Алгоритмы и алгоритмические языки

Лисид Лаконский

2023 год

Содержание

1 Алгоритмы и алгоритмические языки — занятие №1

dll — **динамически подключаемая библиотека**
Для создания dll в Visual Studio необходимо выбрать шаблон «**библиотека классов**»

1.1 Проект «вычисление суммы квадратов двух чисел»

1.1.1 Динамическая библиотека

Для решения поставленной задачи нам необходимо реализовать: **метод нахождения суммы квадратов двух чисел, метод для ввода данных, метод для вывода данных**

```
using System;
using System.Windows.Forms;

namespace ClassLibrary1
{
    public class Class1 {
        public static int Vvod(TextBox t)
        {
            return Convert.ToInt(t.Text);
        }
        public static int Vyvod(TextBox t, int c)
        {
            t.Text = Convert.ToString(c);
        }
        public static int Sum_kv(int x, int y)
        {
            int res = x * x + y * y;
            return res;
        }
    }
}
```

В настоящих проектах рекомендуется давать классам, методам, переменным и так далее **нормальные имена**, а не как в коде выше.

1.1.2 Основная программа

Помещаем в форму с помощью вкладки «элементы управления» следующие компоненты: три надписи (**label**), два поля ввода текста (**textbox**) и одну кнопку (**button**)
Код собственно программы:

```
using System;
// Прочие юзинги...
using LibraryClass1;

namespace ClassProgram1 {
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            int a = Class1.Vvod(textBox1);
            int b = Class1.Vvod(textBox2);
```

```
            int y = Class1.Sum_kv(a, b);
            Class1.Vyvod(textBox3, y);
        }
    }
}
```

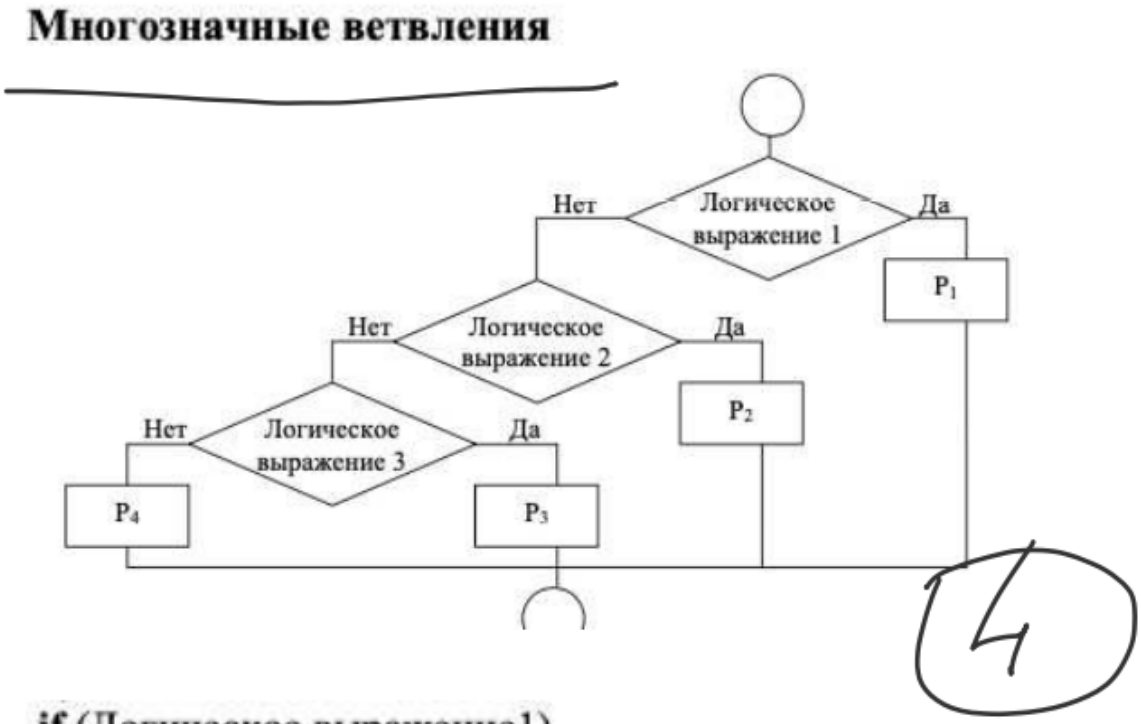
2 Алгоритмы и алгоритмические языки — занятие №2

