КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра системного анализа и информационных технологий

Б.Р. АБАЙДУЛЛИН, Р.Р. ТАГИРОВ

ЛАБОРАТОРНЫЕ РАБОТЫ ПО КУРСУ

КЛАССЫ И СТРУКТУРЫ ДАННЫХ

(языки Си# и Си++)

Учебно-методическое пособие

Казань – 2017

УДК 004.43
ББК 32.973.26 - 018.1

Принято на заседании кафедры системного анализа и информационных технологий Протокол № 7 от 5 апреля 2017 года

Принято на заседании учебно-методической комиссии Института Вычислительной математики и информационных технологий Протокол № 9 от 4 мая 2017 года

Рецензенты:
доктор технических наук, профессор кафедры высшей математики института управления автоматизации и информационных технологий КНИТУ **Е.К. Вачагина;**
кандидат технических наук, старший преподаватель кафедры радиоэлектроники института физики КФУ **Е.А. Марфин**

Абайдуллин Б.Р., Тагиров Р.Р. Лабораторные работы по курсу Классы и структуры данных: Учебно-методическое пособие / Б.Р. Абайдуллин, Р.Р. Тагиров. - Казань: Казан. ун-т, 2017. - 46 с.

В данном пособии описываются структуры и классы языка объектно-ориентированного программирования Visual Си# и Visual C++ при решении конкретных задач. Пособие представляет собой перечень тем и задач и предназначено для студентов, обучающихся по направлениям «Прикладная математика и информатика», «Информационная безопасность», и «Фундаментальная информатика и информационные технологии».

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Одной из важнейших задач при освоении курса фундаментальной информатики является освоение основ программирования на примере какого-либо широко распространенного языка программирования. В связи с широким распространением операционных систем Windows, и соответственно приложений написанных на языке Си++ и Си# (связанных с платформой .NET), акцент в данном пособии сделан на языке Си# и Си++.

Данное учебно-методическое пособие призвано помочь студенту изучить основные конструкции объектно-ориентированного программирования, научить студента строить алгоритмы для решения конкретной задачи и реализовать эти алгоритмы на конкретном языке программирования. Пособие содержит классические задачи по программированию с использованием структур. Оно адресовано студентам, обучающимся по образовательным программам бакалавриата и магистратуры направлений «Фундаментальная информатика и информационные технологии», «Информационная безопасность», «Прикладная математика и информатика», «Прикладная информатика», реализуемым на кафедре системного анализа и информационных технологий в Институте Вычислительной математики и информационных технологий Казанского (Приволжского) федерального университета.

ТЕМА 1: КЛАССЫ (НАСЛЕДОВАНИЕ)

Цель.

Умение выявлять основные характеристики и общее поведение сложных объектов для описания новых типов.

Ключевые слова.

Структура и класс, элементы и сокрытие, статические элементы, конструктор, деструктор, операция, наследование

Задача1. Класс время.

Описать класс "Время" и продемонстрировать работу с ним:

1.Определить метод для подсчета количества секунд по стандартному представлению времени

2. Определить метод нормальной (стандартной) выдачи времени (по заданному количеству секунд)

3. Определить метод (операцию) сложения времен

4. Определить метод (операцию) сравнения времен

5. Определить метод печати времени

Код программы на языке Си#.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace klass_time_bul
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      class Time
      {  //поля (данные) класса
         int ho, mi, se;
         //метод класса для взятия значений из поля
         //либо для занесения в поле
         public int Ho
         {
            get { return ho; }
            set { ho = value; }
         }
         public int Mi
         {
            get { return mi; }
            set { mi = value; }
         }
         public int Se
         {
            get { return se; }
            set { se = value; }
         }
         //конструктор класса
         public Time(int pho, int pmi, int pse)
         {
            ho = pho;
            mi = pmi;
            se = pse;
            reduce();
         }
         //конструктор класса
         public Time(int pho, int pmi)
         {
            ho = pho;
            mi = pmi;
            se = 0;
            reduce();
         }
         //конструктор класса
         public Time(int pho)
```

```
        {
            ho = pho;
            mi = 0;
            se = 0;
            reduce();
        }
        //конструктор класса(с полиморфизмом)
        public Time(char a)
        {
            ho = Convert.ToInt16(a);
            mi = Convert.ToInt16(a);
            se = Convert.ToInt16(a);
           // reduce();
        }
        //конструктор класса
        public Time()
        {
            ho = 0;
            mi = 0;
            se = 0;
        }
        //метод класса
        void reduce()
        {
            mi += se / 60;
            se %= 60;
            ho += mi / 60;
            mi %= 60;
        }
        //метод класса(перегрузка функции - операции арифм)
        public static Time operator +(Time tt, int s)
        {
            Time t = new Time(tt.ho, tt.mi, tt.se + s);
            return t;
        }
        //метод класса
        public string toString()
        {
            return Convert.ToString(ho) + " : " + Convert.ToString(mi) + " : " +
Convert.ToString(se);
        }
        //метод класса
        public int second()
        { return ho * 3600 + mi * 60 + se; }

        //метод класса(перегрузка функции - операции логич)
        public static bool operator ==(Time t1, Time t2)
        {
            return ((t1.ho == t2.ho) && (t1.mi == t2.mi) && (t1.se == t2.se));
        }
        //метод класса(перегрузка функции - операции логич)
        public static bool operator !=(Time t1, Time t2)
```

```
        {
            return ((t1.ho != t2.ho) || (t1.mi != t2.mi) || (t1.se != t2.se));
        }
    }

    private void button1_Click(object sender, EventArgs e)
    {
        int ho1 = Convert.ToInt32(textBox1.Text);
        int mi1 = Convert.ToInt32(textBox2.Text);
        int se1 = Convert.ToInt32(textBox3.Text);
        //создали объект t обладаюий структурой - типа класс,
        //выделили память и конструктором его построили
        Time t = new Time(ho1, mi1, se1);
        int inc = Convert.ToInt32(textBox9.Text);
        //тут использовали метод класса(объекта(класса)) "операция сложения"
        t = t + inc;
        //тут присвоили полю объекта часы число 10
        //t.Ho = 10;
        //тут использовали метод объекта toString()
        textBox7.Text = t.toString();
        ho1 = Convert.ToInt32(textBox4.Text);
        mi1 = Convert.ToInt32(textBox5.Text);
        se1 = Convert.ToInt32(textBox6.Text);
        //тут создали второй объект класса Time
        Time t1 = new Time(ho1, mi1, se1);
        //t1 = t;
        //тут использовали метод класса "операция сравнения"
        if (t1 == t) textBox8.Text = "==";
        else textBox8.Text = "!=";
        textBox10.Text = Convert.ToString(t.second());
        //тут пример использования способности класса к полиморфизму
        //char b = Convert.ToChar(textBox11.Text);
       // int y = Convert.ToInt16(b);
       // textBox12.Text = Convert.ToString(y);
       // Time t = new Time(b);
       // textBox7.Text = t.toString();
    }
  }
}
```

Задача 2. Класс дата.

Описать класс "Дата" (наследник класса "Время"). Включить в описание конструкторы, операции и методы для решения следующих задач:

1. Определить правильность даты
2. Определить день недели для даты
3. Определить разность между двумя датами (сколько дней)
4. Определить сколько секунд межу датами (например сколько Вы прожили от момента рождения до сегодняшнего дня
5. Определить метод печати даты с временем

Визуализация работы программы.

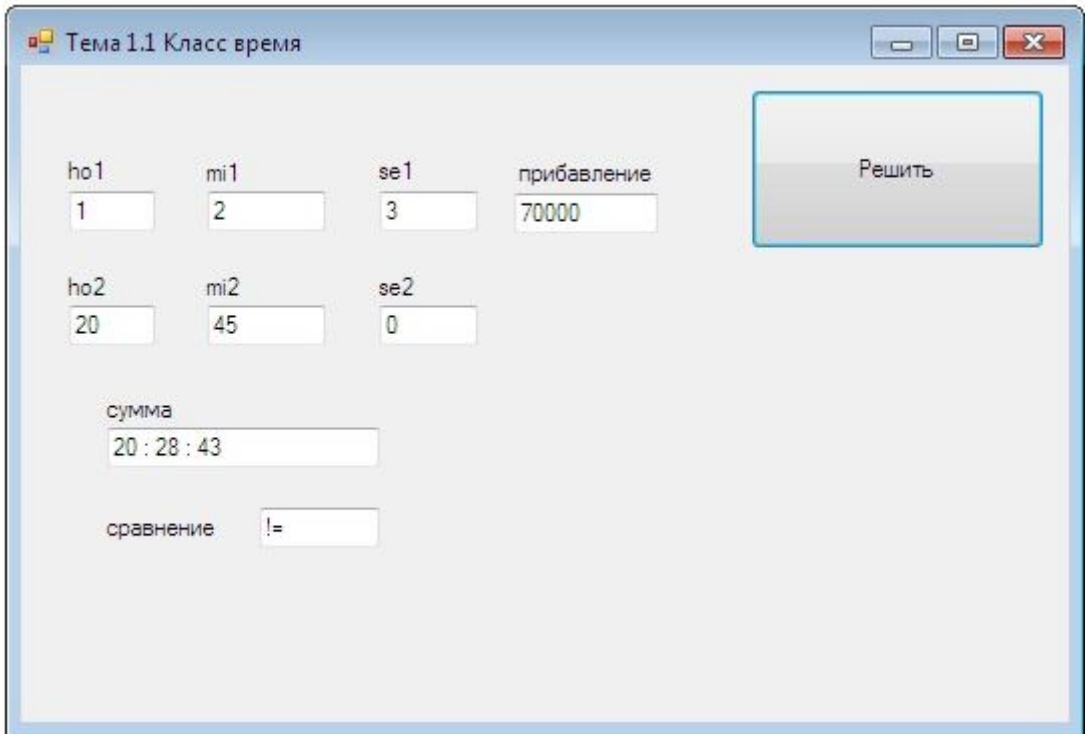

Рис. 1. Визуализация работы программы класс "Время".

Код программы на языке Си++.

