadrate\_.Input();  }  void FactoryQuadrate::deleteObject() {  quadrate\_.Delete();  }  void FactoryPentagon::addObject() {  pentagon\_.Input();  }  void FactoryPentagon::deleteObject() {  pentagon\_.Delete();  } |

Листинг .4 – Текст файла Figures.h

|  |
| --- |
| #include <iostream>  #include <string>  #include <cstdlib>  using namespace std;  class Tetragon  {  protected:  double x1, x2, y1, y2;  size\_t ID\_SEQUENCE\_;  bool deleted\_;  public:  Tetragon(double a, double b, double c, double d) : x1(a), x2(b), y1(c), y2(d), ID\_SEQUENCE\_(rand()), deleted\_(false) {};  Tetragon() : x1(0), x2(5), y1(0), y2(5), ID\_SEQUENCE\_(rand()), deleted\_(false) {};  ~Tetragon() { }  virtual void Input(); // метод ввода с клавиатуры  void Delete();  void Result(); // метод вывода на экран  void Move(); // перемещение  void IsIntersect(); // пересечение 2 прямоугольников  };  class Quadrate : public Tetragon  {  public:  void Input();  };  class Pentagon  {  protected:  double x1, x2, x3, x4, x5, y1, y2, y3, y4, y5;  size\_t ID\_SEQUENCE\_;  bool deleted\_;  public:  Pentagon() : x1(0), y1(0), x2(1), y2(-1), x3(2), y3(1), x4(2), y4(2), x5(1), y5(0), ID\_SEQUENCE\_(rand()), deleted\_(false) {};  ~Pentagon() { }  virtual void Input(); // метод ввода с клавиатуры  void Delete();  void Result(); // метод вывода на экран  void Move(); // перемещение  void IsIntersect(); // пересечение 2 прямоугольников  }; |

