

**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**  
**Московский технический университет связи и информатики**



**А.И. Волков**

# **Программирование ( Visual C++ )**

**Практикум  
по освоению методов и приемов разработки программ**

**Кафедра «Информатика»**

**Москва – 2020**

Волков А.И. Программирование (Visual C++). Практикум по освоению методов и приемов разработки программ. 2-е изд. – М.: МТУСИ, 2020. – 82 с.

При выполнении приведенных заданий изучаются основные приемы и методы разработки программ в оконной среде с использованием различных элементов управления. Описывается на примерах методика использования различных элементов управления. Целью является ознакомление с интегрированной средой разработки программ Microsoft Visual Studio, а также отработка практических методов разработки программ с использованием объектно-ориентированного и визуального программирования.

Для студентов ВУЗов, обучающихся по направлениям подготовки «Информатика и вычислительная техника», «Информационные системы и технологии», «Прикладная информатика», «Программная инженерия», а также студентов и слушателей других направлений подготовки, изучающих объектно-ориентированные методы программирования в среде операционной системы Windows.

© Волков А.И., 2020  
© Московский технический университет  
связи и информатики, 2020

# Задание на занятие №1.

## Изучение интегрированной среды разработки Microsoft Visual Studio

1. Запустите Visual Studio путем выбора команды Главного меню Windows. Для этого нажатием кнопки **Пуск** на панели задач раскройте Главное меню Windows, выберите в нем строку **Все программы**, затем в появившемся меню 2-го уровня выберите строку **Microsoft Visual Studio**, а в меню следующего уровня щелкните на строке **Microsoft Visual Studio**.

После запуска Visual Studio на экране появляется главное окно среды разработки со Стартовой страницей (**Start Page**) на переднем плане и окном Проводника проекта (**Solution Explorer**), прикрепленным к правому краю главного окна.

2. Для создания нового проекта в меню **File** выберите команду **New Project...** (или в стандартной панели инструментов нажмите кнопку **New Project**, или в поле **Recent Projects** Стартовой страницы выберите команду **Create: Project...**).

В открывшемся диалоговом окне **New Project** в поле **Project types:** выделена строка **Visual C++**, а в поле **Templates:** выделен объект **Windows Forms Application** (если это не так, то выделите их). В поле **Name:** (Имя) указано – **<Enter\_name>**, замените содержимое этого поля на **FFirst** (имя создаваемого проекта). При этом аналогично изменится и содержимое поля **Solution Name:**. В поле **Location:** введите путь к папке для размещения нового проекта. Чтобы в случае отсутствия такой папки она была создана, установите флажок **Create directory for solution**. Затем щелчком мыши по кнопке **OK** создайте новый проект.

В результате внутри главного окна среды разработки появится страница Конструктора форм **Form1.h [Design]** с пустой формой **Form1** будущего приложения, а справа под окном Проводника проекта (**Solution Explorer**) появится еще и окно Свойств (**Properties**).

3. Щелчком мыши по форме выделите ее. В окне Свойств (**Properties**) для выбранного здесь объекта **Form1** измените значение свойства **Text** (Заголовок) с **Form1** на **Моя первая программа**. Для этого в строке, соответствующей свойству **Text** этого объекта, щелчком мыши по его значению **Form1** поместите туда мигающий текстовый курсор, а затем, используя клавиши управления курсором, а также клавиши **<Backspace>** и **<Delete>**, удалите старое значение **Form1** и введите новое – **Моя первая программа**. В результате соответственно изменится заголовок формы.

Аналогичным образом установите два значения свойства **Size** (**Width** – ширина и **Height** – высота), определив их соответственно равными **400** и **200** (пикселей). Если поля **Width** и **Height** не видны, нажмите на значок «+» слева от поля **Size**.

Размеры формы можно изменить также захватом и перемещением (с помощью мыши) соответствующих маркеров (маленьких белых

квадратиков) на границах формы. При их перемещении автоматически изменяются и значения свойства **Size**.

Если при переходе в поле значения какого-либо свойства в нем справа появляется кнопка со стрелкой вниз, то вводить значение этого свойства не нужно – его следует выбирать среди значений списка, раскрывающегося указанной кнопкой.

Установите свойства формы в соответствии с таблицей 1.1.


Таблица 1.1. Свойства формы и их значения

Свойство	Описание	Значение
<b>Text</b>	Заголовок формы	<b>Моя первая программа</b>
<b>FormBorderStyle</b>	Стиль обрамления формы	<b>Sizable</b> (изменяемая рамка, а в области заголовка размещены кнопка вызова системного меню, заголовок формы и 3 кнопки управления окном)
<b>StartPosition</b>	Расположение формы на экране после ее открытия	<b>CenterScreen</b> (форма располагается в центре экрана)
<b>WindowState</b>	Состояние окна (размер формы при ее открытии)	<b>Normal</b> (размеры определяются свойствами формы)
<b>Size.Height</b>	Высота формы	<b>400</b>
<b>Size.Width</b>	Ширина формы	<b>200</b>

4. Переименуйте разработанную только что форму, для этого в окне Проводника проекта (**Solution Explorer**) выделите файл формы **Form1.h**, а в окне Свойств (**Properties**) проекта измените значение свойства (**Name**) с **Form1.h** на **FFirst.h**.

Учитывая, что файл формы указывается в файле исходного кода **FFirst.cpp**, то внесем изменение и туда. Для этого в окне Проводника проекта (**Solution Explorer**) выделите файл исходного кода **FFirst.cpp**, а затем щелчком правой кнопки мыши раскройте его контекстное меню и в нем выберите пункт **View Code** (или нажмите кнопку **View Code** (Показать код) на панели инструментов этого окна). В главном окне среды разработки появится страница редактора кода файла **FFirst.cpp**, в котором измените директиву **#include "Form1.h"** на **#include "FFirst.h"**.

5. Сохраните созданный проект, для чего в меню **File** (Файл) выберите команду **Save All** (Сохранить все). Страницу редактора кода файла **FFirst.cpp** теперь можно закрыть.

6. Запустите созданное приложение из среды разработки, для чего нажмите кнопку **Start debugging** (Запуск отладки – ) на стандартной панели инструментов (или в меню **Debug** (Отладка) выберите команду **Start debugging**). В центре экрана появится окно разработанного приложения.

Опробуйте действие различных элементов управления этого приложения (изменение размеров окна, управление состоянием окна, наличие и состав

управляющего меню в заголовке окна), а затем завершите его работу. Вы вернетесь в среду разработки.

7. Закройте данный проект, для чего в меню **File** (Файл) выберите команду **Close Project** (Закрыть проект). Проект при этом будет закрыт, а в главном окне среды разработки Visual Studio 2005 останется Стартовая страница (**Start Page**).

Закройте и само главное окно среды разработки Visual Studio.

8. Откройте окно просмотра Мой компьютер, Проводник или другой файловый менеджер. Найдите там исполняемый файл только что разработанного приложения (**FFirst.exe**) (этот файл находится в папке "...\\FFirst\\Debug") и двойным щелчком мыши по нему запустите его на выполнение. В центре экрана появится окно созданного Вами приложения. Еще раз опробуйте действие различных элементов управления этого приложения (см. п.6), а затем завершите его работу.