```
#pragma once

namespace klassdatabul {

  using namespace System;
  using namespace System::ComponentModel;
  using namespace System::Collections;
  using namespace System::Windows::Forms;
  using namespace System::Data;
  using namespace System::Drawing;

  /// <summary>
  /// Сводка для Form1
  ///
  /// Внимание! При изменении имени этого класса необходимо также изменить
           /// </summary>
   class Time
      {
         int ho, mi, se;
          public:
                  int getHo (){return ho;}
                  void setHo (int pho) {ho=pho;}
                  int getMi (){return mi;}
                  void setMi (int pmi) {mi=pmi;}
                  int getSe (){return se;}
                  void setSe (int pse) {se=pse;}

         //конструктор
```

```
        Time(int pho=0,int pmi=0,int pse=0)
        {
           ho = pho;
           mi = pmi;
           se = pse;
           Reduce();
        }
        //метод приводит секунды в норм время
        void Reduce()
        {
           mi += se / 60;
           se %= 60;
           ho += mi / 60;
           mi %= 60;
        }
                //метод складывает времена
        Time operator +(int s)
        {
           Time t (ho, mi, se + s);
           return t;
        }
        //метод печатает время
        String^ toString()
        {
                        return Convert::ToString(ho) + " : " + Convert::ToString(mi) + " :
" + Convert::ToString(se);
        }
                //метод переводит время в секунды
        int second()
        { return ho * 3600 + mi * 60 + se; }
                //метод сравнивает времена на =
        bool operator ==(Time t)
        {
           return ((t.ho == ho) && (t.mi == mi) && (t.se == se));
        }
                //метод сравнивает времена на !=
        bool operator !=(Time t)
        {
           return ((t.ho != ho) || (t.mi != mi) || (t.se != se));
        }
     };
  //-----------------------------------------------
  public enum Month { Jan = 1, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov,
Dec};
   public enum DayW { Сб = 0, Вс, Пн, Вт, Ср, Чт, Пт, Ошибка};
  //-----------------------------------------------
  //класс дата наследник класса время
  class Date : Time
     {
        int dd, yyyy;
                Month mm;
```

```
        public:
                int getDd (){return dd;}
                void setDd (int pdd) {dd=pdd;}
                int getMm (){return mm;}
                void setMm (Month pmm) {mm=pmm;}
                int getYyyy (){return yyyy;}
                void setYyyy (int pyyyy) {yyyy=pyyyy;}

        //конструктор
       Date(int pdd=0, Month pmm=Jan,int pyyyy=0, int pho=0,int pmi=0,int pse=0) :
Time(pho,pmi,pse)
        {
           setHo(pho);
          setMi (pmi);
          setSe (pse);
                        dd = pdd;
           mm = pmm;
           yyyy = pyyyy;
        }
            //метод печатает дату и время
        String^ toString1()
        {
                        return Convert::ToString(dd) + " : " + Convert::ToString(mm) + "
: " + Convert::ToString(yyyy)+"!!!"+Convert::ToString(getHo()) + " : " +
Convert::ToString(getMi()) + " : " + Convert::ToString(getSe());
        }

                //метод определяет разницу между датами в днях
        int Razn()
        { return 1;}

                //метод определяет разницу между датами в секундах
        int RaznSec()
        { return 1;}

                //метод проверяет правильность даты
        bool Check ()
        {
           return true;
        }
                //метод выводит день недели
        DayW DayWeek ()
        {
           return Сб;
        }
      };

  public ref class Form1 : public System::Windows::Forms::Form
  //лишний код по умолчанию убран

#pragma region Windows Form Designer generated code
  //лишний код по умолчанию убран
```

```
#pragma endregion


  private: System::Void button1_Click(System::Object^  sender, System::EventArgs^
e) {

                                textBox1->Text="6767";
                                //int d1=3;
                                //Month m1=Jan;
                                //int y1=1984;
                                Date d=Date(12,Feb,2001,12,30,45);
                               // d.dd=12;d.mm=Jan;d.yyyy=1986;
                                String^ h= d.toString1();
            textBox1->Text=h;

                               // textBox1->Text="7";
                    }
  };
}
```

Визуализация работы программы.

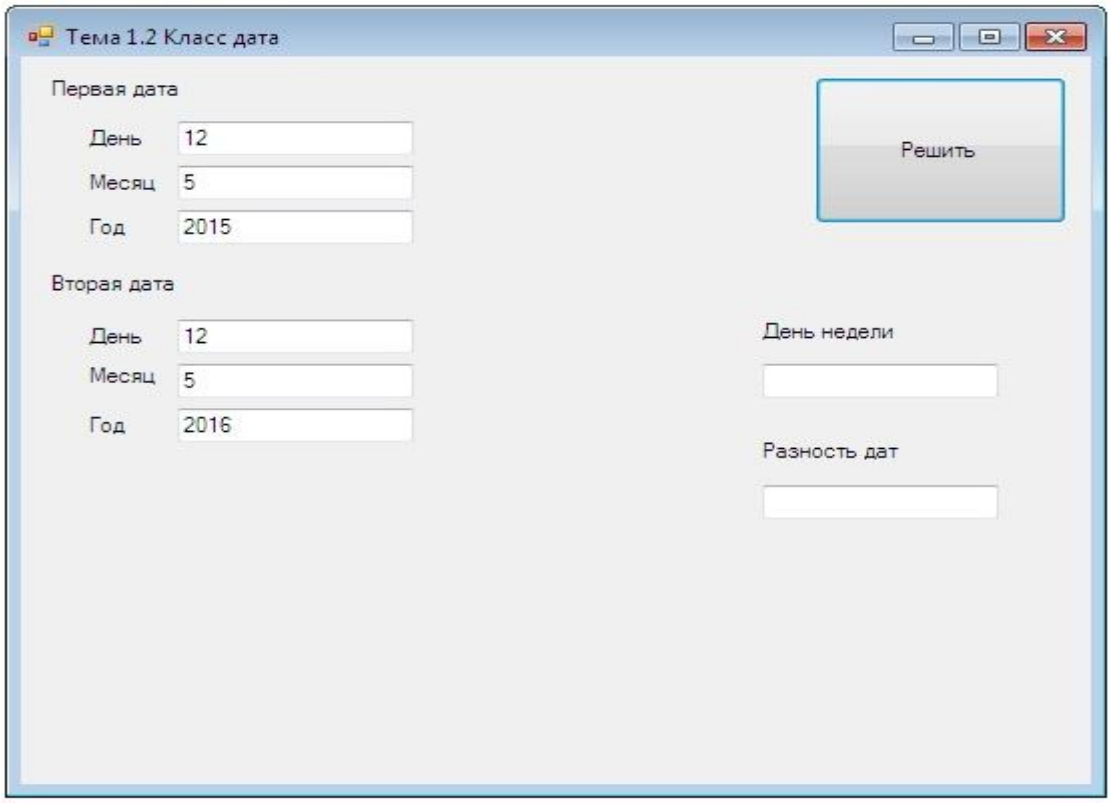

Рис. 2. Визуализация работы программы класс "Дата".

ТЕМА 2: ЛИНЕЙНЫЕ (СВЯЗНЫЕ) СПИСКИ

Цель.

Изучить использование линейных связных списков в тех задачах, где работа с массивами неэффективна, из-за большого числа неиспользуемых элементов. Для списков описание типа пока не вводится.

Основные понятия.

Несмотря на то, что элементы линейных списков обрабатываются последовательно, они могут ускорить работу с нерегулярными множествами значений, т.е. с массивами, в которые нерегулярно добавляются новые элементы и удаляются ненужные.

Описание типа для элемента списка с одной связью (указателем на следующий элемент) может выглядеть следующим образом:

```
struct E
{
  int info;          // для простоты информация это целое число
  E * next;
  E (int pinfo, E * pnext=NULL) { info = pinfo; next = pnext; }
};
```

Если предполагается движение по списку не только от начала к концу, но и наоборот, удобно будет описать новый тип с двумя указателями:

```
struct E
{
  int info;          // для простоты информация это целое число
  E * next, * prev;
  E (int pinfo=0, E * pnext=NULL, E * pprev=NULL)
    { info = pinfo; next = pnext; prev = pprev; }
};
```

Для доступа к списку понадобится описание указателя на начало – на первый элемент

и может быть на конец – на последний элемент:

```
E * first=NULL, * last=NULL;
```

Ключевые слова.

Список, движение по списку, содержательная информация, указатель, доступ к элементу.

Задача 1. Однонаправленный (односвязный) линейный список (Си++).

Дан линейный список с одним указателем на следующий элемент. Написать функции для создания и добавления нового элемента в список и печать списка. В программе продемонстрировать работу написанных функций. Упорядочить список по возрастанию (с разрушением списка).

Результат:

2 -> 3 -> 5 -> 7

Код программы на языке Си++.

Вариант 1

```
using namespace std;

// описания типов и функций

struct E
{
  int info;          // для простоты информация это целое число
  E * next, * prev;
```

```
  E (int pinfo=0, E * pnext=NULL, E * pprev=NULL)
    { info = pinfo; next = pnext; prev = pprev; }
};

E * first=NULL, * last=NULL;

void AddBegin (int x)
{
  first = new E (x, first);
  if ( last == NULL )          // если список был пустой, то новый элемент
    last = first;              // стал первым и последним одновременно!
}

void AddBegin2 (int x)
{
  E * p = first;
  first = new E (x, first);
  if ( p != NULL )
    p->prev = first;
  else
    last = first;
}

void PrintList ()
{
}
```

Вариант 2

В данном задании код программы демонстрируется полностью, в дальнейшем стандартная часть кода между #pragma once и #pragma endregion для экономии места и из-за бессмысленности его повторения будем пропускать

```
#pragma once
#include <stdio.h>
#include <stdlib.h>
#include "stdafx.h"
#include "cstdlib"
#include <iostream>


namespace Список_CPPокно {

  using namespace System;
  using namespace System::ComponentModel;
  using namespace System::Collections;
  using namespace System::Windows::Forms;
  using namespace System::Data;
  using namespace System::Drawing;

  struct Inode
{
  int data;
```

```
  lnode *next;
} *head, *visit;
void llist_add(struct lnode **q, int num);
void llist_bubble_sort(void);
String^ llist_print(void);
lnode *newnode = NULL;


  /// <summary>
  /// Сводка для Form1
  ///
  /// Внимание! При изменении имени этого класса необходимо также изменить
  ///         свойство имени файла ресурсов ("Resource File Name") для средства
компиляции управляемого ресурса,
  ///         связанного со всеми файлами с расширением .resx, от которых
зависит данный класс. В противном случае,
  ///         конструкторы не смогут правильно работать с локализованными
  ///         ресурсами, сопоставленными данной форме.
  /// </summary>
  public ref class Form1 : public System::Windows::Forms::Form
  {
  public:
          Form1(void)
          {
                  InitializeComponent();
                  //
                  //TODO: добавьте код конструктора
                  //
          }

  protected:
          /// <summary>
          /// Освободить все используемые ресурсы.
          /// </summary>
          ~Form1()
          {
                  if (components)
                  {
                          delete components;
                  }
          }
  private: System::Windows::Forms::Button^  button1;
  protected:
  private: System::Windows::Forms::TextBox^  textBox1;
  private: System::Windows::Forms::Label^  label1;
  private: System::Windows::Forms::Label^  label2;
  private: System::Windows::Forms::TextBox^  textBox2;
  private: System::Windows::Forms::Label^  label3;
  private: System::Windows::Forms::TextBox^  textBox3;
  private: System::Windows::Forms::Button^  button2;
  private: System::Windows::Forms::Button^  button3;
```