Листинг .5 – Текст файла Figures.cpp

|  |
| --- |
| #include "Figures.h"  #include <list>  #include <algorithm>  struct xy {  double x, y;  };  struct line {  xy a;  xy b;  };  inline int area(xy a, xy b, xy c) {  return (b.x - a.x) \* (c.y - a.y) - (b.y - a.y) \* (c.x - a.x);  }  inline bool intersect\_1(int a, int b, int c, int d) {  if (a > b) swap(a, b);  if (c > d) swap(c, d);  return max(a, c) <= min(b, d);  }  bool intersect(xy a, xy b, xy c, xy d) {  return intersect\_1(a.x, b.x, c.x, d.x)  && intersect\_1(a.y, b.y, c.y, d.y)  && area(a, b, c) \* area(a, b, d) <= 0  && area(c, d, a) \* area(c, d, b) <= 0;  }  void Tetragon::Input() {  cout << "Введите координату х левых точек" << endl; cin >> x1;  cout << "Введите координату х правых точек" << endl; cin >> x2;  cout << "Введите координату y верхних точек" << endl; cin >> y1;  cout << "Введите координату y нижних точек" << endl; cin >> y2;  ID\_SEQUENCE\_ = rand();  deleted\_ = false;  }  void Tetragon::Delete()  {  if (deleted\_)  cout << "Квадрат уже удалён!" << endl;  else  {  cout << "Квадрат удалён" << endl;  deleted\_ = true;  }  }  void Tetragon::Result() {  if (!deleted\_)  {  cout << "Идентификатор: ID" << ID\_SEQUENCE\_ << endl;  cout << "Координаты:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 << "," << y1 << ")" << endl;  cout << " C:" << "(" << x2 << "," << y2 << ")" << endl;  cout << " D:" << "(" << x1 << "," << y2 << ")" << endl;  }  else cout << "Сначала создайте квадрат" << endl;  }  void Tetragon::Move() {  if (!deleted\_)  {  cout << " На сколько единиц вы хотите переместить квадрат по оси X ?" << endl;  double dx;  cin >> dx;  cout << " На сколько единиц вы хотите переместить квадрат по оси Y ?" << endl;  double dy;  cin >> dy;  x1 += dx; x2 += dx;  y1 += dy; y2 += dy;  cout << "Квадрат перемещён" << endl;  cout << "Текущие координаты:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 << "," << y1 << ")" << endl;  cout << " C:" << "(" << x2 << "," << y2 << ")" << endl;  cout << " D:" << "(" << x1 << "," << y2 << ")" << endl;  }  else cout << "Сначала создайте квадрат" << endl;  }  void Tetragon::IsIntersect() {  if (!deleted\_)  {  double x3, x4, y3, y4;  cout << "Координаты второго квадрата:" << endl;  cout << "Введите координату х левой нижней точки квадрата" << endl;  cin >> x3;  cout << "Введите координату y левой нижней точки квадрата" << endl;  cin >> y3;  cout << "Введите ширину (и одновременно высоту) квадрата" << endl;  double tempW;  cin >> tempW;  x4 = x3 + tempW;  y4 = y3 + tempW;  cout << endl;  cout << "Вы ввели следующие координаты 1 квадрата:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 << "," << y1 << ")" << endl;  cout << " C:" << "(" << x2 << "," << y2 << ")" << endl;  cout << " D:" << "(" << x1 << "," << y2 << ")" << endl;  cout << endl;  cout << "Вы ввели следующие координаты 2 квадрата:" << endl;  cout << " A:" << "(" << x3 << "," << y3 << ")" << endl;  cout << " B:" << "(" << x4 << "," << y3 << ")" << endl;  cout << " C:" << "(" << x4 << "," << y4 << ")" << endl;  cout << " D:" << "(" << x3 << "," << y4 << ")" << endl;  cout << endl;    if ((y1 < y4 && y2 > y3) || (x2 < x3 && x1 > x4))  {  cout << "Квадраты пересекаются" << endl;  }  else cout << "Квадраты не пересекаются" << endl;  }  else cout << "Сначала создайте квадрат" << endl;  }  void Quadrate::Input()  {  cout << "Введите координату х левой нижней точки квадрата" << endl;  cin >> x1;  cout << "Введите координату y левой нижней точки квадрата" << endl;  cin >> y1;  cout << "Введите ширину (и одновременно высоту) квадрата" << endl;  double tempW;  cin >> tempW;  x2 = x1 + tempW;  y2 = y1 + tempW;  deleted\_ = false;  }  void Pentagon::Input() {  cout << "Введите координату точки х1: "; cin >> x1;  cout << "Введите координату точки y1: "; cin >> y1; cout << endl;  cout << "Введите координату точки х2: "; cin >> x2;  cout << "Введите координату точки y2: "; cin >> y2; cout << endl;  cout << "Введите координату точки х3: "; cin >> x3;  cout << "Введите координату точки y3: "; cin >> y3; cout << endl;  cout << "Введите координату точки х4: "; cin >> x4;  cout << "Введите координату точки y4: "; cin >> y4; cout << endl;  cout << "Введите координату точки х5: "; cin >> x5;  cout << "Введите координату точки y5: "; cin >> y5; cout << endl;  ID\_SEQUENCE\_ = rand();  deleted\_ = false;  }  void Pentagon::Delete()  {  if (deleted\_)  cout << "Пятиугольник уже удалён!" << endl;  else  {  cout << "Пятиугольник удалён" << endl;  deleted\_ = true;  }  }  void Pentagon::Result() {  if (!deleted\_)  {  cout << "Идентификатор: ID" << ID\_SEQUENCE\_ << endl;  cout << "Координаты:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 << "," << y2 << ")" << endl;  cout << " C:" << "(" << x3 << "," << y3 << ")" << endl;  cout << " D:" << "(" << x4 << "," << y4 << ")" << endl;  cout << " F:" << "(" << x5 << "," << y5 << ")" << endl;  }  else cout << "Сначала создайте пятиугольник" << endl;  }  void Pentagon::Move() {  if (!deleted\_)  {  cout << " На сколько единиц вы хотите переместить пятиугольник по оси X ?" << endl;  double dx;  cin >> dx;  cout << " На сколько единиц вы хотите переместить пятиугольник по оси Y ?" << endl;  double dy;  cin >> dy;  x1 += dx; x2 += dx; x3 += dx; x4 += dx; x5 += dx;  y1 += dy; y2 += dy; y3 += dy; y4 += dy; y5 += dy;  cout << "Пятиугольник перемещён" << endl;  cout << "Текущие координаты:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 << "," << y2 << ")" << endl;  cout << " C:" << "(" << x3 << "," << y3 << ")" << endl;  cout << " D:" << "(" << x4 << "," << y4 << ")" << endl;  cout << " F:" << "(" << x5 << "," << y5 << ")" << endl;  }  else cout << "Сначала создайте пятиугольник" << endl;  }  void Pentagon::IsIntersect() {  if (!deleted\_)  {  double x11, x22, x33, x44, x55, y11, y22, y33, y44, y55;  cout << "Координаты второго квадрата:" << endl;  cout << "Введите координату точки х1: "; cin >> x11;  cout << "Введите координату точки y1: "; cin >> y11; cout << endl;  cout << "Введите координату точки х2: "; cin >> x22;  cout << "Введите координату точки y2: "; cin >> y22; cout << endl;  cout << "Введите координату точки х3: "; cin >> x33;  cout << "Введите координату точки y3: "; cin >> y33; cout << endl;  cout << "Введите координату точки х4: "; cin >> x44;  cout << "Введите координату точки y4: "; cin >> y44; cout << endl;  cout << "Введите координату точки х5: "; cin >> x55;  cout << "Введите координату точки y5: "; cin >> y55; cout << endl;  cout << endl;  cout << "Вы ввели следующие координаты 1 пятиугольника:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 << "," << y2 << ")" << endl;  cout << " C:" << "(" << x3 << "," << y3 << ")" << endl;  cout << " D:" << "(" << x4 << "," << y4 << ")" << endl;  cout << " F:" << "(" << x5 << "," << y5 << ")" << endl;  cout << endl;  cout << "Вы ввели следующие координаты 2 пятиугольника:" << endl;  cout << " A:" << "(" << x11 << "," << y11 << ")" << endl;  cout << " B:" << "(" << x22 << "," << y22 << ")" << endl;  cout << " C:" << "(" << x33 << "," << y33 << ")" << endl;  cout << " D:" << "(" << x44 << "," << y44 << ")" << endl;  cout << " F:" << "(" << x55 << "," << y55 << ")" << endl;  cout << endl;  list<line> lines1;  xy xy1\_b; xy1\_b.x = x1; xy1\_b.y = y1;  xy xy1\_e; xy1\_e.x = x2; xy1\_e.y = y2;  line ln1; ln1.a = xy1\_b; ln1.b = xy1\_e;  xy xy2\_b; xy2\_b.x = x2; xy2\_b.y = y2;  xy xy2\_e; xy2\_e.x = x3; xy2\_e.y = y3;  line ln2; ln2.a = xy2\_b; ln2.b = xy2\_e;  xy xy3\_b; xy3\_b.x = x3; xy3\_b.y = y3;  xy xy3\_e; xy3\_e.x = x4; xy3\_e.y = y4;  line ln3; ln3.a = xy3\_b; ln3.b = xy3\_e;  xy xy4\_b; xy4\_b.x = x4; xy4\_b.y = y4;  xy xy4\_e; xy4\_e.x = x5; xy4\_e.y = y5;  line ln4; ln4.a = xy4\_b; ln4.b = xy4\_e;  xy xy5\_b; xy5\_b.x = x5; xy5\_b.y = y5;  xy xy5\_e; xy5\_e.x = x1; xy5\_e.y = y1;  line ln5; ln5.a = xy5\_b; ln5.b = xy5\_e;  lines1.push\_front(ln1);  lines1.push\_front(ln2);  lines1.push\_front(ln3);  lines1.push\_front(ln4);  lines1.push\_front(ln5);  list<line> lines2;  xy xy11\_b; xy11\_b.x = x11; xy11\_b.y = y11;  xy xy11\_e; xy11\_e.x = x22; xy11\_e.y = y22;  line ln11; ln11.a = xy11\_b; ln11.b = xy11\_e;  xy xy22\_b; xy22\_b.x = x22; xy22\_b.y = y22;  xy xy22\_e; xy22\_e.x = x33; xy22\_e.y = y33;  line ln22; ln22.a = xy22\_b; ln22.b = xy22\_e;  xy xy33\_b; xy33\_b.x = x33; xy33\_b.y = y33;  xy xy33\_e; xy33\_e.x = x44; xy33\_e.y = y44;  line ln33; ln33.a = xy33\_b; ln33.b = xy33\_e;  xy xy44\_b; xy44\_b.x = x44; xy44\_b.y = y44;  xy xy44\_e; xy44\_e.x = x55; xy44\_e.y = y55;  line ln44; ln44.a = xy44\_b; ln44.b = xy44\_e;  xy xy55\_b; xy55\_b.x = x55; xy55\_b.y = y55;  xy xy55\_e; xy55\_e.x = x11; xy55\_e.y = y11;  line ln55; ln55.a = xy55\_b; ln55.b = xy55\_e;  lines2.push\_front(ln11);  lines2.push\_front(ln22);  lines2.push\_front(ln33);  lines2.push\_front(ln44);  lines2.push\_front(ln55);  bool ex = false;  for (auto pair : lines1)  {  for (auto pair2 : lines2)  {  if (intersect(pair.a, pair.b, pair2.a, pair2.b))  {  ex = true;  }  }  }  if (ex)  {  cout << "Пятиугольники пересекаются" << endl;  }  else cout << "Пятиугольники не пересекаются" << endl;  }  else cout << "Сначала создайте пятиугольник" << endl;  } |