9. Откройте свой проект, для чего в открытом окне просмотра Мой компьютер (Проводник или др.) найдите файл **FFirst.vcproj** Вашего проекта (он находится в папке "...\\FFirst\\FFirst\\") и двойным щелчком мыши по нему откройте его в среде разработки Visual Studio (если в окне просмотра не отображаются типы файлов, то командой **Свойства папки...** в меню **Сервис** раскройте диалоговое окно **Свойства папки**, и на вкладке **Вид** в поле **Дополнительные параметры:** щелчком мыши снимите флажок **Скрывать расширения для зарегистрированных типов файлов**, а затем нажав на кнопку **ОК** закройте диалоговое окно). Загрузится среда разработки с выбранным проектом. В окне проводника проекта (**Solution Explorer**) среды разработки отобразится структура открытого проекта, а в главном окне – страница Конструктора форм – **FFirst.h [Design]** с формой **Моя первая программа**. Если страница Конструктора форм не появилась, то в окне проводника проекта (**Solution Explorer**) выделите файл формы **FFirst.h** и нажмите кнопку **View Designer** (Показать Конструктор) на панели инструментов окна проводника проекта (**Solution Explorer**) или выберите команду **Designer** (Конструктор) в меню **View** главного окна.

### Помещение элементов управления на форму

В Visual Studio командные кнопки, текстовые поля, списки и др. называются **элементами управления**. Для того чтобы поместить элемент управления на форму необходимо в панели элементов управления (**Toolbox**) щелчком мыши выбрать нужный элемент управления. Затем поместить указатель мыши в ту точку формы, где должен находиться левый верхний угол элемента управления, после чего щелчком мыши поместить его на форму (при этом размеры элемента управления устанавливаются по умолчанию). Если при размещении элемента управления на форме необходимо приблизительно определить его размеры, то вместо щелчка мышью нужно нажать ее левую кнопку мыши (зафиксировав левый верхний угол) и, удерживая ее нажатой, переместить указатель мыши в точку, где должен находиться правый нижний угол элемента управления, и там

отпустить кнопку мыши. В результате на форме появится выбранный элемент управления заданного размера.

Элемент управления формы, окруженный тонкой пунктирной рамкой и маркерами (маленькими белыми квадратиками), называется **выделенным**. Для того чтобы выделить какой-либо элемент управления, нужно щелкнуть по нему мышью.

Для удаления элемента управления необходимо его выделить и нажать клавишу <Delete>.

Для изменения положения элемента управления на поверхности формы, необходимо установить на него указатель мыши, нажать левую кнопку мыши и, удерживая ее нажатой, «перетащить» элемент управления в нужную точку формы, а затем отпустить кнопку мыши. Во время «перетаскивания» элемента управления Visual Studio отображает текущие значения координат левого верхнего угла элемента управления (значения свойства **Location: X** и **Y**) в правом углу строки состояния.

Для изменения размеров элемента управления, необходимо его выделить, установить указатель мыши на один из маркеров, помечающих границы компонента, нажать левую кнопку мыши и, удерживая ее нажатой, изменить положение границы элемента управления, а затем отпустить кнопку мыши. Во время изменения размера элемента управления Visual Studio отображает его текущие размеры: ширину и высоту (значения свойства **Size: Width** и **Height**) в правом углу строки состояния.

Свойства выделенного элемента управления так же, как и свойства формы, можно изменять в окне Свойств (**Properties**) среды разработки.

10. Поместите на форму метку, для этого щелчком мыши выберите элемент управления **Label** (Метка) в панели элементов управления. Затем поместите указатель мыши примерно в центр формы, и щелчком мыши поместите метку на форму. После этого в окне Свойств (**Properties**) среды разработки установите значения свойств метки, в соответствии с таблицей 1.2.

Таблица 1.2. Свойства метки и их значения

Свойство	Описание	Значение
(Name)	Имя метки	labelHello
Text	Текст метки	Здравствуй Мир!
TextAlign	Способ выравнивания текста	MiddleCenter
Location.X	Расстояние от левого края	60
Location.Y	Расстояние от верхнего края	45
Font	Шрифт	Шрифт – MS Sans Serif; Начертание – Жирный; Размер – 12 pt

11. В соответствии с таблицей 1.3 измените свойства формы **FFirst**, определяющие ее размер, с помощью окна Свойств (**Properties**) среды разработки.

Таблица 1.3. Свойства формы и их значения

Свойство	Определяет	Значение
<b>Size.Width</b>	Ширина формы	<b>300</b>
<b>Size.Height</b>	Высота формы	<b>150</b>

12. Сохраните измененный проект (п.5), для чего в меню **File** (Файл) выберите команду **Save All** (Сохранить все). Все файлы проекта сохранятся в своих папках, и никаких окон не раскроется, т.к. этот проект уже сохранялся ранее.

13. Закройте главное окно среды разработки Visual Studio вместе с проектом (см. п.7).

14. С помощью окна просмотра Мой компьютер (Проводник или др.) найдите и запустите на выполнение исполняемый файл разработанного проекта (см. п.8). Убедитесь, что в окне приложения отсутствует только что добавленная на форму метка, т.к. измененная программа не транслировалась и новый exe-файл не создавался. Затем завершите работу приложения.

15. Снова запустите на выполнение среду разработки программ Microsoft Visual Studio (см. п.1).

16. В среде разработки откройте свой проект, для чего в меню **File** выберите команду **Open Project...** (Открыть проект). После этого появится диалоговое окно **Open Project**, в котором в поле **Files of type:** (Типы файлов:) должно быть указано значение **All Project Files**. В списке папок и файлов этого диалогового окна найдите и откройте папку **FFirst** с вашим проектом (если открыта какая-либо другая), в ней выделите файл **FFirst.vcproj** и нажмите кнопку **Open** (Открыть).

Если в появившемся окне среды разработки не открыто окно Конструктора форм (**FFirst.h [Design]**), тогда в окне проводника проекта (**Solution Explorer**) с отображенной структурой открытого проекта выделите форму **FFirst.h** и нажмите кнопку **View Designer** (Показать Конструктор) на панели инструментов окна Проводника проекта (**Solution Explorer**) или выберите команду **Designer** (Конструктор) в меню **View** главного окна. В главном окне среды разработки Visual Studio появится страница Конструктора форм (**FFirst.h [Design]**) с формой **Моя первая программа**.

17. Откомпилируйте полученный проект. Для компиляции нужно запустить приложение на выполнение (см. п.6), дождаться появления на экране окна разработанного приложения, а затем завершить это приложение.

18. Сохраните свой проект (см. п.5) и закройте главное окно среды разработки Visual Studio вместе с проектом (см. п.7).