```
  private:
        /// <summary>
        /// Требуется переменная конструктора.
        /// </summary>
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Обязательный метод для поддержки конструктора - не изменяйте
        /// содержимое данного метода при помощи редактора кода.
        /// </summary>
        void InitializeComponent(void)
        {
                this->button1 = (gcnew System::Windows::Forms::Button());
                this->textBox1 = (gcnew System::Windows::Forms::TextBox());
                this->label1 = (gcnew System::Windows::Forms::Label());
                this->label2 = (gcnew System::Windows::Forms::Label());
                this->textBox2 = (gcnew System::Windows::Forms::TextBox());
                this->label3 = (gcnew System::Windows::Forms::Label());
                this->textBox3 = (gcnew System::Windows::Forms::TextBox());
                this->button2 = (gcnew System::Windows::Forms::Button());
                this->button3 = (gcnew System::Windows::Forms::Button());
                this->SuspendLayout();
                //
                // button1
                //
                this->button1->Location = System::Drawing::Point(366, 30);
                this->button1->Name = L"button1";
                this->button1->Size = System::Drawing::Size(75, 23);
                this->button1->TabIndex = 0;
                this->button1->Text = L"список";
                this->button1->UseVisualStyleBackColor = true;
                this->button1->Click += gcnew System::EventHandler(this,
&Form1::button1_Click);
                //
                // textBox1
                //
                this->textBox1->Location = System::Drawing::Point(40, 60);
                this->textBox1->Name = L"textBox1";
                this->textBox1->Size = System::Drawing::Size(283, 20);
                this->textBox1->TabIndex = 1;
                this->textBox1->Text = L"4 6 7 8 4 5 6 ";
                //
                // label1
                //
                this->label1->AutoSize = true;
                this->label1->Location = System::Drawing::Point(31, 28);
                this->label1->Name = L"label1";
                this->label1->Size = System::Drawing::Size(225, 13);
                this->label1->TabIndex = 2;
                this->label1->Text = L"исходный последовательность для списка";
                //
```

```
            // label2
            //
            this->label2->AutoSize = true;
            this->label2->Location = System::Drawing::Point(30, 100);
            this->label2->Name = L"label2";
            this->label2->Size = System::Drawing::Size(95, 13);
            this->label2->TabIndex = 4;
            this->label2->Text = L"исходный список";
            //
            // textBox2
            //
            this->textBox2->Location = System::Drawing::Point(36, 132);
            this->textBox2->Name = L"textBox2";
            this->textBox2->Size = System::Drawing::Size(283, 20);
            this->textBox2->TabIndex = 3;
            //
            // label3
            //
            this->label3->AutoSize = true;
            this->label3->Location = System::Drawing::Point(29, 166);
            this->label3->Name = L"label3";
            this->label3->Size = System::Drawing::Size(124, 13);
            this->label3->TabIndex = 6;
            this->label3->Text = L"упорядоченный список";
            //
            // textBox3
            //
            this->textBox3->Location = System::Drawing::Point(36, 198);
            this->textBox3->Name = L"textBox3";
            this->textBox3->Size = System::Drawing::Size(283, 20);
            this->textBox3->TabIndex = 5;
            //
            // button2
            //
            this->button2->Location = System::Drawing::Point(366, 72);
            this->button2->Name = L"button2";
            this->button2->Size = System::Drawing::Size(84, 23);
            this->button2->TabIndex = 7;
            this->button2->Text = L"упорядочить";
            this->button2->UseVisualStyleBackColor = true;
            this->button2->Click += gcnew System::EventHandler(this,
&Form1::button2_Click);
            //
            // button3
            //
            this->button3->Location = System::Drawing::Point(366, 116);
            this->button3->Name = L"button3";
            this->button3->Size = System::Drawing::Size(75, 23);
            this->button3->TabIndex = 8;
            this->button3->Text = L"закрыть";
            this->button3->UseVisualStyleBackColor = true;
```

```
                this->button3->Click += gcnew System::EventHandler(this,
&Form1::button3_Click);
                //
                // Form1
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                this->ClientSize = System::Drawing::Size(462, 360);
                this->Controls->Add(this->button3);
                this->Controls->Add(this->button2);
                this->Controls->Add(this->label3);
                this->Controls->Add(this->textBox3);
                this->Controls->Add(this->label2);
                this->Controls->Add(this->textBox2);
                this->Controls->Add(this->label1);
                this->Controls->Add(this->textBox1);
                this->Controls->Add(this->button1);
                this->Name = L"Form1";
                this->Text = L"Тема 2.1 Список C++";
                this->ResumeLayout(false);
                this->PerformLayout();

        }
#pragma endregion


        void llist_add(struct lnode **q, int num)
{
  struct lnode *tmp;
  tmp = *q;
  if (*q == NULL) {
        *q = (lnode*) malloc(sizeof(struct lnode));
        tmp = *q;
  }
  else {
        while (tmp->next != NULL)
                tmp = tmp->next;
        tmp->next = (lnode*)malloc(sizeof(struct lnode));
        tmp = tmp->next;
  }

  tmp->data = num;
  tmp->next = NULL;
}

String^ llist_print(void)
{
  visit = head;
String^h1="";
  while (visit != NULL) {
        h1=h1+Convert::ToString(visit->data)+" ";
```

```
        visit = visit->next;
  }

  return h1;
}

void llist_bubble_sort(void)
{
  struct lnode *a = NULL;
  struct lnode *b = NULL;
  struct lnode *c = NULL;
  struct lnode *e = NULL;
  struct lnode *tmp = NULL;

  while (e != head->next)
  {
        c = a = head;
        b = a->next;
        while (a != e)
        {
              if (a->data > b->data)
              {
                    if (a == head)
                    {
                          tmp = b->next;
                          b->next = a;
                          a->next = tmp;
                          head = b;
                          c = b;
                    }
                    else
                    {
                          tmp = b->next;
                          b->next = a;
                          a->next = tmp;
                          c->next = b;
                          c = b;
                    }
              }
              else
              {
                    c = a;
                    a = a->next;
              }
              b = a->next;
              if (b == e)
                    e = a;
        }
  }
}
  private: System::Void button1_Click(System::Object^  sender, System::EventArgs^
e) {
```

```
    //int n;
 array <wchar_t>^ mass=textBox1->Text->ToCharArray();
   int n=mass->Length;
   int a[1000];
   int l=0, j=0, sum=0, k=0;
   //int max=-1,min=100000;
   String^ ls;
   for(int i=0; i<n; i++)
   {
   if (Convert::ToString(mass[i]) != " ")
    {
    ls=Convert::ToString(mass[i]);  l=l*10+Convert::ToInt16(ls);
              if (i==n-1) { a[j]=l; k++; j++; }
     }
   else {  a[j]=l; k++; j++; l=0; }
   }
   n=j;ls="";
  for(int i=0; i<n; i++)
  ls=ls+Convert::ToString(a[i])+" ";
//textBox3->Text=ls;
//-------------------------------------------
  //struct lnode *newnode = NULL;
  for(int i=0; i<n; i++)
  llist_add(&newnode, a[i]);
//llist_add(&newnode, 3);
//llist_add(&newnode, 2);
String^ h="";
head = newnode;
h=llist_print();
textBox2->Text=h;
                    }
private: System::Void button3_Click(System::Object^  sender, System::EventArgs^
e) {
                    Form1::Close();
            }
private: System::Void button2_Click(System::Object^  sender, System::EventArgs^
e) {
head = newnode;
llist_bubble_sort();
String^ h1="";
h1=llist_print();
textBox3->Text=h1;
            }
};
}
```

Визуализация работы программы.

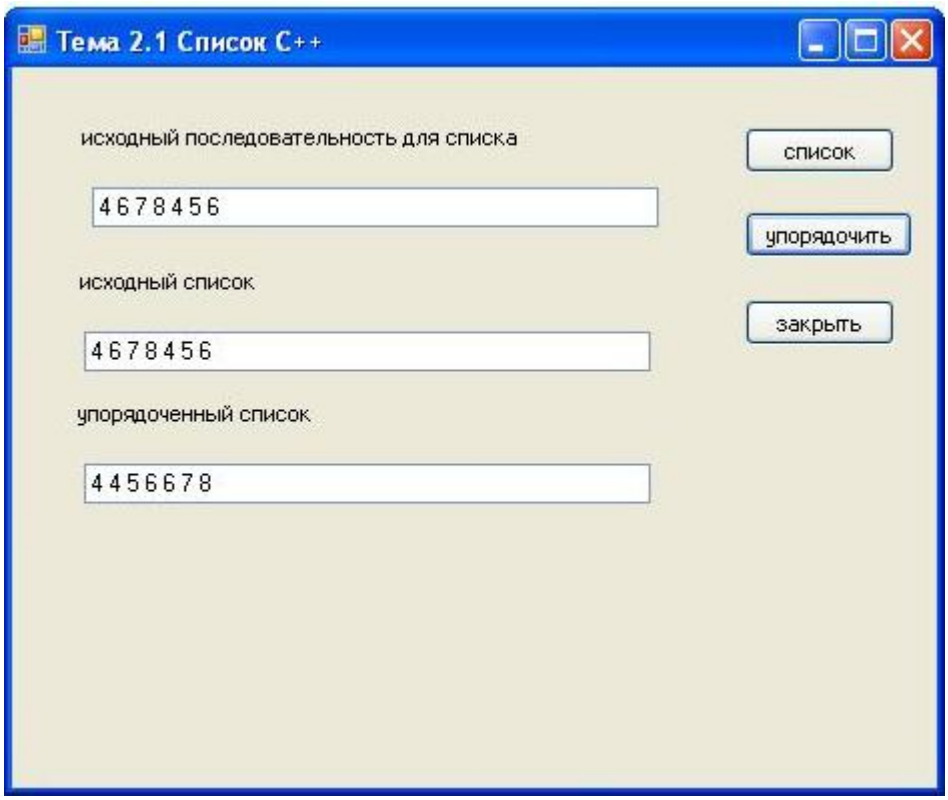

Рис. 3. Визуализация работы программы с линейным списком.

Задача 2. Двунаправленный (двусвязный) линейный список (Си#).

Дан линейный список с указателями на следующий и на предыдущий элементы. Написать функции для создания и добавления нового элемента в список, печать списка, упорядочение по возрастанию (без разрушения списка). В программе продемонстрировать работу написанных функций.

Дополнительное задание. Упорядочить список (с разрушением списка) по возрастанию.

Результат:

2 <-> 3 <-> 5 <-> 7

Код программы на языке Си#.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace _2.Список_CS
{
   public partial class Form1 : Form
   {
      public class E
      {
         int info;
         E next, prev;
```

```
    public int INFO
    {
        get { return info; }
        set { info = value; }
    }
    public E Next
    {
        get { return next; }
        set { next = value; }
    }
    public E Prev
    {
        get { return prev; }
        set { prev = value; }
    }
    public E(int pinfo, E pnext)
    {
        INFO = pinfo;
        Next = pnext;
        prev = null;
    }
}
public static E first = null, last = null;
public static void Add(int pinfo)
{
    E p = first;
    first = new E(pinfo, first);
    if (last == null)
        last = first;
    else p.Prev = first;
}
public static string tostr()
{
    E p = first;
    string h = "";
    for (; p != null; p = p.Next)
        h = h + Convert.ToString(p.INFO) + " ";
    return h;
}
public Form1()
{
    InitializeComponent();
}

private void button1_Click(object sender, EventArgs e)
{
    string[] sNums = textBox1.Text.Split(' ');
    int n = sNums.Length;
    for (int i = 0; i < n; i++)
        Add(int.Parse(sNums[i]));
    textBox2.Text = tostr();
```

```
        }
    }
}
```

ТЕМА 3: ЖАДНЫЕ АЛГОРИТМЫ

Цель.