## Тестирование программы

Результат работы программы приведен на рисунке 2.1,2.2.

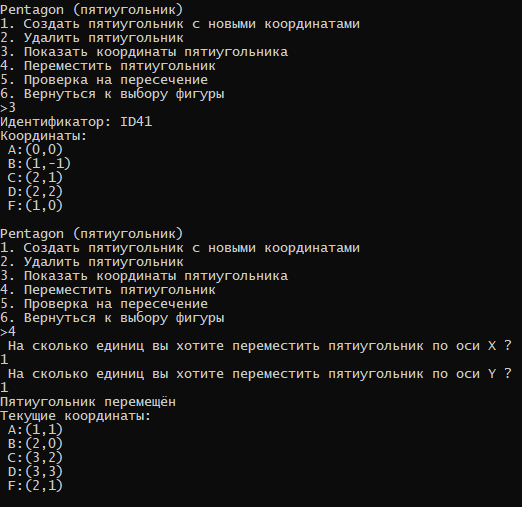


Рисунок .1 – Результат работы программы по наследованию

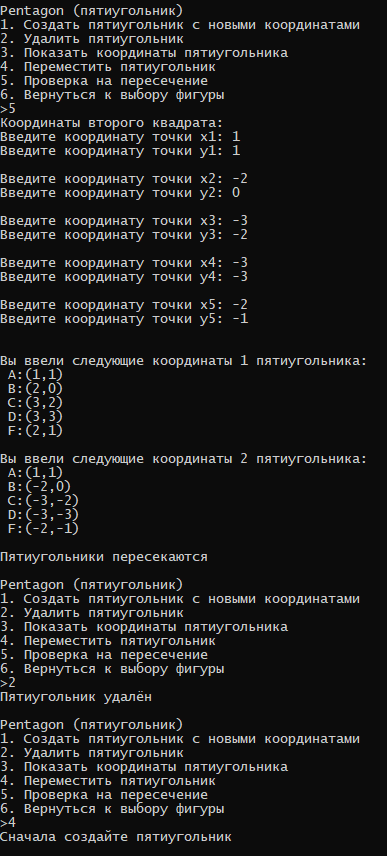


Рисунок .2 – Результат работы программы по наследованию

# Шаблоны классов. Обработка исключительных ситуаций

Шаблоны классов наряду с шаблонами функций поддерживают парадигму обобщенного программирования, то есть программирования с использованием типов в качестве параметров. Механизм шаблонов в C++ допускает применение абстрактного типа в качестве параметра при определении класса или функции. После того как шаблон класса определен, он может использоваться для определения конкретных классов.

## Описание программы

*Требуется создать шаблон некоторого целевого класса Deque.*

*В каждом варианте уточняются требования к реализации — указанием на применение некоторого серверного класса Vect. Это означает, что объект класса Vect используется как элемент класса Deque. В качестве серверного класса может быть указан либо класс, созданный программистом в рамках того же задания, либо класс стандартной библиотеки. Deque - Двусторонняя очередь (допускает вставку и удаление из обоих концов);*

## Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 3.1.

Листинг 3. – Текст файла Lab12.cpp

|  |
| --- |
| #include "Deque.h"  #include "string"  int main()  {  setlocale(LC\_ALL, "rus");  cout << "Deque - Двусторонняя очередь(допускает вставку и удаление из обоих концов)" << endl;  cout << "Демонстрация интерфейса целевого шаблонного класса Deque, основанном на применении серверного класса Vect, со значениями разных типов" << endl << endl;  try  {  Deque<int> dequeInt;  cout << "Добавление значений типа int от 0 до 1 в список" << endl;  for (int i = 0; i < 2; ++i)  dequeInt.push\_back(i);  dequeInt.show();  cout << "Добавление значений типа int от 0 до 1 в начало списка" << endl;  for (int i = 0; i < 2; ++i)  dequeInt.push\_begin(dequeInt.begin(),i);  dequeInt.show();  cout << "Удаление одного элемента с начала и конца очереди" << endl;  dequeInt.pop\_back();  dequeInt.pop\_begin();  dequeInt.show();  cout << "Удаление одного элемента с начала и конца очереди" << endl;  dequeInt.pop\_back();  dequeInt.pop\_begin();  dequeInt.show();  cout << "Удаление одного элемента с начала очереди" << endl;  dequeInt.pop\_begin();  dequeInt.show();  cout << endl << endl;  }  catch (underflow\_error& e)  {  cerr << e.what() << endl;  }  try  {  Deque<double> dequeInt;  cout << "Добавление значений типа double от 0.5 до 1.5 в список" << endl;  for (double i = 0.5; i < 2.5; ++i)  dequeInt.push\_back(i);  dequeInt.show();  cout << "Добавление значений типа double от 0.5 до 1.5 в начало списка" << endl;  for (int i = 0.5; i < 2.5; ++i)  dequeInt.push\_begin(dequeInt.begin(), i);  dequeInt.show();  cout << "Удаление одного элемента с начала и конца очереди" << endl;  dequeInt.pop\_back();  dequeInt.pop\_begin();  dequeInt.show();  cout << "Удаление одного элемента с начала и конца очереди" << endl;  dequeInt.pop\_back();  dequeInt.pop\_begin();  dequeInt.show();  cout << "Удаление одного элемента с начала очереди" << endl;  dequeInt.pop\_begin();  dequeInt.show();  }  catch (underflow\_error& e)  {  cerr << e.what() << endl;  }  try  {  Deque<string> dequeInt;  cout << "Добавление значений типа string от '0' до '1' в список" << endl;  for (int i = 0; i < 2; ++i)  dequeInt.push\_back("'" + to\_string(i) + "'");  dequeInt.show();  cout << "Добавление значений типа string от '0' до '1' в начало списка" << endl;  for (int i = 0; i < 2; ++i)  dequeInt.push\_begin(dequeInt.begin(), "'" + to\_string(i) + "'");  dequeInt.show();  cout << "Удаление одного элемента с начала и конца очереди" << endl;  dequeInt.pop\_back();  dequeInt.pop\_begin();  dequeInt.show();  cout << "Удаление одного элемента с начала и конца очереди" << endl;  dequeInt.pop\_back();  dequeInt.pop\_begin();  dequeInt.show();  cout << "Удаление одного элемента с начала очереди" << endl;  dequeInt.pop\_begin();  dequeInt.show();  cout << endl << endl;  }  catch (underflow\_error& e)  {  cerr << e.what() << endl;  }  cout << endl;  system("pause");  return 0;  } |