19. После этого запустите на выполнение исполняемый файл созданного приложения (см. п.8). Убедитесь, что теперь в окне приложения присутствует метка с надписью **"Здравствуй мир!"** (рис.1.1). Еще раз опробуйте действие различных элементов управления этого приложения (см. п.6). Убедитесь, что при разворачивании окна приложения или сильном увеличении его размеров надпись остается в левом верхнем углу окна. Очевидно, было бы лучше, если надпись оставалась бы в центре окна. Завершите работу приложения.

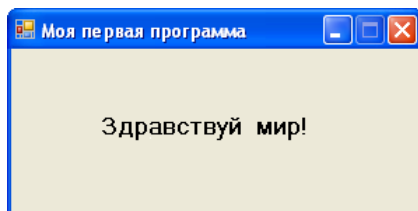


Рис.1.1.1. Окно приложения **Моя первая программа**

20. Снова запустите на выполнение среду разработки программ Microsoft Visual Studio (см. п.1), а в ней откройте свой проект (см. п.16).

21. На странице Конструктора форм (**FFirst.h [Design]**) с формой **Моя первая программа** установите для размещенной там метки свойство **Anchor** (управление прикреплением содержимого формы) в значение **None**.

22. Откомпилируйте измененный проект (см. п.17) и сохраните его (см. п.5), а затем закройте главное окно среды разработки Visual Studio вместе с проектом (см. п.7).

23. Запустите на выполнение исполняемый файл измененного приложения (см. п.8). Убедитесь, что теперь при разворачивании окна приложения или сильном увеличении его размеров надпись остается в центре окна приложения. Такой эффект был получен благодаря изменению значения свойства **Anchor** метки (см. п.21).

Результаты своей работы покажите преподавателю.

### Контрольные вопросы по теме: «Интегрированная среда разработки Microsoft Visual Studio»

1. Какие команды сгруппированы в меню **File** (Файл), **Edit** (Правка), **View** (Вид), **Project** (Проект), **Build** (Сборка), **Debug** (Отладка), **Data** (Данные), **Format** (Формат), **Tools** (Сервис), **Window** (Окно), **Community** (Сообщество) и **Help** (Справка) интегрированной среды разработки Visual Studio?

2. Что содержит Главное меню, Стандартная панель инструментов, Панель элементов управления среды разработки Visual Studio?

3. Для чего предназначены окна Обозревателя решений (**Solution Explorer**), Свойств (**Properties**), Конструктора форм, Редактора кода, Просмотра объектов (**Object Browser**), Вывода (**Output**)?

4. Что происходит в редакторе кода при нажатии комбинаций клавиш **<Ctrl+X>**, **<Ctrl+C>**, **<Ctrl+V>**?

5. Какие команды могут использоваться для компиляции проекта? Чем они отличаются?

6. В чем заключается отладка приложения? Какие операции отладки возможны в редакторе кода, и какие команды для этого используются?



## Задание на занятие №2.

### Разработка линейной программы «Простой калькулятор»

1. Загрузите среду разработки Visual Studio и создайте в ней новый проект **Calc**. В главном окне среды разработки появится страница Конструктора форм с пустой формой нового проекта. Установите свойства формы в соответствии с таблицей 2.1, а затем переименуйте разработанную форму на **FFirst.h**.

Таблица 2.1. Свойства формы и их значения

Свойство	Описание	Значение
<b>Text</b>	Заголовок формы	<b>Calc</b>
<b>FormBorderStyle</b>	Стиль обрамления формы	<b>FixedSingle</b> (неизменяемая рамка, не допускающая изменения размеров окна)
<b>MaximizeBox</b>	Доступность кнопки «Развернуть» в заголовке окна	<b>False</b> (кнопка «Развернуть» не доступна)
<b>Size.Width</b>	Ширина формы	<b>360</b>
<b>Size.Height</b>	Высота формы	<b>260</b>

2. Разместите в правом нижнем углу формы командную кнопку (элемент управления **Button**) и установите для нее значения свойств, в соответствии с таблицей 2.2.

Таблица 2.2. Свойства кнопки и их значения


Свойство	Описание	Значение
<b>(Name)</b>	Имя кнопки	<b>btnClose</b>
<b>Text</b>	Заголовок кнопки	<b>Заккрыть</b>
<b>Location.X</b>	Расстояние от левого края	<b>240</b>
<b>Location.Y</b>	Расстояние от верхнего края	<b>185</b>
<b>Size.Width</b>	Ширина кнопки	<b>100</b>
<b>Size.Height</b>	Высота кнопки	<b>30</b>

3. Создайте для этой кнопки обработчик события **Click** (щелчок мышью по кнопке), для чего произведите двойной щелчок по ней мышью. В результате откроется окно редактора кода с созданным там шаблоном метода обработки события **Click** (этот шаблон создается автоматически):

```
...
private: System::Void btnClose_Click(System::Object^ sender,
                                     System::EventArgs^ e) {
}
...
```

4. Установите текстовый курсор внутри составного оператора тела метода (между фигурными скобками) и введите имя метода: **this->Close();** . Метод обработки события должен приобрести следующий вид:

```
private: System::Void btnClose_Click(System::Object^ sender,
                                     System::EventArgs^ e) {
    this->Close();
}
```

5. Запустите приложение на выполнение. Для этого нажмите кнопку **Start Debugging** (Запуск отладки – ) на стандартной панели инструментов.

6. Убедитесь, что приложение правильно реагирует на событие – нажатие кнопки **Заккрыть**.

7. Вернитесь в окно конструктора форм и разместите на форме еще две командные кнопки такого же размера и чуть выше первой. Установите для них значения свойств, в соответствии с таблицей 2.3.

Таблица 2.3. Свойства кнопки и их значения

Свойство	Описание	Значение
(Name)	Имя кнопки	btnAdd, btnReset
Text	Текст кнопки	Сложить, Сброс
Location.X	Расстояние от левого края	240
Location.Y	Расстояние от верхнего края	10, 150
Size.Width	Ширина кнопки	100
Size.Height	Высота кнопки	30

8. Добавьте сверху по центру формы метку (элемент **Label**), в свойстве **Text** которой ничего не записано. Установите для нее значения свойств, в соответствии с таблицей 2.4.


Таблица 2.4. Свойства метки и их значения

Свойство	Описание	Значение
(Name)	Имя метки	lblOper
Text	Текст метки	
TextAlign	Способ выравнивания текста	MiddleCenter
Location.X	Расстояние от левого края	110
Location.Y	Расстояние от верхнего края	10
Font	Шрифт	Шрифт – MS Sans Serif; Начертание – Жирный; Размер – 10 пт

9. Создайте для кнопок **Сложить** и **Сброс** обработчики событий **Click** (щелчок по кнопке) аналогично п.3. В методах обработки событий введите коды, изменяющие значение свойства **Text** метки **lblOper** следующим образом:

```
private: System::Void btnAdd_Click(System::Object^ sender,
                                   System::EventArgs^ e) {
    this->lblOper->Text = L"Сложение";
}

private: System::Void btnReset_Click(System::Object^ sender,
                                       System::EventArgs^ e) {
    this->lblOper->Text = L"";
}
}
```

10. Запустите приложение на выполнение. Для этого нажмите кнопку **Start Debugging** (Запуск отладки – ) на стандартной панели инструментов.