Умение находить быстрое «жадное» решение задачи или построить соотношения для последовательного динамического вычисления ответов. Сравнить алгоритмы динамического вычисления ответов с использованием массивов и очередей.

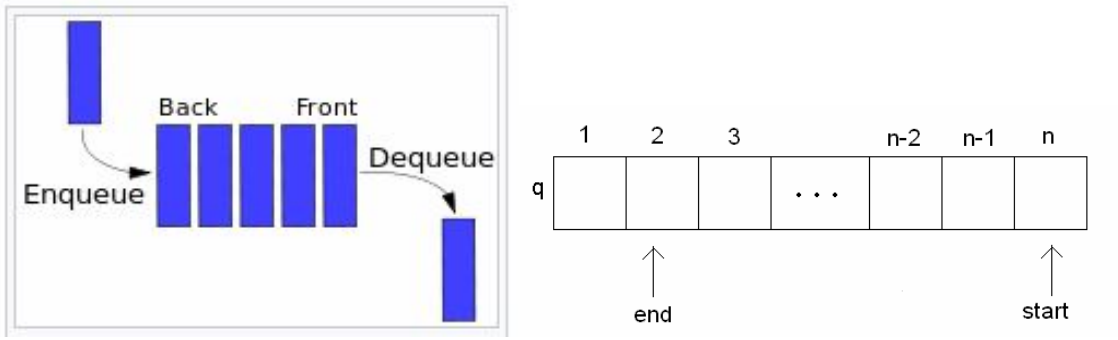

Рис. 4. Способы реализации очереди.

Основные понятия.

Очередь – абстрактный тип данных с дисциплиной доступа к элементам «первый пришёл - первый вышел» (FIFO, *first in - first out*), в отличии от стека.

Стек – абстрактный тип данных с дисциплиной доступа к элементам «последним пришёл — первым вышел» (LIFO, *last in - first out*).

Существует несколько способов реализации очереди в языках программирования

1) массив с двумя переменными start и end
2) связный список
3) с использованием двух стеков

процедура enqueue(*x*): S1.push(*x*)
функция dequeue():
если S2 пуст:
если S1 пуст: сообщить об ошибке: очередь пуста
пока S1 не пуст: S2.push(S1.pop())
вернуть S2.pop()

Практически во всех развитых языках программирования реализованы очереди по умолчанию. В языках си++ (c#) они объявляются зарезервированным словом queue <int> q; (Queue<int> q = new Queue<int>();)

Команды при использовании очередей (стеков) по умолчанию:

q.push() – помещает элемент в конец очереди (очередь изменяется) в c# это q.enqueue()

q.size() – выдает размер очереди (только c++) в c# q.count() – выдает количество элементов в очереди (стеке)

q.pop() – удаляет первый элемент очереди, в c# это q.dequeue() и очередь изменяется

q.front() – берет первый элемент очереди или верх стека (очередь не изменяется), в c# это q.peek()

q.back() – берет последний элемент очереди (очередь не изменяется) только c++

s.contains() – проверяет есть ли элемент в стеке только c#

q.empty() – проверяет пустая ли очередь

q.clear() – удалить все элементы из очереди только c#

Стеки также можно реализовать 3 способами, как и очереди, например как линейный список:

```
struct stack
{   char *data;
struct stack *next;};
```

Практически во всех развитых языках программирования реализованы стеки по умолчанию. В языке c# они объявляются зарезервированным словом stack, например Stack<int> s=new Stack<int> ();

Ключевые слова.

Очереди, стеки, массивы, жадные алгоритмы.

Задача 1. Максимальная подпоследовательность Си++.

В последовательности целых чисел найти самую длинную строго возрастающую подпоследовательность, т.е. последовательность элементов с возрастающими номерами и значениями. В окне задаются через пробел целые числа исходной последовательности. Напечатать в окне длину найденной подпоследовательности и саму подпоследовательность. Реализовать алгоритм нахождения подпоследовательности с использованием массивов и очередей.

Алгоритм.

Заведём вспомогательный массив, и для каждого элемента исходного массива будем хранить в нём длину самой длинной подпоследовательности чисел, которая заканчивается в этом элементе. Для каждого элемента, начиная с первого, будем просматривать предыдущие элементы, и смотреть, какую цепочку может продолжить этот элемент. Самую большую длину будем запоминать. В конце найдём самое большое число, которое попадёт в этот вспомогательный элемент. Сложность алгоритма O (n^2) и затраты памяти O (n).

Исходные данные для проверки:

1 3 7 2 4 9 5 7 8 11 2

Результат: 7

Примечание. В качестве такой подпоследовательности могут быть выданы такие числа:

1 2 4 5 7 8 11

Код программы на языке Си++.

```
#pragma once
```

```
# include <iostream>
# include <queue>
# include <cstdlib>

int n;
int* a, *len;
namespace oceredbulcpp {

  using namespace System;
  using namespace System::ComponentModel;
  using namespace System::Collections;
  using namespace System::Windows::Forms;
  using namespace System::Data;
  using namespace System::Drawing;
    using namespace std;
  /// <summary>
  /// Сводка для Form1
  ///
  /// Внимание! При изменении имени этого класса необходимо также изменить

#pragma region Windows Form Designer generated code
#pragma endregion

  private: System::Void button1_Click(System::Object^   sender, System::EventArgs^
e) {
           String ^h="";
                       queue <int> q;
                    queue <int> q1;
               q.push(1);
                    q.push(2);
                    q.push(3);
                    int x=q.size();
                    textBox4->Text=Convert::ToString(x);
                   while (!q.empty())
                    {
               h=h+" "+Convert::ToString(q.front());
            q.pop();
           // h=h+" "+Convert::ToString(q.front());
                    // q.pop();
                    // h=h+" "+Convert::ToString(q.front());
                    // q.pop();
                     }
                     textBox2->Text=h;
                    //-----------------------------------------------
          //n=Convert::ToInt32(textBox1->Text);

                            len= new int[n];
                     h="";
                            array<String^> ^sNums = textBox1->Text->Split(' ');
                            n=sNums->Length;
              a= new int[n];
                            len= new int[n];
```

```
                textBox4->Text=Convert::ToString(n);
                for (int i = 0; i < n; i++)
                {
                        a[i]=Convert::ToInt32(sNums[i]);
   //q.push(a[i]);
                h=h+Convert::ToString(a[i])+" ";
}
textBox2->Text=h;
                for (int i = 0; i < n; i++)
                        q.push(a[i]);
                h="";
for (int i = 0; i < n; i++)
                {h=h+Convert::ToString(q.front())+" "; if (!q.empty()) q.pop();}
textBox2->Text=h;
for (int i = 0; i < n; i++)
                        q.push(a[i]);



                len[0]=1;
                //q1.push(1); //int pred=0,sled=0,predj=0;
                for (int i=1;i<n;i++)
                { //if (!q1.empty()) q1.pop(); q1.push(q.front()); q.pop();
                        int k=1;
   //for (int j=0;j<i;j++)
   //q1.push(q.front()); q.pop();
                        for (int j=0;j<i;j++)
                        { ////predj=q1.front;
                         //int k=1;
                        //if ((q1.front()<q.front())&&(k<len[j]+1))
                                //if (!q1.empty()) q1.pop();
                                if(a[j]<a[i]&&k<len[j]+1)
            //k=q1.front()+1;
                                        k=len[j]+1;
                        }
                        len[i]=k;
   //q1.push(k);
                }
                h="";
for (int i=0;i<n;i++)
                        h=h+Convert::ToString(len[i]);
textBox5->Text=h;

                int k=1;
                for (int i=0;i<n;i++)
                        if(len[i]>k)
                                k=len[i];
                textBox3->Text=Convert::ToString(k);
                int j=n-1;
                for (;len[j]!=k;j--)
                        ;
                int flag=a[j];
```

```
                        String^ h1=Convert::ToString(a[j]);
                        k--;
                        for (;k>0;j--)
                        {
                                if ((len[j]==k)&&(a[j]<flag))
                                {
                                        h1=Convert::ToString(a[j])+" "+h1;
                                        flag=a[j];
                                        k--;
                                }
                        }
                        textBox2->Text=h1;
//-----------------------------------------------
                }
};
}
//-----------------------------------пример программы с использованием только
очередей
        for (int i = 0; i < n; i++)
                        q.push(a[i]);
                int f=0;
        for (int i = 0; i < n-1; i++)
                {q1.push(q.front()); q.pop();
                if ((q.front()>q1.front())&&(f==0)) {q2.push(q1.front());
q2.push(q.front()); f=0; }
                else {f=1; if (q.front()>q2.back()) {q2.push(q.front());f=0;};}
                }
                //-------------------------------------
                h="";
                textBox3->Text=Convert::ToString(q2.size());
                while (!q2.empty()) {h=h+Convert::ToString(q2.front())+" ";
q2.pop();}
                q2.push(3);
                q.push(3);
                textBox2->Text=h;
//-----------------------------------------------
```

Визуализация работы программы.