2.1 Разветвляющийся алгоритм

Алгоритм называется разветвляющимся, если последовательность выполнения его шагов изменяется в зависимости от некоторых условий (логических выражений, которые могут принимать одно из двух значений – **истина** и **ложь**).
Пример с паролем — **простой условный оператор**:


```
if (Логическое выражение)
        P1 ;
else
        P2;
```

Пример с возрастом — **многозначные ветвления**:




```
if (Логическое выражение1)
        P1;
  else
    if (Логическое выражение2)
            P2;
      else
        if (Логическое выражение3)
                P3;
          else
                P4 ;
```

Таблица 6.1 – Операции отношения (сравнения)

Операции отношения	Значение
<	*Меньше, чем*
>	*Больше, чем*
<=	*Меньше или равно*
>=	*Больше или равно*
==	*Равно*
!=	*Не равно*

Таблица 6.2 – Логические операции

Логическая операция	Результат
`!` (НЕТ, отрицание)	Преобразует значение **true** в **false** и наоборот – **false** в **true**
`&&` (И, конъюнкция)	Результат операции **true**, если оба операнда имеют значение **true**
`||` (ИЛИ, дизъюнкция)	Результат операции **true**, если хотя бы один из операндов имеет значение **true**

Остановимся на распространенных *ошибках* при записи условных операторов:

- Нередко *в выражениях вместо проверки на равенство* (==) *ошибочно записывают простое присваивание* (=), например, `if (a=1) b=0;`
  Синтаксической ошибки нет, так как операция присваивания формирует результат, который и оценивается на равенство/неравенство нулю. Здесь присваивание нуля переменной `b` будет выполнено независимо от значения переменной `a`.
- Второй распространенной ошибкой является *неверная запись проверки на принадлежность диапазону*. Например, чтобы проверить условие `n<x<m`, нельзя записать его в условном операторе непосредственно – `if (n<x<m)…`, так как будет выполнено сначала сравнение `n<x`, а его результат (`true` или `false`, преобразованное в тип `int`), будет сравниваться со значением `m`. Правильный способ записи: `if (n<x && x<m)…`
- Еще важное замечание. В условном выражении оператора `if` возможно определение локальной переменной, например, `if (int a=f())…` Область видимости такой переменной распрст-раняется на обе ветви `if...else`.

3 Алгоритмы и алгоритмические языки — занятие №3

В лабораторной работе следует использовать три TextBox для организации ввода данных, кнопку, табличный компонент для вывода таблицы. В таблице должно быть два столбца: «икс» и «игрек».
Виды:

1. Сумма
2. Количество
3. Максимальное и минимальное
4. Произведение
5. Ничего кроме таблицы

Пять методов. В dll-библиотеке должны быть методы за все лабораторные работы — ввод, вывод, метод расчета выражений.
Метод для вывода результатов табулирования функции в табличный компонент:

```
public static void _ (double x, double y, DataGridView DGV) {
    DGV.Rows.Add(x.ToString(), y.ToString());
}
```

Схемы должны быть выполнены согласно требованиям ГОСТ.
Циклы — с известным и неизвестным количеством повторений.

4 Алгоритмы и алгоритмические языки — занятие №4

Через какое-то время будет ещё одна контрольная работа по 3 и 4 лабораторной работе.
«Создание приложения с итеративной циклической структурой»
Виды циклов:

1. for
2. while
3. do.. while

Выведение рекуррентной формулы.
При суммировании ряда необходимо решать следующие задачи:

1. Свести вычисления к простейшим арифметическим операциям
2. Уменьшить число этих операций и время расчёта
3. Уменьшить погрешность округлений

Источники погрешностей:

1. Ошибки в исходных данных

Задачи сокращения количества операций и уменьшения погрешности вычислений решает рекуррентная формула, позволяющая вычислить значение очередного члена ряда, используя уже найденные значения предыдущего.
Рекуррентная формула имеет вид:

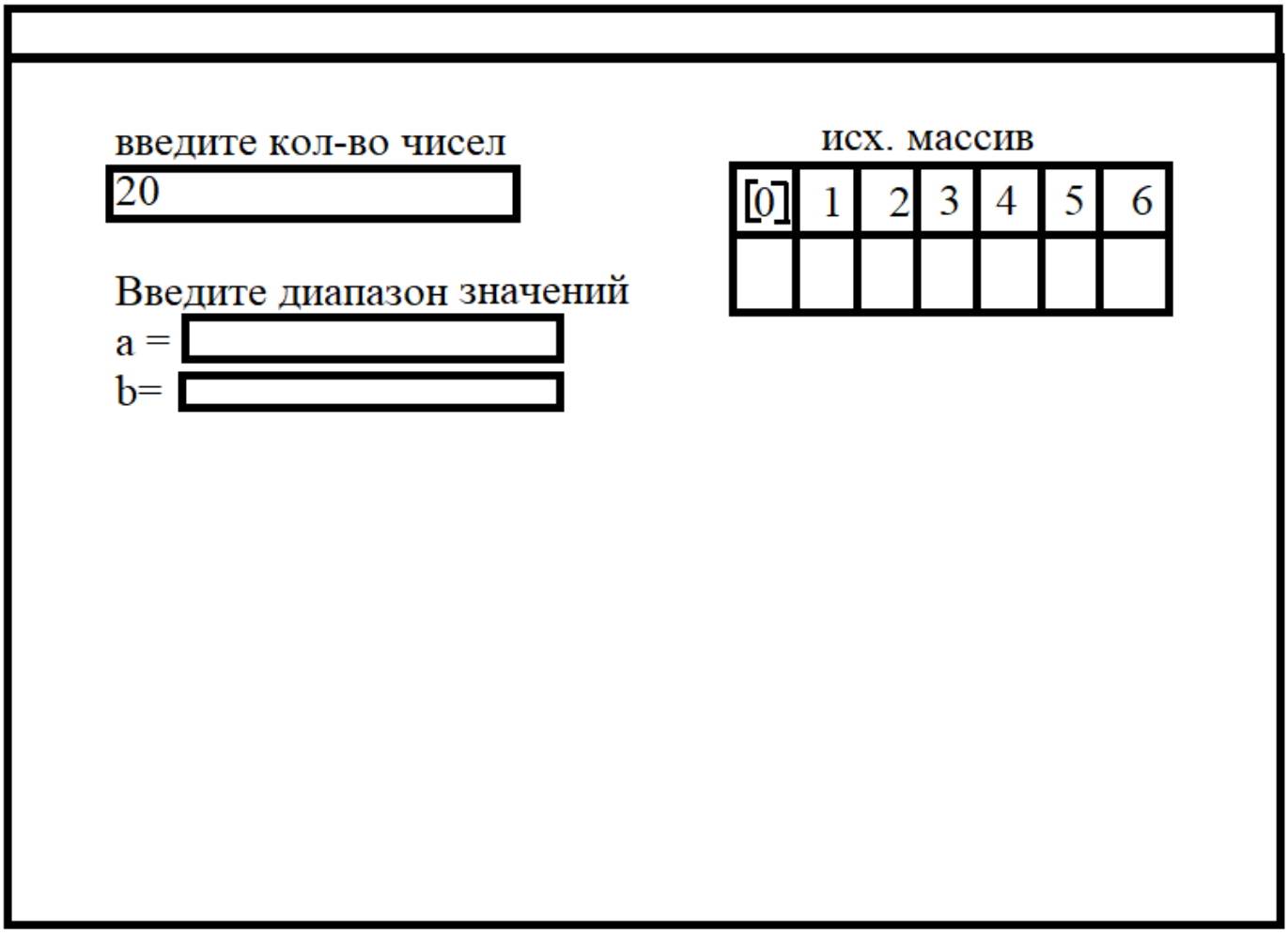
$$a_{n+1} = a_n * \ln n!$$

5 Алгоритмы и алгоритмические языки — занятие №5

Синтаксис определения одномерного массива: тип данных[] имя массива = new тип данных[общее количество элементов массива]
Второй вариант: тип данных[] имя массива

Для вывода должны использоваться элементы DataGridView и MessageBox, TextBox использовать запрещено.