11. Убедитесь, что после щелчка по кнопке **Сложить** вверху по центру формы появляется надпись «Сложение», а после щелчка по кнопке **Сброс** эта надпись исчезает. После этого завершите работу приложения нажатием кнопки **Заккрыть**.

12. Вернитесь в окно конструктора форм и разместите на форме еще три метки (элементы **Label**) и три текстовых поля (элементы **TextBox**) так, как показано на рис.2.1. Установите для них значения свойств, в соответствии с таблицей 2.5.

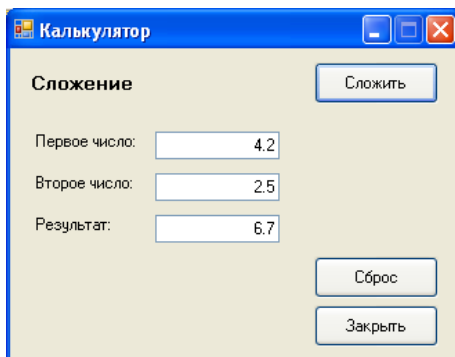


Рис.2.1. Окно приложения **Калькулятор** (предварительное)


13. Измените для кнопок **Сложить** и **Сброс** обработчики событий **Click** (щелчок по кнопке). В методы обработки событий добавьте следующие коды:

```
private: System::Void btnAdd_Click(System::Object^ sender,
                                   System::EventArgs^ e) {
    int i1, i2, i3;
    this->lblOper->Text = L"Сложение";
    i1 = Convert::ToInt32(txt1->Text);
    i2 = Convert::ToInt32(txt2->Text);
    i3 = i1 + i2;
    this->txtResult->Text = Convert::ToString(i3);
}
}
```

```
private: System::Void btnReset_Click(System::Object^ sender,
                                     System::EventArgs^ e) {
    this->lblOper->Text = L"";
    this->txt1->Text = L"";
    this->txt2->Text = L"";
    this->txtResult->Text = L"";
}
```

Таблица 2.5. Свойства элементов и их значения

Свойство	Описание	Значения	
		Элементы Label	Элементы TextBox
(Name)	Имя элемента	lbl1, lbl2, lblResult	txt1, txt2, txtResult
Text	Текст элемента	Первое число: Второе число: Сумма:	(пустые поля)
TextAlign	Способ выравнивания текста	MiddleLeft	Right
Location.X	Расстояние от левого края	10	110
Location.Y	Расстояние от верхнего края	60, 90, 120	60, 90, 120
Size.Width	Ширина	–	90
Size.Height	Высота	–	20
Font	Шрифт	Шрифт – MS Sans Serif; Начертание – Обычный; Размер – 8 pt	Шрифт – MS Sans Serif; Начертание – Обычный; Размер – 8 pt

14. Запустите приложение на выполнение. Для этого нажмите кнопку **Start Debugging** (Запуск отладки – ) на стандартной панели инструментов.

15. Убедитесь, что после ввода исходных данных (двух целых чисел) и нажатия кнопки **Сложить** происходит вычисление их суммы, а после нажатия кнопки **Сброс** происходит очистка текстовых полей и метки названия операции. После этого завершите работу приложения нажатием кнопки **Закрыть**.

16. Вернитесь в окно конструктора форм и разместите на форме еще три кнопки: **Вычесть**, **Умножить** и **Разделить** (рис.2.2). Установите для них соответствующие значения свойств (см. таблицу 2.3).

17. Самостоятельно разработайте для этих кнопок обработчики событий **Click**.

18. Так как результатом различных операций могут быть и сумма, и разность, и произведение, и частное, то название третьего поля нужно заменить на универсальное. Для этого измените значение свойства **Text** для метки **lblResult** на **Результат:** (рис.2.2).

19. Запустите приложение на выполнение (см. п.5) и проверьте правильность его работы, после чего завершите его работу.

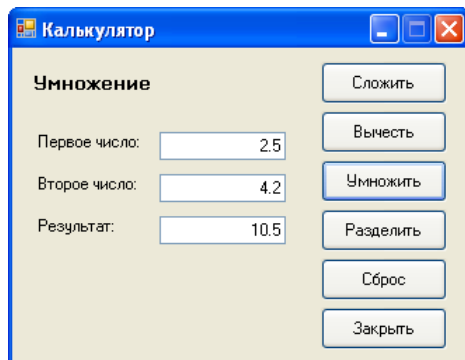


Рис.2.2. Окно приложения **Калькулятор** (итоговое)

20. Сохраните проект командой **Save All (Сохранить все)** меню **File** (Файл) в папке "...\\Lab2\\Calc\\".

21. Закройте окно среды разработки Visual Studio. После этого запустите на выполнение исполняемый файл созданного приложения (он находится в папке "...\\Lab2\\Calc\\bin\\Debug\\"), еще раз проверьте правильность его работы, а затем завершите его работу.

22. Еще раз откройте свой проект и самостоятельно доработайте созданный калькулятор следующим образом:

- он должен правильно реагировать на попытку деления на 0, выдавая соответствующее сообщение;
- он должен правильно работать с действительными (вещественными) числами (рис.2.2);
- он должен корректно обрабатывать возможные ошибки в работе программы или при вводе исходных данных.

Сообщения об ошибках или необходимых действиях рекомендуется выводить в дополнительную метку (элемент **Label**) под результатом операции.

После доработки не забудьте сохранить свой проект, а затем закройте окно среды разработки.

Подготовьте необходимые наборы исходных данных для тестирования созданного приложения. Результаты своей работы покажите преподавателю.

### **Домашнее задание.**

Самостоятельно разработайте свой (возможно упрощенный) аналог программы **Калькулятор** (от компании Microsoft), например, такой как показан на рис.2.3 с однострочным полем для отображения данных.

Убедитесь, что разработанный калькулятор правильно выполняет все допустимые операции с различными исходными данными, включая «цепочки» операций. С правилами выполнения отдельных операций можно ознакомиться, выполняя аналогичные операции во встроенном калькуляторе Windows или в обычном «железном» калькуляторе.



Рис.2.3. Вариант программы **Калькулятор**

### Контрольные вопросы по теме: «Основные элементы программирования»

1. Какие типы данных описываются с помощью ключевых слов **bool**, **char**, **unsigned char**, **signed char**, **short**, **unsigned short**, **int**, **unsigned int**, **\_\_intn**, **long**, **unsigned long**, **long long**, **unsigned long long**, **float**, **double**, **long double**, **String**?
2. Какой объем в памяти ЭВМ занимают переменные типов **bool**, **char**, **unsigned char**, **signed char**, **short**, **unsigned short**, **int**, **unsigned int**, **\_\_intn**, **long**, **unsigned long**, **long long**, **unsigned long long**, **float**, **double**, **long double**, **String**?
3. Какие значения могут принимать данные типов **bool**, **char**, **unsigned char**, **signed char**, **short**, **unsigned short**, **int**, **unsigned int**, **\_\_intn**, **long**, **unsigned long**, **long long**, **unsigned long long**, **float**, **double**, **long double**, **String**?
4. Что определяет тип данных?
5. Что называется константой, переменной? Как они объявляются?
6. Каковы области действия глобальных и локальных переменных?
7. Какой объект может использоваться в качестве стартового при запуске приложения на выполнение?