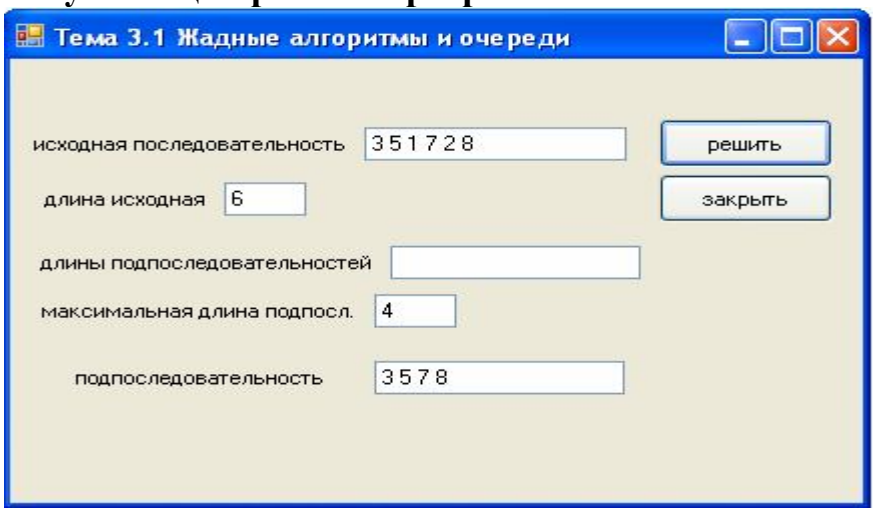

Рис. 5. Визуализация работы программы с жадным алгоритмом.

Задача 2. Максимальная подпоследовательность Си#.

В последовательности целых чисел найти самую длинную строго возрастающую подпоследовательность, т.е. последовательность элементов с возрастающими номерами и значениями. В окне задаются через пробел целые числа исходной последовательности. Напечатать в окне длину найденной подпоследовательности и саму подпоследовательность. Реализовать алгоритм нахождения подпоследовательности с использованием массивов и очередей.

Дополнительное задание. Написать жадный алгоритм с использованием только очередей.

Код программы на языке Си#.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace ocered_csh_bul
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         string[] sNums = textBox2.Text.Split(' ');
         int n = sNums.Length;


         Queue<int> q = new Queue<int>();

         for (int i = 0; i < n; i++) q.Enqueue(Convert.ToUInt16(sNums[i]));

         string h="";
         int i1 = 0;
         while (q.Count != 0)
         {
            i1++;
```

```
            int p = q.Peek(); q.Dequeue();
            h = h + Convert.ToString(p)+" ";

        };
        textBox1.Text = h;

    }
  }
}
```

ТЕМА 4: ДВОИЧНЫЕ ДЕРЕВЬЯ

Цель.

Изучение методов работы со сложными структурами данных, организованными в виде дерева. Для дерева описание типа пока не вводится.

Основные понятия.

Дерево - структура данных имеющая узлы (вершины) и ветви (связи со следующими вершинами). Первый узел, с которого начинаются связи с последующими узлами называется корнем дерева и узлом-предком. Следующий узел (узел следующего уровня) называется узлом-потомком. Дерево называется двоичным, когда узлы содержат связь только с одним или двумя другими узлами.

Варианты хранения деревьев – массив узлов или нелинейный список.

Описание объекта для узла двоичного дерева (нелинейный список).

```
struct Node
{   int info;
   Node * left, * right;
   E (int pinfo, E * pleft=NULL, E * pright=NULL) { info = pinfo; left = pleft; right =
pright; }
};
```

Описание указателя на корень дерева.

```
   Node * root = NULL;
```

Основные алгоритмы – добавление/удаление узлов, обход дерева.

Использование рекурсивных алгоритмов

Если дерево не двоичное, но количество потомков у каждого узла не очень большое, то можно использовать массив указателей на потомков:

```
const int K=10;
struct Node
{   Node * ref [K];
…};
```

Для ускорения движения по дереву и поиска иногда в каждом узле дополнительно хранят указатель на предка этого узла (для корня - NULL):

```
struct Node
{   Node * parent;
…};
```

Эти способы хранения деревьев характеризуются направлением движения

по дереву сверху вниз. Но можно использовать и обратный способ хранения указателей – от потомков к предку, т.е. каждый узел хранит содержательную информацию и адрес родительского узла. Это способ можно применить и к произвольным деревьям, а сами узлы хранить в обычном или динамическом массиве:

```
struct Node
{   int info;
   int parent;
   E (int pinfo, int pparent=-1) { info = pinfo; parent = pparent; }
};
```

Здесь значение -1 указывает на отсутствие родителя – корневой узел.

Ключевые слова.

Дерево, двоичное дерево, узел, корень, узел-предок, узел-потомок, нелинейный список.

Задача 1. Дерево родословной. (Си++)

Создать двоичное дерево своей родословной (дерево хранится в виде нелинейного списка). Выполнить обход всех узлов дерева по уровням и напечатать на форме все имена, хранящиеся в узлах.

Алгоритм.

Прогоним все узлы дерева последовательно через очередь, начиная с корня. На каждом шаге будем вытаскивать из очереди первый узел и помещать в конец очереди его потомков. Повторять будем до тех пор, пока не опустеет очередь.

Код программы на языке Си

```
#pragma once
# include <iostream>
# include <queue>
# include <cstdlib>

namespace My1Дерево_CPP {

  using namespace System;
  using namespace System::ComponentModel;
  using namespace System::Collections;
  using namespace System::Windows::Forms;
  using namespace System::Data;
  using namespace System::Drawing;
  using namespace std;
  struct Node
  {
        int info;
        Node * left, * right;
        Node (int pinfo, Node * pleft=NULL, Node * pright=NULL)
        {
              info = pinfo;
              left = pleft;
              right = pright;
```

```
        }
  };
  /// <summary>
  /// Сводка для Form1
  ///
  /// Внимание! При изменении имени этого класса необходимо также изменить
  ///          свойство имени файла ресурсов ("Resource File Name") для средства
компиляции управляемого ресурса,


#pragma region Windows Form Designer generated code
#pragma endregion
  private: System::Void button1_Click(System::Object^  sender, System::EventArgs^
e) {
            String^ h;
            Node * root;
            root = new Node (5);
            root->left = new Node (3);
            root->left->left = new Node (1);
            root->left->right = new Node (91);
            root->right = new Node (11);
            root->right->right = new Node (71);
            root->right->left = new Node (42);
            root->right->right->right = new Node (14);
            root->right->right->left = new Node (24);
            root->right->left->right = new Node (784);
            root->right->left->left = new Node (74);
            String^ h="";
            queue <Node *> q;
            q.push (root);int i=0;
            while ( ! q.empty () )
            {       Node * p = q.front ();
                    q.pop ();
                    int x=p->info;
                    h=h+" "+x.ToString();
                       i++;
              int l=(int)Math.Log(i,2);
              int x1 = (int)(1200 / Math.Pow(2, l) * (i % (int)Math.Pow(2, l)+0.5));
              int y = 30 + 40 * l;
              // gr.DrawString(p.INFO, fnt, p3, x, y);
                            if ( p->left != NULL ) q.push (p->left);
                    if ( p->right != NULL ) q.push (p->right);
            }
                    textBox1->Text=h;
                     }
  };
}
```

Визуализация работы программы.

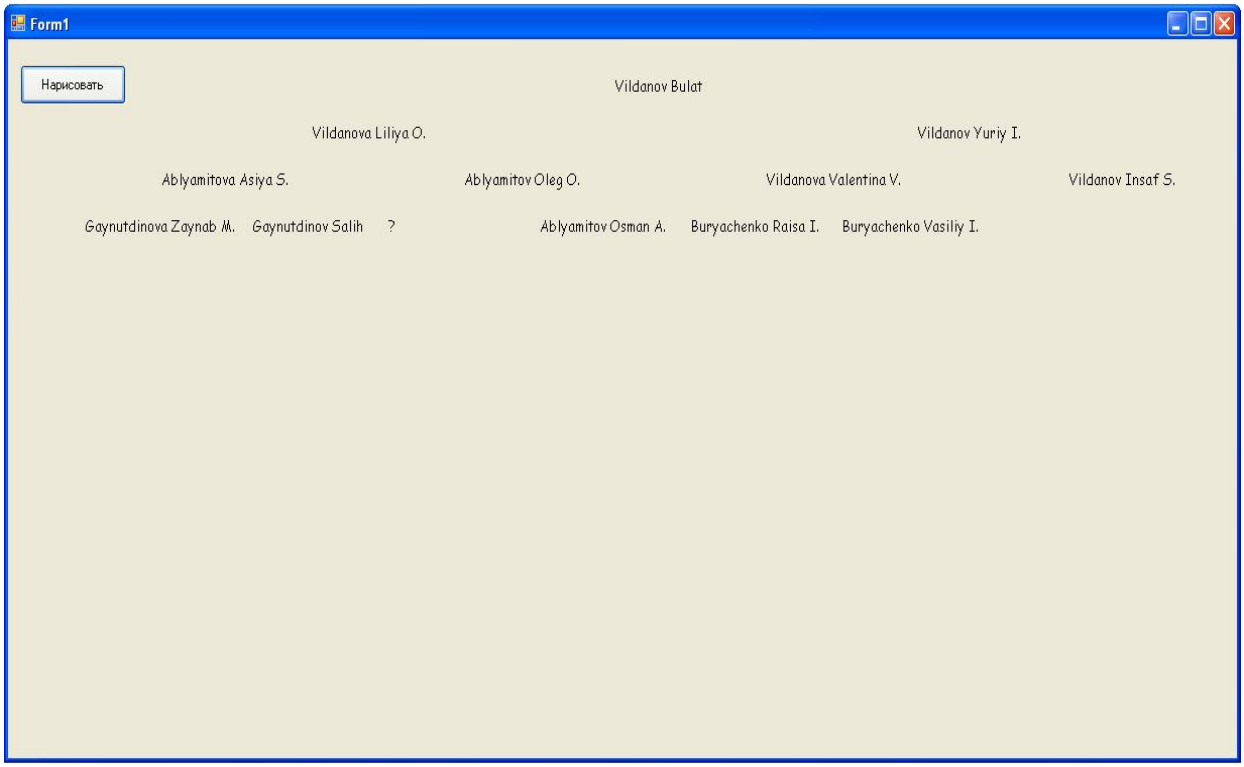

Рис. 6. Визуализация работы программы с двоичным деревом.

Задача 2. Дерево родословной. (Си#)

Создать двоичное дерево своей родословной (дерево хранится в виде нелинейного списка). Выполнить обход всех узлов дерева по уровням и напечатать на форме все имена, хранящиеся в узлах.

Код программы на языке Си#.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace _2.Дерево_CS
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      public class Node
      {
         string info;
         Node left, right;
         public string INFO
         {
```

```
            get { return info; }
            set { info = value; }
        }
        public Node Left
        {
            get { return left; }
            set { left = value; }
        }
        public Node Right
        {
            get { return right; }
            set { right = value; }
        }
        public Node(string pinfo, Node pleft, Node pright)
        {
            info = pinfo;
            left = pleft;
            right = pright;
        }
    }
    private void button1_Click(object sender, EventArgs e)
    {  Node root = new Node("Vildanov Bulat", null, null);
        root.Left = new Node("Vildanova Liliya O.", null, null);
        root.Left.Right = new Node("Ablyamitov Oleg O.", null, null);
        root.Right = new Node("Vildanov Yuriy I.", null, null);
        root.Right.Right = new Node("Vildanov Insaf S.", null, null);
        Graphics gr = this.CreateGraphics();
        Brush p3 = new SolidBrush(Color.Black);
        Font fnt = new System.Drawing.Font("Comic Sans MS", (float)10);
        Queue<Node> q = new Queue<Node>();
        int i=0;
        q.Enqueue(root);
        while (q.Count != 0)
        {
            i++;
            int l=(int)Math.Log(i,2);
            int x = (int)(1200 / Math.Pow(2, l) * (i % (int)Math.Pow(2, l)+0.5));
            int y = 30 + 40 * l;
            Node p = q.Peek(); q.Dequeue();
            gr.DrawString(p.INFO, fnt, p3, x, y);
            if (p.Left != null) q.Enqueue(p.Left);
            if (p.Right != null) q.Enqueue(p.Right);
        }
    }
  }
}
```