## Тестирование программы

Результат работы программы приведен на рисунке 3.1.

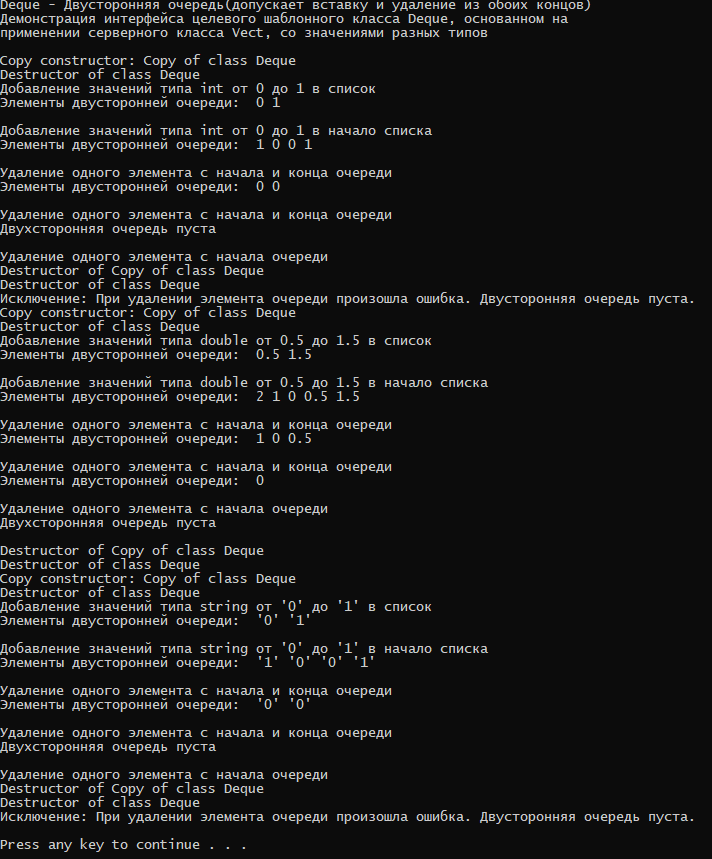


Рисунок 3.1 – Результат работы программы по шаблонам классов

# Стандартные потоки

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 ввода вывода. Класс streambuf обеспечивает общие средства управления буферами потоков и их взаимодействие с физическими устройствами, являясь родительским для других буферных классов.

## Описание программы

*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.*

*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.*

## Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 4.1.