### Задание на занятие №3.

## Разработка программы с разветвлениями

### «Решение квадратного уравнения»

Реализуйте на Visual C++ приложение для решения квадратного уравнения, для чего:

1. Создайте форму, имеющую например вид, представленный на рис.3.1.

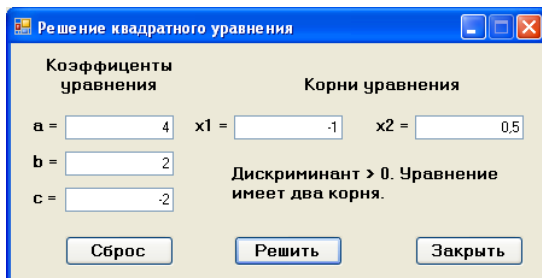


Рис.3.1. Окно приложения **Решение квадратного уравнения**

2. Коэффициенты квадратного уравнения **a**, **b**, и **c** удобно вводить посредством текстовых полей (**TextBox**). Не забудьте, что они могут не только целыми, но и вещественными.

3. Корни уравнения **x1** и **x2** можно выводить в соответствующие текстовые поля (**TextBox**).

4. Комментарии к результатам решения уравнения целесообразно выводить в компонент метки (**Label**).

5. Напишите обработчик события нажатия кнопки **Решить** в соответствии с алгоритмом решения квадратного уравнения.

6. Обработчик события нажатия кнопки **Закреть** должен обеспечивать корректное завершение работы приложения.

7. Доработайте приложение таким образом, чтобы учитывать возможность того, что коэффициенты квадратного уравнения **a**, **b**, **c** (в любом сочетании) могут быть нулевыми. При этом необходимо выводить и соответствующие комментарии.

Для любых значений коэффициентов (в том числе и нулевых) выводимые комментарии должны быть математически точными.

8. Сохраните свой проект.

9. Запустите приложение на выполнение и убедитесь, что оно правильно реагирует на события, связанные с нажатием кнопок **Решить** и **Закреть**.

При тестировании работы приложения можно ввести, например, следующие значения коэффициентов:

– в случае **a = 4**, **b = 2**, **c = -2** ответ должен быть следующим: **x1 = -1**, **x2 = 0.5** и комментарий – «Дискриминант > 0. Уравнение имеет два корня.»;

– в случае  $a = 4$ ,  $b = 4$ ,  $c = 1$  ответ должен быть следующим:  $x_1 = -0.5$ ,  $x_2 = -0.5$  и комментарий – «Дискриминант = 0. Уравнение имеет два одинаковых корня.»;

– в случае  $a = 4$ ,  $b = 4$ ,  $c = 4$  ответ должен содержать только комментарий – «Дискриминант < 0. Уравнение не имеет действительных корней.»;

– в случае  $a = 0$ ,  $b = 2$ ,  $c = -2$  ответ должен быть следующим:  $x_1 = 1$  и комментарий – «Линейное уравнение – имеет один корень.».

Продумайте комментарии для случаев, когда коэффициенты  $a$  и  $b$  нулевые, а  $c$  – любой. В любом случае для любых значений коэффициентов выводимые комментарии должны быть математически корректными.

В созданном приложении должны корректно обрабатываться возможные ошибки в работе программы или при вводе исходных данных.

Подготовьте необходимые наборы исходных данных для тестирования созданного приложения. Результаты своей работы покажите преподавателю.

## Контрольные вопросы по теме: «Строки»

1. Что представляет собой переменная типа **System::String**? Краткая характеристика типа. Объявление, инициализация и использование строк типа **System::String**.

2. Что представляет собой переменная типа **System::Text::StringBuilder**? Чем отличаются строки типов **System::String** и **System::Text::StringBuilder**? Объявление, инициализация и использование строк типа **System::Text::StringBuilder**.

3. Назначение и краткая характеристика структуры **System::Char**.

4. Основные поля, свойства и методы классов **System::String** и **System::Text::StringBuilder**.

5. Основные поля и методы структуры **System::Char**.

6. Что представляет собой строка в стиле C? Краткая характеристика этих строк. Объявление, инициализация и использование строк в стиле C.

7. Функции стандартной библиотеки для работы со строками и символами.

8. Характеристика строк C++, представляемых библиотечным классом **string**. Объявление, инициализация и использование строк C++.

9. Функции библиотечного класса **string** для работы со строками C++.

10. Как преобразуются строки различных типов в C++?

## Контрольные вопросы по теме: «Регулярные выражения»

1. Что представляют собой регулярные выражения? Поясните смысл использования регулярных выражений для обработки текста? Какие задачи обработки текста решаются с помощью регулярных выражений?

2. Основные методы класса **System::Text::RegularExpressions::Regex**.



3. Назовите и охарактеризуйте основные элементы языка регулярных выражений. Приведите примеры.

### Контрольные вопросы по теме: «Ввод-вывод»

1. Основные свойства и методы класса **System::Console**, предоставляемые для организации консольного ввода-вывода.

2. Основные возможности, предоставляемые перечислениями **System::ConsoleKey** и **System::ConsoleModifiers** и структурой **System::ConsoleKeyInfo** для организации ввода-вывода.

3. В чем смысл форматирования строк, в каких случаях оно применяется? Синтаксис элемента форматирования, его составляющие.

4. Строки стандартных числовых форматов: структура строк формата, спецификаторы стандартных числовых форматов, примеры использования.

5. Строки настраиваемых числовых форматов: описатели настраиваемых числовых форматов и примеры их использования.

6. Форматирование с помощью управляющих последовательностей: управляющие последовательности, их интерпретация и применение.

7. Ввод-вывод с использованием элементов управления графического интерфейса. Использование стандартного окна **MessageBox** для вывода сообщений.

8. Обработка событий нажатия клавиш для элементов управления: используемые события, свойства классов **KeyPressEventArgs** и **KeyEventArgs**, перечисление **Keys**.

9. Использование свойств классов **System::Globalization::NumberFormatInfo** и **System::Globalization::CultureInfo** для получения сведений о языке и региональных параметрах при форматировании и анализе числовых значений.

10. Защита от дурака и способы ее реализации. Примеры использования.

11. Ввод-вывод данных в языке C++: объекты для ввода и вывода, их использование.

12. Ввод-вывод данных в языке C: функции для ввода и вывода, спецификаторы формата и их параметры.

### Контрольные вопросы по теме: «Операторы управления и циклы»

1. Что называется выражением?

2. Для чего используются круглые скобки в выражениях?

3. Перечислите арифметические операторы и выполняемые ими функции в порядке убывания приоритетов.

4. Какие операторы используются в операциях отношения?

5. Какие операторы используются в логических выражениях?

6. Какие операторы могут использоваться в поразрядных операциях над числовыми выражениями?