ТЕМА 5: ГРАФЫ

Цель.

Изучить основные и задачи на графах и с графами и алгоритмы их решения. Оценка сложности алгоритмов, оптимизация алгоритмов.

Основные понятия.

Граф – структура, имеющая вершины и ребра (связи с другими вершинами). В отличии от дерева две вершины могут быть соединены несколькими ребрами (тогда это кратные ребра) и вершины какого либо уровня могут быть связаны с вершинами любого уровня (могу быть организованы циклы). Также одна вершина может быть соединена ребром сама с собой, тогда это называется петля.

Варианты хранения графа в памяти компьютера – в виде массива (матрица смежности) и в виде структуры (нелинейный список). Существуют понятия обход графа в ширину (BFS), и обход графа в глубину (DFS, Backtracking).

Ключевые слова.

Граф, вершина, ребро, дуга, вес, петля, матрица смежности, списки смежности, путь, цикл, каркас.

Задача 1. Граф в Си++.

Для неориентированного графа (без петель и кратных рёбер) задаётся число вершин и квадратная матрица связности из 0 и 1. Написать программу для нахождения и печати кратчайшего пути от одной заданной вершины (начальной) до другой (целевой). Печатать длину пути или -1, если пути нет. Также напечатать последовательный список вершин, через которые проходит кратчайший путь.

Дополнительное задание. Решить ту же задачу, но граф описать в виде структуры.

Алгоритм.

Будем последовательно пропускать все достижимые вершины через очередь, отмечая их в специальном массиве, чтобы повторно не включать в очередь. В этом вспомогательном массиве хранится длина кратчайшего пути до каждой вершины от начальной. Значение 2000000000 означает, что до данной вершины ещё не дошли. На каждом шаге выбираем из начала очереди очередную вершину и в конец очереди добавляются те вершины, куда ведут рёбра из выбранной вершины (и которые раньше не встречались, т.е. не были отмечены). Если встретится целевая вершина, то печатаем вычисленную длину и выход. Если очередь опустела, и мы не встретили целевую вершину, то пути нет, и программа печатает -1.

Код программы на языке Си++

```
#include <iostream>
#include <queue>
#include <algorithm>
#include <cstring>
#include <cstdlib>
using namespace std;
#pragma once
const int INF=2000000;
int n;
```

```
bool **mm;
int MinPath(int a, int b)
{
  if (a==b) return 0;
  int *vis=new int[n];
  for (int i = 0; i < n; i++)
          vis[i]=INF;
  queue<int> q;
  q.push(a);
  vis[a]=0;
  for(;!q.empty();)
  {
          int v=q.front();
          q.pop();
          for(int j=0;j<n;j++)
                  if (mm[v][j] && vis[j]==INF)
                  {
                          if (j==b)
                          {
                                  delete[]vis;
                                  return vis[v]+1;
                          }
                  }
                          q.push(j);
                          vis[j]=vis[v]+1;


  }
  delete[]vis;
  return -1;
}
namespace My1Graph_CPP {

  using namespace System;
  using namespace System::ComponentModel;
  using namespace System::Collections;
  using namespace System::Windows::Forms;
  using namespace System::Data;
  using namespace System::Drawing;
  using namespace std;
  /// <summary>
  /// Сводка для Form1
  ///
  /// Внимание! При изменении имени этого класса необходимо также изменить
  ///          свойство имени файла ресурсов ("Resource File Name") для средства
компиляции управляемого ресурса,
  #pragma region Windows Form Designer generated code
#pragma endregion
  private: System::Void button1_Click(System::Object^  sender, System::EventArgs^
e)
                  {
                          n=Convert::ToInt32(textBox1->Text);
```

```
                mm = new bool*[n];
                for (int i = 0;i < n; i++)
                        mm[i]=new bool [n];
                array<String^>^ lines = richTextBox1->Lines;
                int count = lines->Length;

                for (int i = 0; i < n; i++)
                        for (int j = 0; j < n; j++)
                                {
                                        if (lines[i][j]=='0') mm[i][j]=0;
                                        else mm[i][j]=1;
                                };
                int a=Convert::ToInt32(textBox2->Text);
                int b=Convert::ToInt32(textBox3->Text);
                int k=0;
                /*for (int i = 0; i < n; i++)
                        for (int j = 0; j < n; j++)
                                if(mm[i][j]==0) k++;*/
                textBox4->Text=Convert::ToString(MinPath(a,b));
        }
};
}
```

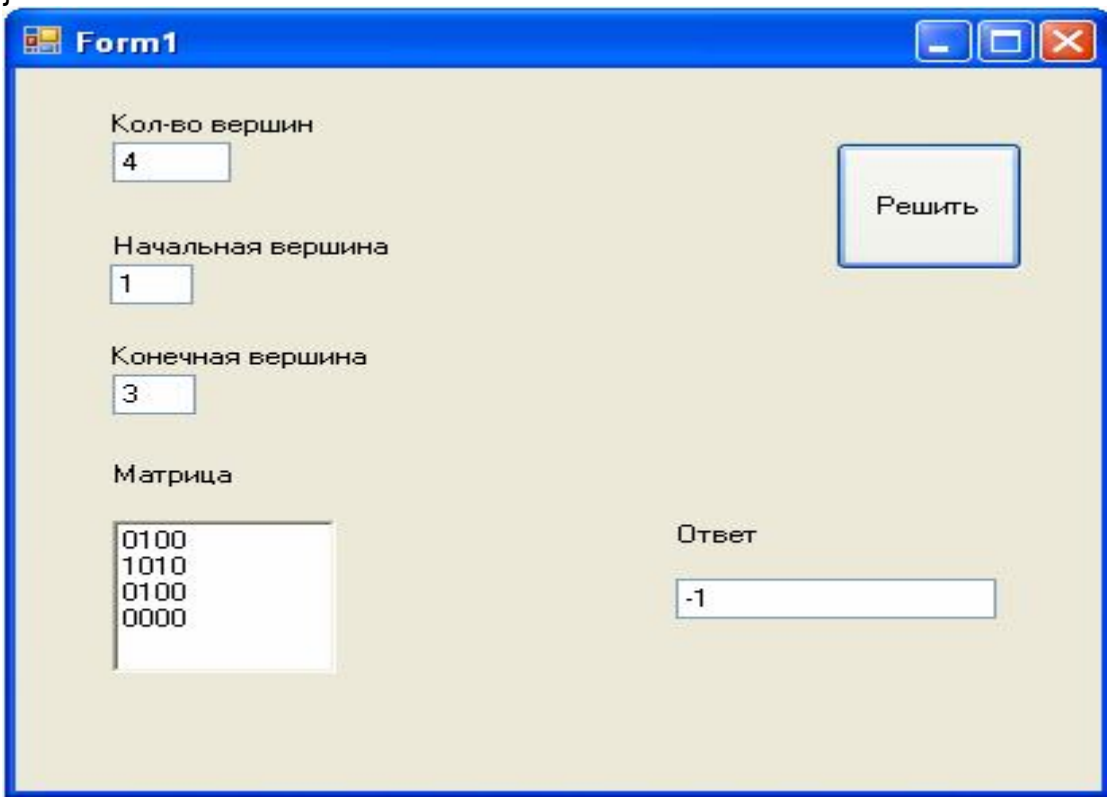

Рис. 7. Визуализация работы программы с графом.

Задача 2. Граф в Си#.

Для неориентированного графа (без петель и кратных рёбер) задаётся число вершин и квадратная матрица связности из 0 и 1. Написать программу для нахождения и печати кратчайшего пути от одной заданной вершины (начальной) до другой (целевой). Печатать длину пути или -1, если пути нет. Также напечатать последовательный список вершин, через которые проходит

кратчайший путь.

Код программы на языке Си#.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace _2.Graph_CS
{
   public partial class Form1 : Form
   {

      public const int INF = 2000000000;
      public static bool[,] e1;
      public static int n;
      public Form1()
      {
         InitializeComponent();
      }
      public static int MinPath(int a, int b)
      {
         if (a == b)
            return 0;

         int[] vis = new int[n];
         for (int i = 0; i < n; i++)
            vis[i] = INF;

         Queue<int> q = new Queue<int>();
         q.Enqueue(a);
         vis[a] = 0;

         for (; q.Count != 0; )
         {
            int v = q.Peek(); q.Dequeue();
            for (int j = 0; j < n; j++)
               if (e1[v, j] && vis[j] == INF)
               {
                  if (j == b)
                     return vis[v] + 1;
                  q.Enqueue(j);
                  vis[j] = vis[v] + 1;
               }
         }
         return -1;
      }
      private void button1_Click(object sender, EventArgs e)
```

```
        {
            n = int.Parse(textBox1.Text);
            e1 = new bool[n, n];
            for (int i = 0; i < n; i++)
            {
                string s = richTextBox1.Lines[i];
                for (int j = 0; j < n; j++)
                    if (s[j] == '1')
                        e1[i, j] = true;
                    else
                        e1[i, j] = false;
            }
            int a = int.Parse(textBox2.Text) - 1;
            int b = int.Parse(textBox3.Text) - 1;
            textBox4.Text = Convert.ToString(MinPath(a, b));
        }
    }
}
```

ТЕМА 6: ГРАФИЧЕСКИЕ СОБЫТИЯ

Цель.

Научиться использовать графические примитивы для рисования (с иллюзией движения) в оконном интерфейсе.

Ключевые слова.

Управление событиями, графические примитивы и их свойства и методы

Основные понятия.

События – изменения состояний и перемещения объектов и предметов в зависимости от времени. События в классах – это методы в классах, посредством которых один объект класса (источник события) может сообщить другому объекту (получателю события) об изменении своего состояния.

Задача 1. Самолетик (Си++).

Создать мультик со следующим сюжетом: простой самолётик начинает полёт в первом окне и продолжает его во втором. От края окна отскакивает, переходит на другой эшелон и летит обратно и т.д. в цикле.

Код программы на языке Си++.