Листинг 4. – Текст файла Lab13.cpp

|  |
| --- |
| #include<iostream>  #include<vector>  #include<string>  #include <algorithm>  using namespace std;  class Note {  public:  Note() {}  Note(string fio, string phoneNumber, string birthday0, string birthday1, string birthday2) {  fio\_ = fio;  phoneNumber\_ = phoneNumber;  birthday\_[0] = birthday0;  birthday\_[1] = birthday1;  birthday\_[2] = birthday2;  }  friend ostream& operator<<(ostream& out, const Note& rv);  friend istream& operator>>(istream& in, Note& rv);  friend class Notebook;  private:  string fio\_;  string phoneNumber\_;  string birthday\_[3];  };  ostream& operator<<(ostream& out, const Note& rv) {  cout << "Фамилия и имя: " << rv.fio\_ << endl;  cout << "Номер телефона: " << rv.phoneNumber\_ << endl;  cout << "День рождения (массив из трех чисел ГГ.ММ.ДД): " << rv.birthday\_[0] << "." << rv.birthday\_[1] << "." << rv.birthday\_[2] << endl;  return out;  }  istream& operator>>(istream& in, Note& rv) {  cout << "Введите фамилию и имя: ";  getline(cin, rv.fio\_);  cout << "Введите номер телефона: ";  getline(cin, rv.phoneNumber\_);  cout << "Введите день рождения (массив из трех чисел ГГ.ММ.ДД)" << endl;  cout << "Введите год: ";  getline(cin, rv.birthday\_[0]);  cout << "Введите месяц: ";  getline(cin, rv.birthday\_[1]);  cout << "Введите день: ";  getline(cin, rv.birthday\_[2]);  return in;  }  class Notebook {  public:  Notebook(int count) {  for (size\_t i = 0; i < count; i++)  addNote();  }  void addNote() {  Note note;  cin >> note;  cout << endl;  notebook\_.push\_back(note);  }  void printNotebook() {  for (auto note : notebook\_)  {  cout << note;  cout << endl;  }  }  void sortNotebook() {  sort(  notebook\_.begin(),  notebook\_.end(),  [](Note& a, Note& b) {  if (a.birthday\_[0] == b.birthday\_[0])  {  if (a.birthday\_[1] == b.birthday\_[1])  {  return a.birthday\_[2] < b.birthday\_[2];  }  else return a.birthday\_[1] < b.birthday\_[1];  }  else return a.birthday\_[0] < b.birthday\_[0];  }  );  }  void searchAndPrintNoteByPhoneNumber(string phoneNumber) {  bool inp = false;  for (auto note : notebook\_)  {  if (note.phoneNumber\_ == phoneNumber)  {  cout << note;  inp = true;  }  }  if (inp != true) cout << "Номер телефона не найден" << endl;  }  private:  vector <Note> notebook\_;  };  int main(void)  {  setlocale(LC\_ALL, "rus");  cout << "Введите начальное количество записей: ";  size\_t count;  cin >> count;  cout << endl;  cin.ignore(256, '\n');  Notebook notebook(count);  cout << "База записей" << endl;  notebook.printNotebook();  cout << "База записей после сортировки" << endl;  notebook.sortNotebook();  notebook.printNotebook();  cout << "Введите номер телефона для поиска записи: ";  string phoneNumber;  getline(cin, phoneNumber);  notebook.searchAndPrintNoteByPhoneNumber(phoneNumber);  system("pause");  return 0;  } |

## Тестирование программы

Результат работы программы приведен на рисунке 4.1.

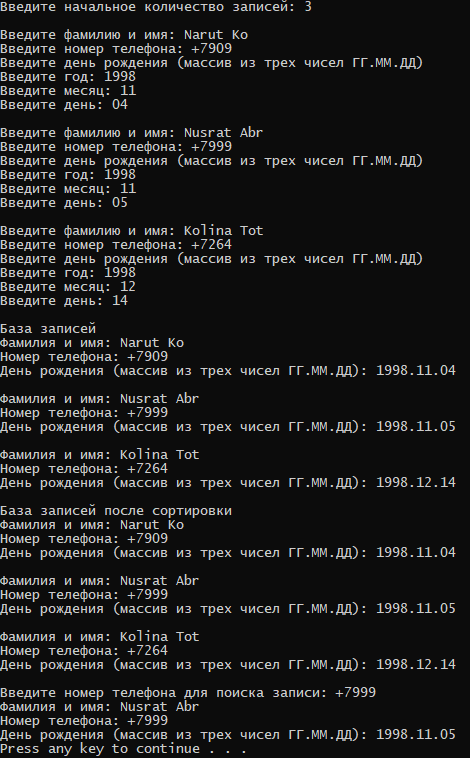


Рисунок 4.1 – Результат работы программы по стандартным потокам

# Файловые и строковые потоки. Строки класса string.

Для поддержки файлового ввода и вывода стандартная библиотека C++ содержит классы fstream, ofstream и ifstream. Как и стандартные потоки, файловые потоки обеспечивают гораздо более надежный ввод-вывод, чем старые функции библиотеки C. Для использования файловых потоков необходимо подключить заголовок <fstream>.

## Описание программы

*Используя файловые потоки написать программу, которая считывает текст из файла и выводит на экран только предложения, начинающиеся с тире, перед которым могут находиться только пробельные символы.*

## Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 5.1.

Листинг 5. – Текст файла Lab14.cpp

|  |
| --- |
| #include <iostream>  #include <fstream>  #include <cstdlib>  #include <cctype>  void sentence(std::istream& in)  {  bool state = false;  char c;  while ((c = in.get()) && isspace(c))  ;  if (c == '-') // вместо тире знак "минус"  state = true;  else  state = false;  if (state) {  while ((c = in.get()) && isspace(c)); // пропуск пробелов  std::cout << c;  while ((c = in.get()) && state && !in.eof()) {  std::cout << c;  if (c == '.' || c == '!' || c == '?')  state = false;  }  std::cout << std::endl;  }  }  int main()  {  setlocale(LC\_ALL, "rus");  std::string file\_name;  std::cout << "Полный путь к файлу: ";  std::cin >> file\_name;  std::ifstream in;  in.open(file\_name.c\_str(), std::ios::in);  if (!in.good()) {  std::cout << "Не удаётся открыть файл.\n";  exit(1);  }  while (!in.eof())  sentence(in);  in.close();  system("pause");  return 0;  } |

## Тестирование программы

Результат работы программы приведен на рисунках 5.1, 5.2.