7. В каких выражениях (операциях) используются операторы: ==, !=, <, >, <=, >= ?
8. В каких выражениях (операциях) используются операторы: !, &&, || ?
9. В каких выражениях (операциях) используются операторы: &, |, ^, <<, >> ?
10. Результатом каких выражений (операций) могут служить значения **true** и **false**?
11. Какие действия выполняет оператор присваивания?
12. Какой(ие) символ(ы) используется(ются) для записи полного оператора присваивания?
13. Допускаются ли сокращенные операторы присваивания?
14. Что представляет собой блок операторов?
15. Каким(и) символом(и) в тексте программы выделяется однострочный комментарий?
16. Каким(и) символом(и) в тексте программы выделяется многострочный комментарий?
17. Как разбить один длинный оператор на несколько строк, т.е. разместить его на нескольких строках?
18. Как разместить два (или более) коротких оператора в одной строке?

## Задание на занятие №4.

### Разработка программы с разветвлениями «АРМ оператора обменного пункта»

Реализуйте на Visual C++ приложение, имитирующее АРМ оператора обменного пункта, для чего:

1. Создайте форму, имеющую например вид, представленный на рис.4.1.
2. Результаты работы приложения должны выглядеть приблизительно так, как показано на рис.4.1.

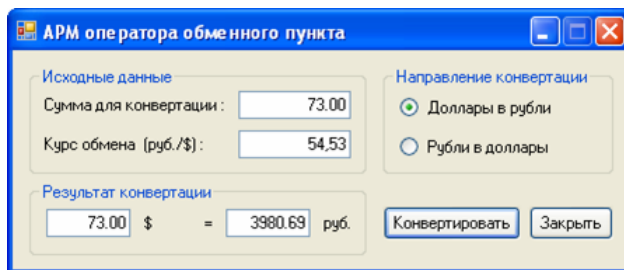


Рис.4.1. Окно приложения **АРМ оператора обменного пункта**  
(предварительное)

3. Реализуйте обработчик события нажатия кнопки **Конвертировать** в соответствии с алгоритмом конвертации валют.

4. При изменении направления конвертации должны «очищаться» поля результатов работы, отображаться соответствующий курс обмена, а единицы измерения должны меняться.

*Следует заметить, что при обмене валюты в реальности происходит не обмен, а ее продажа или покупка, выполняемые в рублях.*

5. Введенный курс обмена в каждом направлении должен запоминаться и появляться при очередном выборе данного направления обмена.

6. Если не введены исходные данные (сумма для конвертации или курс обмена) или введены недопустимые значения, то они должны корректироваться или выдаваться соответствующие сообщения.

7. Полученный результат конвертирования валюты необходимо округлять строго до сотых (т.е. до копейки или цента).

*Имейте в виду, что в банковской сфере все округления производятся в пользу банка, а не клиента.*

8. Нажатие кнопки **Закрыть** должно осуществлять корректное завершение работы приложения.

9. Доработайте приложение для получения возможности конвертировать рубли в евро и наоборот (рис.4.2). Можно добавить и другие валюты.

10. Сохраните свой проект.

В созданном приложении должны корректно обрабатываться возможные ошибки в работе программы или при вводе исходных данных.

Подготовьте необходимые наборы исходных данных для тестирования созданного приложения. Результаты своей работы покажите преподавателю.

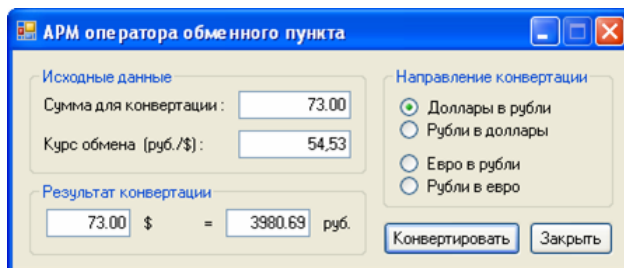


Рис.4.2. Окно приложения АРМ оператора обменного пункта (итоговое)

## Контрольные вопросы по теме: «Операторы управления и циклы»

1. Какие операторы относятся к операторам ветвления?

2. Какое действие в программе выполняет следующий оператор:

```
if (<условие>) <оператор>;
```

3. Чем отличаются две формы записи условного оператора **if**: полная и сокращенная?

4. В каком случае в условном операторе **if-else** выполняется группа операторов, указанная между <условием> и ключевым словом **else**?

5. В каком случае в условном операторе **if-else** выполняется группа операторов, указанная после ключевого слова **else**?

6. Какое действие в программе выполняет следующий оператор

```
switch (<выражение>)  
{ case <константное_выражение_1>:  
    <операторы_1>;  
    <оператор_перехода_1>;  
  case <константное_выражение_2>:  
    <операторы_2>;  
    <оператор_перехода_2>;  
  ...  
  default:  
    <операторы_N>;  
    <оператор_перехода_N>;  
}
```

7. Чем отличаются две формы записи оператора выбора варианта (**switch-case**): полная и сокращенная?

8. В каком случае в условном операторе (**switch-case**) выполняется группа операторов, указанная в ветви, начинающейся с ключевого слова **case**?

9. В каком случае в условном операторе (**switch-case**) выполняется группа операторов, указанная в ветви, начинающейся с ключевого слова **default**?

10. В какой последовательности в операторе выбора варианта (**switch-case**) должны размещаться ветви "**case**" и ветвь "**default**"?

11. Что произойдет, если в ветвях оператора выбора варианта (**switch-case**) будут отсутствовать <операторы\_перехода>?

## Задание на занятие №5.

### Разработка простой циклической программы «Расчет значения $\exp(x)$ »

#### Постановка задачи:

Вычислить значение функции  $\exp(x)$  с точностью до  $\text{eps} = 0.1, 0.01, \dots, 0.000001$  путем разложения функции в ряд.

#### Математическое описание задачи:

Для разложения функции экспоненты в ряд может использоваться следующее выражение:

$$\exp(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots = \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

Последнее слагаемое суммы ряда не должно превышать значения  $\text{eps}$ .

Реализуйте на Visual C++ приложение для расчета значения  $\exp(x)$  с помощью цикла, для чего:

1. Используя необходимые элементы управления создайте форму, аналогичную представленной на рис.5.1.

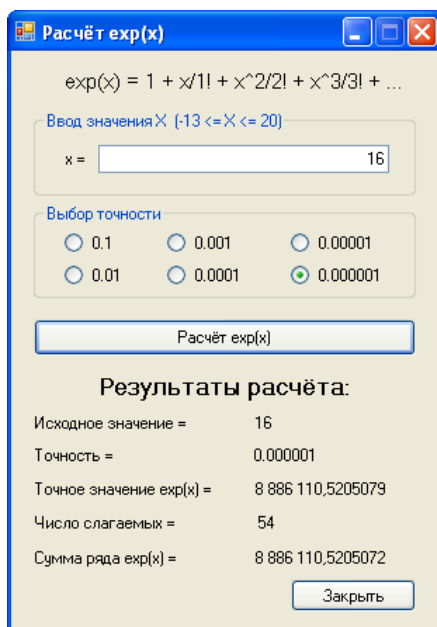


Рис.5.1. Окно приложения **Расчет  $\exp(x)$**

2. Результаты работы приложения должны выглядеть примерно так, как показано на рис.5.1.