```
// Тема 6 задача 1.cpp: главный файл проекта.

#include "stdafx.h"
# using <System.dll>
# using <System.Windows.Forms.dll>
# using <System.Drawing.dll>

using namespace System;
using namespace System::Collections;
```

```
using namespace System::Windows::Forms;
using namespace System::Drawing;
using namespace System::Threading;

# define iabs(a) ( (a) < 0 ? (-(a)) : (a) )

const int DelayTime = 300;   // задержка между перерисовками
const int Delta = 20;        // сдвиг самолёта
const int H1 = 50;           // высоты полёта
const int H2 = 100;
const int KP = 5;            // MAX number of planners

delegate void Del (int, int, int, int); // Делегат события

ref class Planer
{
public :
   event Del ^ ev;
   Thread ^ t;
   bool active, life;
   Color col;
   int w, x, dx, y, num, nump;

   // Конструктор
   Planer (int n, bool act, int W, int N)
   {
      num = n;         //
  nump = N;
      active = act;    //
      w = W;
      dx = Delta;        // delta step
      x = 0;             // start point
      y = H1;
      life = true;
  switch ( N )
    {
    case 1 : col = Color::Black; break;
    case 2 : col = Color::Blue; break;
    case 3 : col = Color::Red; break;
    case 4 : col = Color::Yellow; break;
    case 5 : col = Color::Green; break;
  }
      t = gcnew Thread (gcnew ThreadStart (this, &Planer::Move));
      t->Start ( );
   }

private :
   void Move ( )
   {
      while ( life )
      {
         if ( active )
```

```
      {
        x += dx;
        if ( dx > 0 && x >= w && num==1 || dx < 0 && x <= 0 && num == 2 )
        {
          active = false;           //  перелёт из окна в другое
          y = num == 1 ? H1 : H2;
        }
        else if ( dx>0 && x >= w && num==2 || dx<0 && x <= 0 && num==1 )
        {
          dx = -dx;         // отскок от края окна и обратный разворот
          y = num == 1 ? H1 : H2;
        }
        this->ev (x, dx, y, nump);
        Thread::Sleep (DelayTime);
      }
    }
  }

public :
  void PlanerHandler (int X, int DX, int Y, int NUM)
  {
    if ( num == 1 && X <= 0 && DX < 0 && nump == NUM )// из 2-го окна в 1-е
      x = w, dx = -Delta, active = true, y = Y;
    else if ( num == 2 && X >= w && DX > 0 && nump==NUM )// из 1-го во 2-е
      x = 0, dx = +Delta, active = true, y = Y;
    else                                  // остаёмся в своём окне
      ;
  }

public :
  void Finish ()
  {
    life = false;
  }
};

ref class Wind : Form
{
public :
  int x, y;
  int num;
  ArrayList ^ p; // Массив самолётов

public :
  Wind (int n, bool a)
  {
    if ( n == 1 )
      this->Text = "Window 1";
    else
      this->Text = "Window 2";
    num = 0;
    p = gcnew ArrayList ();
```

```
    this->Visible = true;
    this->Size = * gcnew System::Drawing::Size (240, 300);
  }
  // Обработчик события
public :
  void Hand (int X, int DX, int Y, int N)
  {
    Invalidate ();
  }

protected :
  virtual void OnPaint (PaintEventArgs ^ e) override
  {
    for ( int i=0; i < num; i++ )
  {
      Planer ^ pp = (Planer ^) p [i];
    if ( pp->active )
      {
  e->Graphics->FillEllipse (gcnew SolidBrush(pp->col),pp->x,pp->y,30,20);
  e->Graphics->FillEllipse (gcnew SolidBrush(Color::Red),pp->x,pp->y,10,10);
      }
   }
  }
};

ref class Manager : Form
{
public :
  Wind ^ w1, ^ w2;
  Button ^pButAdd;        // Дескриптор кнопки ADD
  Button ^pButDel;        // Дескриптор кнопки ADD
  Button ^pButOK;         // Дескриптор кнопки OK

  Manager ()
  {
    this->Text = "Main Window";
    w1 = gcnew Wind (1, true);        // Создать первое окно
    w2 = gcnew Wind (2, false);       // Создать второе окно
    pButAdd= gcnew Button;            // Создать объект кнопки
    pButAdd -> Location = Drawing::Point (70,10);  // Разместить
    pButAdd -> Size = Drawing::Size (40, 20);      // Размер
    pButAdd -> Text = "ADD";          // Поместить текст в кнопку
    Controls -> Add (pButAdd);                // Добавить кнопку к форме
                // Подписать обработчик на событие Click кнопки
    pButAdd -> Click += gcnew System::EventHandler (this, &Manager:: Add);

    pButDel= gcnew Button;                    // Создать объект кнопки
    pButDel -> Location = Drawing::Point (120,10);  // Разместить
    pButDel -> Size = Drawing::Size (40, 20);      // Размер
    pButDel -> Text = "DEL";              // Поместить текст в кнопку
    Controls -> Add (pButDel);            // Добавить кнопку к форме
```

```
                // Подписать обработчик на событие Click кнопки
    pButDel->Click += gcnew System::EventHandler (this, &Manager:: Delete);

    pButOK= gcnew Button;                       // Создать объект кнопки
    pButOK -> Location = Drawing::Point (170,10);  // Разместить
    pButOK -> Size = Drawing::Size (40, 20);       // Размер
    pButOK -> Text = "OK";                  // Поместить текст в кнопку
    Controls -> Add (pButOK);                // Добавить кнопку к форме
                // Подписать обработчик на событие Click кнопки
    pButOK -> Click += gcnew System::EventHandler (this, &Manager:: OK);
  }

  void Add (Object ^obj, EventArgs ^args)
  {
    if ( w1->num < KP )
   {
     w1->num++; w2->num++;
     Planer ^ pp1 = gcnew Planer (1, true, w1->Size.Width, w1->num);
     Planer ^ pp2 = gcnew Planer (2, false, w2->Size.Width, w2->num);

     pp1->ev += gcnew Del (w1, &Wind::Hand);
        pp1->ev += gcnew Del (pp2, &Planer::PlanerHandler);
     w1->p->Add (pp1);

        pp2->ev += gcnew Del (w2, &Wind::Hand);
        pp2->ev += gcnew Del (pp1, &Planer::PlanerHandler);
     w2->p->Add (pp2);

     Invalidate (); // Перерисовать область клиента окна
   }
  }

  void Delete (Object ^obj, EventArgs ^args)
  {
    if ( w1->num > 0 )
  {
    w1->num--; w2->num--;
    Planer ^ p1 = (Planer ^) w1->p [w1->num];
       Planer ^ p2 = (Planer ^) w2->p [w2->num];
       w1->p->Remove (p1);
       w2->p->Remove (p2);
  }
   }

  void OK (Object ^obj, EventArgs ^args)
  {
   this->Close ();
  }
};

int main(array<System::String ^> ^args)
{
```

```
System::Windows::Forms::Application::Run (gcnew Manager ());