ColumnHeadersVisible — параметр для заголовков столбцов.
В таблице индексы должны располагаться в нулевой строке.
AutoScaleColumnsMode -> AllColls, AutoSizeRowsMode, DefaultCellStyle.
Всё это для исправления негибкости таблицы под размеры DataGridView.



Дополнительной подпрограммы для результирующего массива не нужно — также использовать output_mas.
Для решения задачи чаще всего достаточно одного метода.

1. Enter_mas — генерирует элементы массива
2. Output_mas – вывод элементов массива на элемент DataGridView
3. Метод ввода – для конвертации строкового значения в числовое

На примере задачи нахождения чисто положительных чисел: Методы выше

```
public static int kol(int[] mas, int length) {
    int count = 0;
    for (int i = 0; i != length; ++i) {
        if (mas[i] > k)
            count++;
    }
    return count;
}
```

Вопрос для зачёта: какие в C# какие бывают методы? Вопрос(ы) могут повторяться. Если метод считает одно значение – использовать return.

6 Алгоритмы и алгоритмические языки — занятие №6

Проблемы с set_mas не такие страшные – изменять можно, пусть и желательно соблюдать структуру.

Шестая работа – продолжение пятой работы. В задание шестой работы вставить задание из лабораторной работы №5.
Перечисляем алгоритмы, которые нам необходимо реализовать.
Перечисляем методы (по подсчётам Сергея Ростиславовича – 3-4). Вывод в MessageBox. Рисовать прямоугольником такой вывод – неправильно, он рисуется параллелограммом (вывод по ГОСТу), где писать не всю строчку кода, а просто «Вывод // Что выводится, необязательно со словом вывод, можно просто «Нет таких чисел».

Ничего переставлять и убирать в алгоритмах не надо. Блок-схемы там не всегда верны, периодически сделаны по устаревшим ГОСТам.
Циклы тут часто начинаются с единицы. Учесть, что индексация массивов начинается с нуля, так что, очевидно, слегка изменить код под это.
К каждому методу давать комментарии с кратким пояснением, что это такое.

Если в пятой лабораторной работе массив выходит результирующим – то сортировать алгоритмами именно его. В противном случае – изначальный. Сортировка и изначальная задача, желательно, по одной кнопке. Методы излагать в логической последовательности.
Основная ошибка – вылет индекса за пределы размера массива.

```
public static void Insert(int [] mas, int length, int poz, int chislo) {
    for (int i = length - 1; i >= poz; i--) { // цикл по убыванию
        mas[i + 1] = mas[i];
        mas[i] = chislo;
    }
}
```

Событийная кнопка:

```
Button1_Click() {
    Int length = ...
    Int[] arr = new int[length];
    Int A = ...
    Int B = ...
    Enter_mas();
    Output_mas();
    String chislo_ = Microsoft.VisualBasic.Interaction.InputBox(
        "Введите число для добавления в массив", "Ввод", "", ...
    );
    String poz = Microsoft.VisualBasic.Interaction.InputBox...
    Class1.Insert(arr, length,  poz, chislo_);
    Class1.Output_mas(arr, length + 1, dataGridView2);
}
```

Если VisualBasic не работает, то нужно добавить ссылку на него в проект.

```
public static void Resize(ref int[] mas, int new_size) {
    Int[] newArr = new int[newSize];
    for (int i = 0; i < arr.length; i++)
        newArr[i] = mas[i];
    mas = newArr;
}
```

7 Алгоритмы и алгоритмические языки — занятие №7

7.1 Работа с двумерными массивами

Надо относиться к творчески. Некоторым образом можно менять задания, если они как-то слишком неудобны.
Ещё 1 лекция. 7 июня лекции не будет. 6 июня – защита курсовой работой. Чуть раньше – зачёт.