3. Ввод значения  $x$  реализуйте с помощью элемента управления **TextBox** со значением по умолчанию  $x = 1$ .

4. Выбор значения точности **eps** удобно осуществлять с помощью элемента управления **RadioButton**.

5. Реализуйте обработчик события нажатия кнопки **Расчет  $\exp(x)$** , в котором:

- разработайте метод для расчета суммы ряда в соответствии с математическим описанием задачи. Для вычисления слагаемых суммы выведите рекуррентную формулу. Сумму ряда вычислите с помощью цикла, подсчитав при этом и количество суммируемых слагаемых;

- «точное» значение  **$\exp(x)$**  вычислите с помощью метода **Exp(x)** класса **System.Math**;

- для удобства сравнения выводимые на форму результаты представляйте в формате с фиксированной запятой и строго друг под другом;

- для корректного сравнения любых полученных результатов округляйте их в зависимости от выбранной точности (+ один знак, чтобы избежать ошибок округления).

6. Аналитическим или опытным путем определите максимально возможный диапазон изменения значений  $x$  и укажите его на форме. Обоснуйте свое решение.

7. При изменении исходных данных должны «очищаться» результаты расчета.

8. В создаваемом приложении реализуйте корректную обработку (и устранение) возможных ошибок в работе программы или при вводе исходных данных.

9. Нажатие кнопки **Заккрыть** должно осуществлять корректное завершение работы приложения.

10. Сохраните свой проект.

Подготовьте необходимые наборы исходных данных для тестирования созданного приложения. Результаты своей работы покажите преподавателю.

### Домашнее задание.

Выполните данное задание, используя другие разновидности циклов. Сравните результаты различных вариантов решения.

### Другие варианты заданий.

По решению преподавателя исследуемая на занятии функция может быть заменена (рис.5.2).

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!} ;$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}.$$

**Примечание.** Графики функций, исследуемых на занятии, имеют вид:

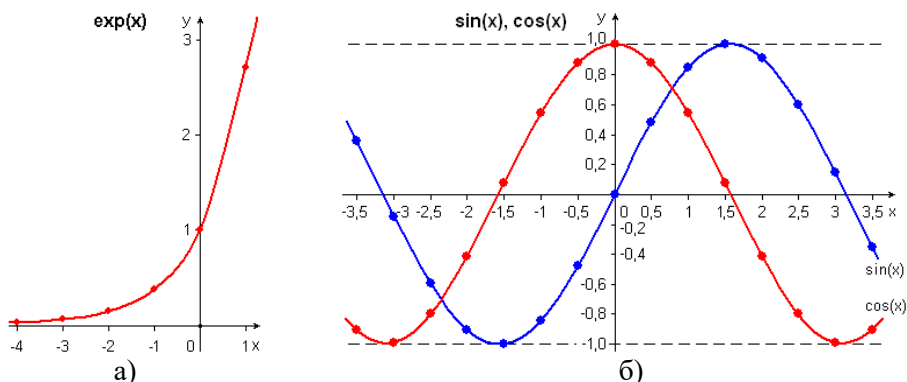


Рис.5.2. Графики функций: а)  $y = e^x$ ; б)  $y = \sin(x)$  и  $y = \cos(x)$

## Контрольные вопросы по теме: «Операторы управления и циклы»

1. Какой оператор цикла представляет управляющая конструкция **while**?
2. Какой оператор цикла представляет управляющая конструкция **do-while**?
3. Какой оператор цикла представляет управляющая конструкция **for**?
4. Какая из приведенных управляющих конструкций реализует «Цикл с предусловием», «Цикл с постусловием», «Цикл с параметром»?

1) **for** (<инициализация\_счетчиков>; <условие>;

<изменение\_счетчиков>)

<операторы\_тела\_цикла>;

2) **while** (<условие>)

<операторы\_тела\_цикла>;

3) **do**

<операторы\_тела\_цикла>;

**while** (<условие>);

5. В каком случае в управляющей конструкции **for** выполняются <операторы тела цикла>?

6. В каком случае в управляющей конструкции **while** выполняются <операторы тела цикла>?

7. В каком случае в управляющей конструкции **do-while** выполняются <операторы тела цикла>?



8. В каком случае в управляющей конструкции **for** <операторы тела цикла> не выполняются ни разу?

9. В каком случае в управляющей конструкции **while** <операторы тела цикла> не выполняются ни разу?

10. В каком случае в управляющей конструкции **do-while** <операторы тела цикла> не выполняются ни разу?

11. В каком случае в управляющей конструкции **do-while** <операторы тела цикла> выполняются только один раз?

12. В какой(их) из приведенных управляющих конструкций при любых обстоятельствах <операторы тела цикла> выполняются как минимум один раз?

1) **for** (<инициализация\_счетчиков>; <условие>;

<изменение\_счетчиков>)

<операторы\_тела\_цикла>;

2) **while** (<условие>)

<операторы\_тела\_цикла>;

3) **do**

<операторы\_тела\_цикла>;

**while** (<условие>);

13. В какой(их) из приведенных управляющих конструкций <операторы тела цикла> могут не выполниться ни разу?

1) **for** (<инициализация\_счетчиков>; <условие>;

<изменение\_счетчиков>)

<операторы\_тела\_цикла>;

2) **while** (<условие>)

<операторы\_тела\_цикла>;

3) **do**

<операторы\_тела\_цикла>;

**while** (<условие>);

14. Какой смысл имеет <условие> в следующем операторе?

**for** (<инициализация\_счетчиков>; <условие>;

<изменение\_счетчиков>)

<операторы\_тела\_цикла>;

15. Какой смысл имеет <условие> в следующем операторе?

**while** (<условие>)

<операторы\_тела\_цикла>;

16. Какой смысл имеет <условие> в следующем операторе?

**do**

<операторы\_тела\_цикла>;

**while** (<условие>);

17. Какое действие в теле цикла выполняет оператор **break**?

18. Какое действие в теле цикла выполняет оператор **continue**?

19. Какое действие в коде программы выполняет оператор **goto**?

20. Какое действие в подпрограмме выполняет оператор **return**?

## Задание на занятие №6.

### Разработка программы обработки массива

#### Постановка задачи:

Разработать приложение, позволяющее выполнять предусмотренные операции обработки введенного одномерного массива:

- определение суммы элементов массива;
- определение среднего значения элементов массива;
- определение номера и значения минимального элемента массива;
- определение номера и значения максимального элемента массива;
- вывод четных значений элементов массива;
- вывод нечетных значений элементов массива;
- сортировка элементов массива по возрастанию;
- сортировка элементов массива по убыванию.

В приложении предусмотреть возможность ввода одномерного массива тремя способами:

- генерацией массива с заданным количеством случайных целых чисел, равномерно распределенных в заданном диапазоне;
- вводом из существующего текстового файла (с возможностью задавать имя этого файла) с автоматическим определением количества вводимых элементов;
- вводом исходного массива с клавиатуры в текстовое поле или добавлением/удалением элементов уже имеющегося массива.