    return 0;
}
```

Визуализация работы программы.

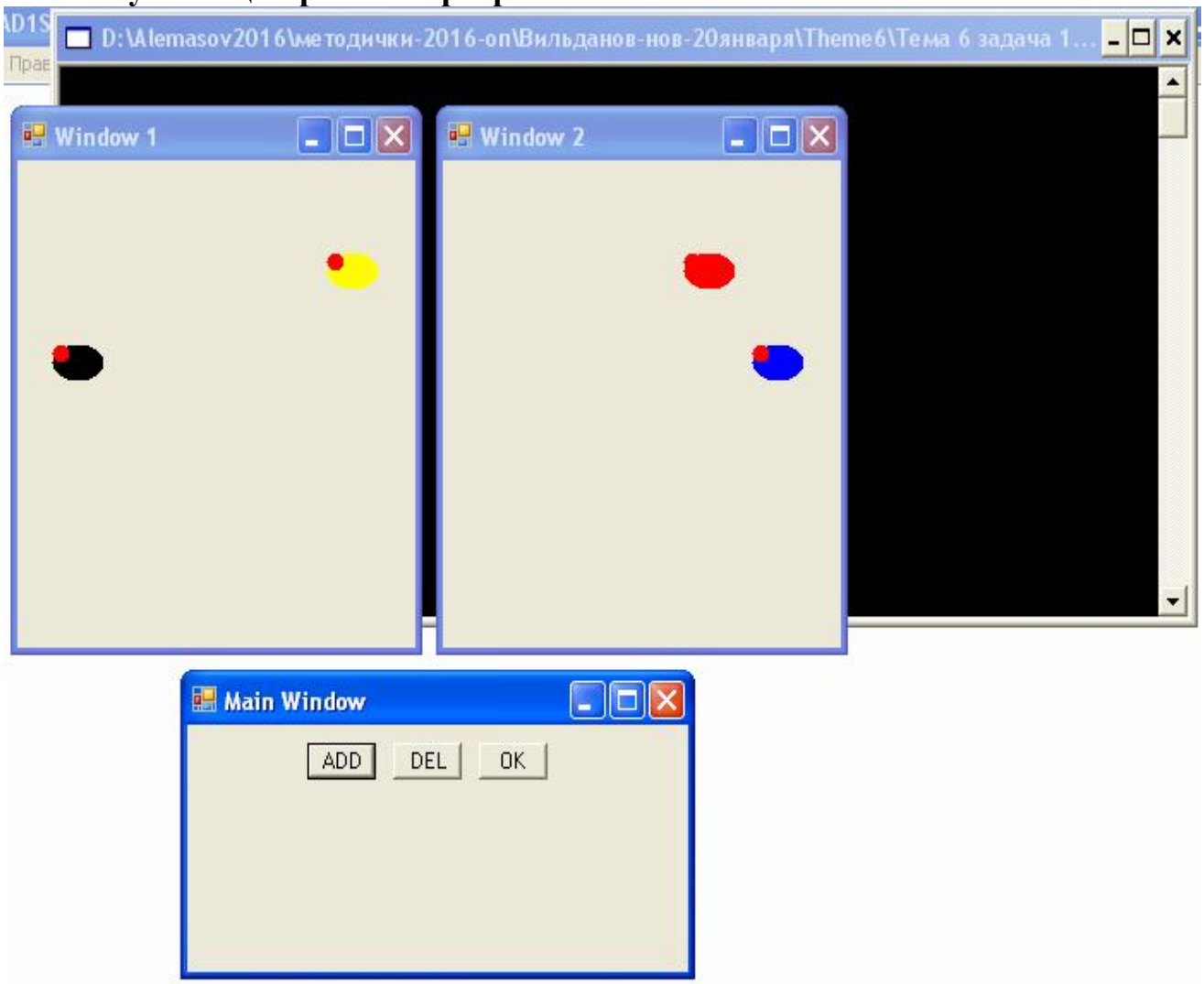

Рис. 8. Визуализация работы программы с самолетиками.

Задача2. Грузовик Си#.

Сюжет: едет машина и просыпает навоз на землю, после проливного дождя под лучами палящего солнца вырастают цветы.

Примечание.

Необходимо иметь файлы с картинками, использующиеся в этой программе, и поместить их в каталог, где будет находиться исполняемый файл (или указать их точное местоположение – папки и диск).

Код программы на языке Си#.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace _2.Машина2
{
    public partial class Form1 : Form
    {
        int x, y, dx;
        int ti = 0;
        bool poz, rain;
        public Form1()
        {
            InitializeComponent();
            x = 50; y = 150;
            dx = 10;
            ti = 0;
            poz = true;
            rain = false;
            timer1.Start();
        }
        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Font f = new Font("Arial", 20);
            Pen p = new Pen(Color.Yellow, 2);
            if (ti < 50)
            {
                x += dx;
                poz = !poz;

                //солнышко
                e.Graphics.FillPie(Brushes.Yellow, 0, 0, 100, 100, 0, 360);

                // машина
                e.Graphics.FillRectangle(Brushes.Moccasin, x, y, 180, 50);
                e.Graphics.DrawRectangle(Pens.Black, x + 120, y - 20, 30, 20);
                e.Graphics.DrawEllipse(Pens.Black, x + 20, y + 50, 50, 50);
                e.Graphics.DrawEllipse(Pens.Black, x + 110, y + 50, 50, 50);

                // крутятся колёса
                if (poz)
                {
                    e.Graphics.DrawLine(Pens.Blue, x + 20, y + 75, x + 70, y + 75);
                    e.Graphics.DrawLine(Pens.Blue, x + 45, y + 50, x + 45, y + 100);
                    e.Graphics.DrawLine(Pens.Blue, x + 110, y + 75, x + 160, y + 75);
                    e.Graphics.DrawLine(Pens.Blue, x + 135, y + 50, x + 135, y + 100);
                    e.Graphics.DrawLine(p, 50, 100, 50, 128);
                    e.Graphics.DrawLine(p, 105, 50, 128, 50);
                    e.Graphics.DrawLine(p, 85, 85, 105, 105);
                }
                else
```

```
    {
        e.Graphics.DrawLine(Pens.Blue, x + 28, y + 58, x + 62, y + 92);
        e.Graphics.DrawLine(Pens.Blue, x + 28, y + 92, x + 62, y + 58);
        e.Graphics.DrawLine(Pens.Blue, x + 118, y + 58, x + 152, y + 92);
        e.Graphics.DrawLine(Pens.Blue, x + 118, y + 92, x + 152, y + 58);
    }
}
else if (ti < 70)
{
    if (rain)
    {
        e.Graphics.FillEllipse(Brushes.Aqua, 310, y - 30, 5, 10);
        e.Graphics.FillEllipse(Brushes.Aqua, 330, y - 30, 5, 10);
        e.Graphics.FillEllipse(Brushes.Aqua, 350, y - 30, 5, 10);
        e.Graphics.FillEllipse(Brushes.Aqua, 320, y - 10, 5, 10);
        e.Graphics.FillEllipse(Brushes.Aqua, 340, y - 10, 5, 10);
        Bitmap image = new Bitmap(Properties.Resources.Cloude);
        e.Graphics.DrawImage(image, 10 + 170, y - 205);
        e.Graphics.FillEllipse(Brushes.Black, 610, y - 30, 5, 10);
        e.Graphics.FillEllipse(Brushes.Black, 630, y - 30, 5, 10);
        e.Graphics.FillEllipse(Brushes.Black, 650, y - 30, 5, 10);
        e.Graphics.FillEllipse(Brushes.Black, 620, y - 10, 5, 10);
        e.Graphics.FillEllipse(Brushes.Black, 640, y - 10, 5, 10);

        Bitmap image1 = new Bitmap(Properties.Resources.Cloude);
        e.Graphics.DrawImage(image, 10 + 470, y - 205);
    }
    else
    {
        e.Graphics.FillEllipse(Brushes.Aqua, 310, y - 10, 5, 10);
        e.Graphics.FillEllipse(Brushes.Aqua, 330, y - 10, 5, 10);
        e.Graphics.FillEllipse(Brushes.Aqua, 350, y - 10, 5, 10);
        e.Graphics.FillEllipse(Brushes.Aqua, 320, y - 30, 5, 10);
        e.Graphics.FillEllipse(Brushes.Aqua, 340, y - 30, 5, 10);
        e.Graphics.FillEllipse(Brushes.Black, 610, y - 30, 5, 10);
        e.Graphics.FillEllipse(Brushes.Black, 630, y - 30, 5, 10);
        e.Graphics.FillEllipse(Brushes.Black, 650, y - 30, 5, 10);
        e.Graphics.FillEllipse(Brushes.Black, 620, y - 10, 5, 10);
        e.Graphics.FillEllipse(Brushes.Black, 640, y - 10, 5, 10);
    }
    rain = !rain;
}
else
{
    Bitmap image = new Bitmap(Properties.Resources.Tulip);
    e.Graphics.DrawImage(image, 50 + 250, y - 50);
    Bitmap image1 = new Bitmap(Properties.Resources.Tulip);
    e.Graphics.DrawImage(image, 50 + 550, y - 50);
    e.Graphics.FillPie(Brushes.Yellow, 0, 0, 100, 100, 0, 360);
    e.Graphics.DrawLine(p, 50, 100, 50, 128);
    e.Graphics.DrawLine(p, 105, 50, 128, 50);
    e.Graphics.DrawLine(p, 85, 85, 105, 105);
```

```
        }
    }
    private void timer1_Tick(object sender, EventArgs e)
    {
        if (ti < 100)
        {
            ti++;
            Refresh();
        }
        else
            timer1.Stop();
    }
  }
}
```

Визуализация работы программы.

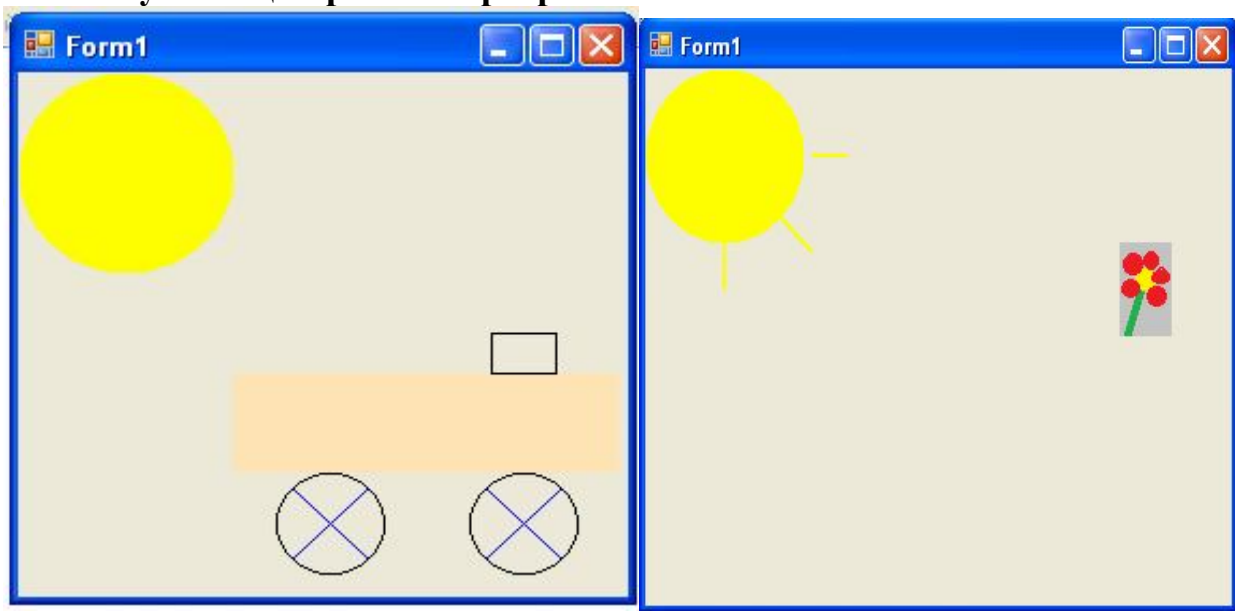

Рис. 9. Визуализация работы программы с грузовиком.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Пышкин, Е.В. Основные концепции и механизмы объектно-ориентированного программирования: Учебное пособие [Текст] / Е.В. Пышкин. – СПб.: БХВ-Петербург, 2005. – 130 с.

2. Культин, Н.Б. Microsoft Visual C++ в задачах и примерах [Текст] / Н.Б. Культин – СПб.: БХВ-Петербург, 2010. – 272 с.

3. Фаронов, В.В. Программирование на языке C# [Текст] / В.В. Фаронов – СПб.: Питер, 2007. – 240 с.

4. Шилдт, Г., C# 4.0: полное руководство [Текст]: пер. с англ. / Герберт Шилдт. – М., ООО «ИД Вильямс», 2011. - 1056 с.

Абайдуллин Булат Равилевич **Тагиров** Равиль Равгатович

ЛАБОРАТОРНЫЕ РАБОТЫ ПО КУРСУ ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Дизайн обложки ***М.А. Ахметов***

Подписано в печать 14.05.2017. Бумага офсетная. Печать цифровая. Формат 60x84 1/16. Гарнитура «Times New Roman». Усл. печ. л. . Тираж экз. Заказ

Отпечатано с готового оригинал-макета в типографии Издательства Казанского университета

420008, г. Казань, ул. Профессора Нужина, 1/37 тел. (843) 233-73-59, 233-73-28