Обязательно скопировать индивидуальные задачи в пункт 4.1, правильными словами описать свой функционал.
Не писать разработку конспекта и системы тестирования как функционал. Лишь доп. задачи и собственную дополнительную функциональность. Оптимизировать пункт 2.3 под это.
В случае ухода на перезачёт – в ЭОС будет прикрепление со сдачей доработок за лето.

Следует создать новую, последнюю на семестр DLL-библиотеку. Второе прикрепление лабораторных работ к 20 мая. Последний этап будет накануне зачёта. Возможно, стоит всё перепечатать. Реализация алгоритмов с двумерными массивами – необязательна к этому этапу, потому что мало времени. Базовые алгоритмы обработки массивов вообще не выложены.

1. $i = j$ — главная диагональ
2. $i > j$ — ниже главной диагонали, $i < j$ — выше главной диагонали
3. $i + j - 1 = n$ — побочная диагональ
4. $i + j - 1 < n$ — выше побочной диагонали, $i + j - 1 > n$ — ниже побочной диагонали

Синтаксис вложенного цикла:

```
for (i = n1; i <= n2; ++i) {
    for(j = m1; j <= m2; ++j) {
        P1;
        P2;
        ...
        Pn;
    }
}
```

Пример вложенного цикла:

```
int a = 5;
for (int i = 1; i != 2; ++i) {
    for (int j = 1; j <= 2; ++j) {
        a = a + 5;
    }
}
```

На зачете может понадобиться построить таблицу с раскрытием изменения параметров в цикле.

Общий вид объявления двумерного массива: тип данных [,] имя массива = new тип данных[кол-во строк, кол-во столбцов];
Пример:

```
int[,] a = new int[4, 4];
```

DataGridView при выводе в интерфейс подтянуть под 5x5 матрицу. Убрать пустые строки и пустые столбцы.
Перед решением задачи, очевидно, убедиться, что массив на экране, и потом решать.
К зачёту скачать visual studio 2010 и потренироваться в нём, потому что если на зачёте дадут что-то писать – будет именно visual studio 2010.
Все работы иметь с собой. Сообщения выводить в MessageBox. Ввод в текстовые поля легче всего делать с помощью InputBox.

Код событийной кнопки:

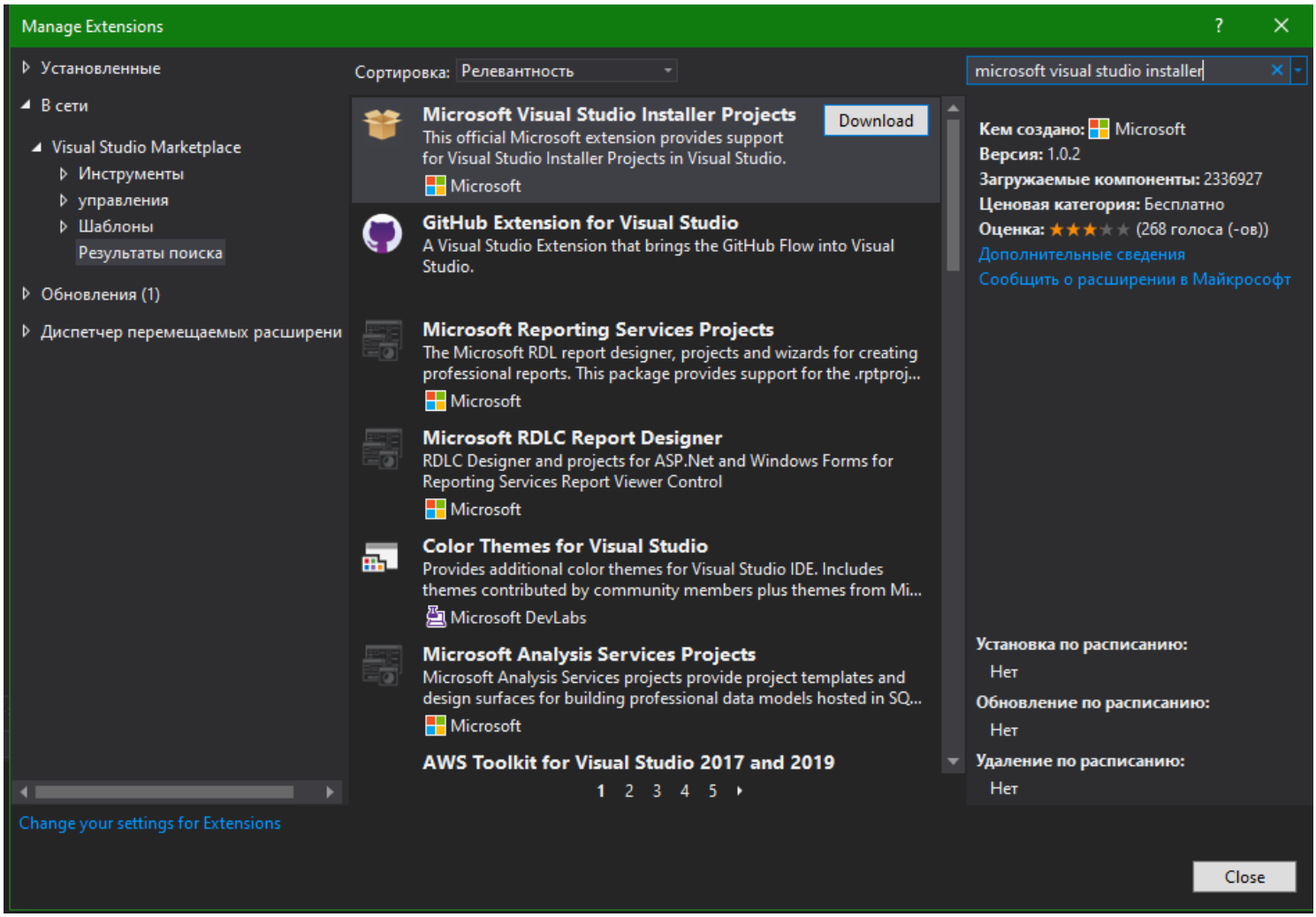
```
Private void button1.Click() {
    int A = Class1.Vvod(textBox1);
    int B = Class1.Vvod(textBox2);
    int [,] arr = new int[A, B];
    Class1.ArrayGenerate(arr, A, B);
    Class1.Output_mas(arr, A, B, dataGridView1);
    int K = Class1.Kol(arr, A, B);
    int[] rezarr = new int[K];
    MessageBox.Show("Кол-во полезных элементов" + K, ...);
    Class1.Set_rezmas(arr, rezarr, A, B, K, out int g_;
    Class1.Output_mas(rezarr, g, dataGridView2);
}
```

Перепись задания – полностью.
Пропуски и недописки в записи во внешние программы – частично дополнять из кода к одномерным массивам, + свои названия и прочее.

8 Алгоритмы и алгоритмические языки — занятие №8

Вместо лекции 07.06 – зачёт у первой группы. Можно подводить итоги.
Основные требования к выпускной квалификационной работе – пятый раздел.
Иметь с собой промежуточные зачеты. Вопросы к зачету: первые 20 легкие, оставшиеся 29 — для злостных прогульщиков. Зачет безоценочный, поэтому не страшно ответить не на все.
С собой иметь dll-шаблон для вывода одномерного и двумерного массива в DataGridView. Очень важно, время строго ограничено, поэтому методы вывода массивов на экран очень сильно упростят жизнь.
Некоторым будут даваться дополнительные задачия. На всякий случай принести электронные проекты курсовых работ на флешке.

Создание setup-файла. Необязательно к курсовой работе, но желательно. В меню Visual Studio пункт «Расширения»—«Управление расширениями»
Нажать в обозревателе решений по проекту и нажать «Добавить проект». В строке «Поиск шаблонов», там найти через ключевые слова setup.



1. Application Folder – папка установки программы
2. User's Desktop – буквально рабочий стол пользователя
3. User's Program Menu – меню в папке пуск

Добавление выходной группы проекта – просто нажать OK. Просьба о создании ярлыка. Потом как-то перетащить его в user desktop. Правой кнопкой мыши и командой app добавляем папку внутрь Program menu, даём ей имя.
Суть записи функционала в 2.3 – описать в том числе и как оно делается. То есть создавая setup-файл – не просто показать работу setup файла, но и заснять процесс его создания.