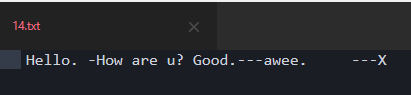


Рисунок 5.1 – Текстовый файл

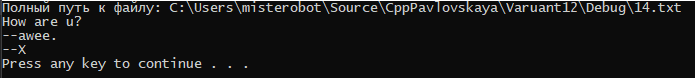


Рисунок 5.2 – Результат работы программы по файловым потокам

# Стандартная библиотека шаблонов

Стандартная библиотека шаблонов (Standard Template Library, STL) состоит из двух основных частей: набора контейнерных классов и набора обобщенных алгоритмов.

Контейнеры — это объекты, содержащие другие однотипные объекты. Контейнерные классы являются шаблонными, поэтому хранимые в них объекты могут быть как встроенных, так и пользовательских типов. Эти объекты должны допускать копирование и присваивание. Встроенные типы этим требованиям удовлетворяют; то же самое относится к классам, если конструктор копирования или операция присваивания не объявлены в них закрытыми или защищенными. В контейнерных классах реализованы такие типовые структуры данных, как стек, список, очередь и т. д.

## Описание программы

*Написать программу, решающую игру-головоломку «Игра в 15». Начальное размещение номеров — случайное. В программе использовать контейнерные классы STL.*

## Текст программы

Проект состоит из 3 файлов исходного кода, которые приведены в листингах 6.1-6.3.

Листинг 6. – Текст файла Lab15.cpp

|  |
| --- |
| #include "Game15Puzzle.h"  int main()  {  setlocale(LC\_ALL, "rus");  Game15Puzzle game15Puzzle;  game15Puzzle.mainMenu();  return 0;  } |

Листинг 6.2 – Текст файла Game15Puzzle.h

|  |
| --- |
| #pragma once  #include <iostream>  #include <iomanip>  #include <vector>  using namespace std;  #define size 4  class Game15Puzzle  {  public:  Game15Puzzle();  void mainMenu();  private:  vector<vector<int>> board;  void cleanBoard();  bool checkForSolution();  bool checkBoard();  void showBoard();  void changeBoard();  void fillBoard();  void startGame();  }; |

Листинг 6.3 – Текст файла Game15Puzzle.cpp

|  |
| --- |
| #include "Game15Puzzle.h"  Game15Puzzle::Game15Puzzle()  {  board.resize(size);  for (size\_t i = 0; i < size; ++i)  {  board[i].resize(size);  for (size\_t j = 0; j < size; ++j)  board[i][j] = 0;  }  }  void Game15Puzzle::mainMenu()  {  system("cls");  std::cout << "\nИгра в 15"  << std::endl;  std::cout << "1.Новая игра\n"  << "2.Правила игры\n"  << "3.Выход\n";  bool y = 0;  int choice;  while (!y)  {  std::cout << ">";  std::cin >> choice;  if (choice < 1 || choice > 3)  std::cout << "Некорректный ввод! Попробуйте снова." << std::endl;  else  y = 1;  }  switch (choice)  {  case 1:  system("cls");  std::cout << "Подсказка: в текущей версии игры пустая клетка представлена нулём." << std::endl;  std::cout << "Подсказка: для перемещения пустой клетки (нуля) вверх, вниз, влево или вправо\n"  << "требуется ввести значение соседнего элемента, который поменяется с пустой клеткой." << std::endl;  system("pause");  startGame();  break;  case 2:  system("cls");  std::cout << endl  << "Игра в 15 это популярная головоломка, придуманная в 1878 году Ноем Чепмэном.\n"  << "Игра представляет собой поле для игры размером 4х4 на котором случайным образом расположены числа от 1 до 15.\n"  << "Одна клетка остается пустой. Перемещая соседние числа, упорядочите их расположение (слева – направо и сверху – вниз).\n"  << "Например, первый ряд – 1, 2, 3, 4, второй – 5, 6, 7, 8 и т.д. «Вытаскивать» числа и перескакивать через числа не разрешается."  << std::endl;  system("pause");  mainMenu();  break;  case 3:  exit(0);  }  }  void Game15Puzzle::startGame()  {  do  {  cleanBoard();  fillBoard();  } while (!checkForSolution());  int turns = 0;  while (!checkBoard())  {  showBoard();  changeBoard();  turns++;  }  std::cout << "Вы победили за "  << turns << " хода!"  << std::endl;  system("pause");  }  void Game15Puzzle::cleanBoard()  {  for (int i = 0; i < size; i++)  for (int j = 0; j < size; j++)  board[i][j] = 0;  }  bool Game15Puzzle::checkForSolution()  {  int solution[size\*size] = { 0 };  int index = 0;  for (int i = 0; i < size; i++)  for (int j = 0; j < size; j++, index++)  solution[index] = board[i][j];  int inv = 0;  for (int i = 0; i < size \* size; i++)  if (solution[i])  for (int j = 0; j < i; j++)  if (solution[j] > solution[i])  ++inv;  for (int i = 0; i < size \* size; i++)  if (solution[i] == 0)  inv += 1 + i / 4;  return (inv & 1) ? 0 : 1;  }  bool Game15Puzzle::checkBoard()  {  int count = 1;  for (int i = 0; i < size; i++)  for (int j = 0; j < size; j++, count++)  {  if (i == size - 1 && j == size - 1 && board[i][j] == 0) break;  if (board[i][j] != count)  return 0;  }  return 1;  }  void Game15Puzzle::showBoard()  {  system("cls");  std::cout << "Игра в 15"  << std::endl;  for (int i = 0; i < size; i++)  {  std::cout << "-------------" << std::endl;  std::cout << "|";  for (int j = 0; j < size; j++)  {  std::cout << std::left  << std::setw(2)  << board[i][j]  << "|";  }  std::cout << std::endl;  }  std::cout << "-------------" << std::endl;  }  void Game15Puzzle::changeBoard()  {  bool y = 0;  while (!y)  {  int turn = 0;  do  {  std::cout << ">";  std::cin >> turn;  } while (turn < 0 || turn > 15);  if (turn == 0)  {  mainMenu();  return;  }  int getx = 0;  int gety = 0;  for (int i = 0; i < size; i++)  for (int j = 0; j < size; j++)  if (board[i][j] == turn)  {  getx = i;  gety = j;  }  int dx[] = { 0, 0, 1, -1 };  int dy[] = { 1, -1, 0, 0 };  for (int i = 0; i < 4; i++)  {  int newx = getx + dx[i];  int newy = gety + dy[i];  if (newx < 0 || newx > size - 1 ||  newy < 0 || newy > size - 1)  continue;  if (board[newx][newy] == 0)  {  board[newx][newy] = turn;  board[getx][gety] = 0;  y = 1;  break;  }  }  if (!y)  std::cout << "Введите числа находящиеся слева, справа, сверху или снизу (обращайте внимание на стены)!" << std::endl;  }  }  void Game15Puzzle::fillBoard()  {  bool already[15] = { 0 };  int count = 1;  for (int i = 0; i < size; i++)  for (int j = 0; j < size; j++, count++)  {  if (i == size - 1 && j == size - 1)  break;  while (true)  {  int randNum = rand() % 15;  if (!already[randNum])  {  already[randNum] = 1;  board[i][j] = randNum + 1;  break;  }  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунках 6.1-6.5.

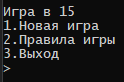


Рисунок 6.1 – Главное меню

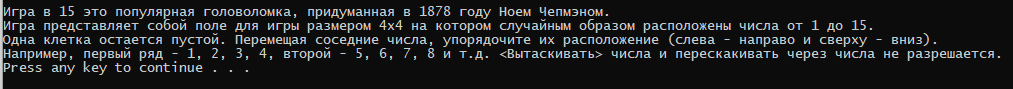


Рисунок 6.2 – Правила игры

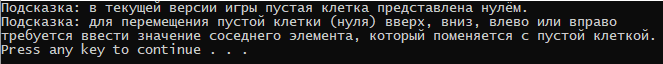


Рисунок 6.3 – Подсказки перед игрой

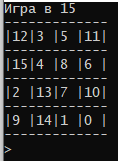


Рисунок 6.4 – Игровой стол

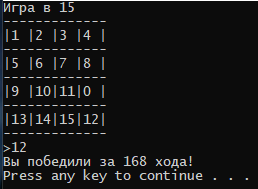


Рисунок 6.4 – Победа

Заключение

В ходе изучения дисциплины «?» по изучению языка программирования C++ были рассмотрены такие темы как:

1. Классы.
2. Наследование.
3. Шаблоны классов. Обработка исключительных ситуаций
4. Стандартные потоки.
5. Файловые и строковые потоки. Строки класса string.
6. Стандартная библиотека шаблонов.

Полученные навыки и знания будут использоваться в дальнейших проектах.

Список использованных источников

1 Павловская Т. А., C++ Программирование на языке высокого уровня: Практикум. — СПб.: Питер, 2011. — 432 с.: ил. — (Серия «Учебное пособие»).