Реализовать возможность сохранения исходного массива и результатов его обработки в текстовом файле. При этом при последовательном сохранении результатов нескольких операций в один файл (методом добавления) исходный массив должен выводиться только при сохранении результата первой операции над заданным массивом.

Все ошибочные ситуации должны правильно обрабатываться, о чем при необходимости должны выдаваться соответствующие сообщения.

Реализуйте на Visual C++ приложение для выполнения указанных операций обработки массива, для чего:

1. Используя необходимые элементы управления, создайте форму, аналогичную представленной на рис.6.1.

2. Результаты работы приложения при выполнении одной из операций должны выглядеть приблизительно так, как показано на рис.6.1.

3. Реализуйте обработчик события нажатия кнопки **Генерация массива** , используя для этого методы генерации случайных чисел класса **System.Random** .

4. Реализуйте обработчик события нажатия кнопки **Ввод из файла** с возможностью задавать имя файла ввода и использования стандартного диалогового окна открытия файлов. При этом размер вводимого из файла массива должен определяться программно, а не задаваться пользователем в файле. Здесь же следует реализовать и использовать функцию удаления

возможно присутствующих в файле лишних или недопустимых символов (пробелов, символов табуляции и перевода строк, букв и др.).

Обратите внимание на то, что после ввода данных из файла или методом случайной генерации, они могут быть дополнены или изменены вручную.

Обработка массива

Исходные данные

Количество элементов массива: 10

Минимальное значение диапазона: -12

Максимальное значение диапазона: 20

Генерация массива

Имя файла ввода: d:\CS\_C31\Lab6\In.txt

Ввод из файла

Исходный массив:  
-6 10 3 -11 0 11 18 5 -9 12

Операции с массивом

☐ Сумма элементов ☐ Чётные элементы

☐ Среднее значение ☐ Нечётные элементы

☐ Минимальный элемент ☒ Сортировка по возрастанию

☐ Максимальный элемент ☐ Сортировка по убыванию

Выполнить

Имя файла вывода: d:\CS\_C31\Lab6\Out.txt

Сохранить в файл

Результат операции:  
Сортировка по возрастанию: -11 -9 -6 0 3 5 10 11 12 18

Заккрыть

Рис.6.1. Окно приложения **Обработка массива**

5. Если при генерации массива исходные данные для этого (размер массива и диапазон его значений) не введены, то должны выдаваться соответствующие сообщения.

6. Реализуйте обработчик события нажатия кнопки **Выполнить** в зависимости от выбранной переключателем операции. Каждую операцию обработки массива оформите в виде метода (функции) с передачей ему исходного массива в качестве входного параметра и получением результата в качестве выходного параметра.

7. Если при выполнении операции обработки массива он еще не задан, то должно выдаваться соответствующее сообщение.

8. Реализуйте обработчик события нажатия кнопки **Сохранить в файл** для сохранения исходного массива и результатов его обработки в файл, задаваемый пользователем. Для возможности накопления результатов в файле рекомендуется использовать сохранение с добавлением. При этом не следует сохранять повторяющиеся данные (как исходные, так и результаты).

9. Реализуйте обработчик возможных ошибочных ситуаций при работе с файлами.

10. Реализуйте обработчик события нажатия кнопки **Заккрыть**, которое должно осуществлять корректное завершение работы приложения.

11. Сохраните свой проект.

В созданном приложении должны корректно обрабатываться возможные ошибки в работе программы или при вводе исходных данных.

Подготовьте необходимые наборы исходных данных для тестирования созданного приложения. Результаты своей работы покажите преподавателю.

### **Домашнее задание.**

Самостоятельно доработайте свою программу, используя указатели для выполнения операций обработки массива. Сравните варианты вашей программы и сделайте выводы.

## **Контрольные вопросы по теме: «Массивы»**

1. Что называется массивом? Как он объявляется? Какой параметр указывается в скобках при объявлении и использовании массива?
2. Что называется размером массива? Чем ограничен максимальный размер массива?
3. Что называется индексом? Какие данные могут быть использованы в качестве индексов при обозначении элементов массива?
4. Сколько размерностей максимально может иметь многомерный массив?
5. Какие массивы называются статическими, динамическими?
6. Каковы особенности объявления и использования статических и динамических массивов? Как можно изменить размер динамического массива?
7. Может ли быть динамический массив многомерным и почему?
8. Как массивы могут передаваться в методы?

## **Контрольные вопросы по теме: «Подпрограммы как методы»**

1. Что называется методом-процедурой и методом-функцией?
2. Что задается при определении метода?
3. Как организуется информационная связь между программой и методом?
4. Какие соответствия должны соблюдаться между формальными и фактическими параметрами метода?
6. Как выглядит оператор вызова метода-процедуры?
7. Какие разновидности (статусы) формальных параметров возможны при определении метода?
8. Каким образом могут передаваться параметры в метод? Какой способ передачи параметров в метод используется по умолчанию?
9. Чем могут быть выражены фактические параметры, передаваемые в методы по значению и по ссылке?

10. Какую роль (входной и/или выходной параметр) могут выполнять параметры, передаваемые в методы по значению и по ссылке?
11. Как могут передаваться в методы массивы?
12. Как можно задать произвольное количество параметров метода при его определении?

### **Контрольные вопросы по теме: «Указатели»**

1. Что называется указателем? Как он объявляется, инициализируется и используется? Для чего используются оператор получения адреса и оператор косвенного доступа?
2. Особенности указателей на тип **char**. Особенности массивов указателей. Их объявление, инициализация и использование.
3. В каких выражениях могут использоваться указатели? Операторы, используемые с указателями, примеры использования. Использование указателей для работы с массивами.
4. Что называется ссылкой? Как она объявляется, инициализируется и используется. Назначение и особенности использования оператора **sizeof**.
5. Особенности динамического выделения памяти. Какие операторы используются для выделения и возврата памяти? Форматы и примеры их использования.

### **Контрольные вопросы по теме: «Структуры»**

1. Что называется структурой? Как объявляется структура и переменная типа структуры? Как можно использовать структуры?
2. Как объявляются и используются битовые поля в структурах?

### **Контрольные вопросы по теме: «Перечисления»**

1. Что называется перечислением? Как оно объявляется, инициализируется и используется?
2. Какой тип могут иметь перечисления? На что он указывает?
3. Какие операции могут применяться к значениям перечислений?
4. Как создается перечисление с битовыми флагами?

### **Контрольные вопросы по теме: «Организация работы с файлами»**

1. На какие типы делятся файлы по организации хранения информации в них и способам доступа к этой информации?

2. Какие файлы являются файлами последовательного доступа, произвольного доступа, двоичными (бинарными)?
3. Какие типовые операции используются при работе с файлами?
4. Для чего предназначены управляющие элементы: **OpenFileDialog**, **SaveFileDialog**, **FolderBrowserDialog**? Каковы особенности их использования?
5. Методы каких классов (для работы с файловой системой) из пространства имен **System.IO** (**File**, **FileInfo**, **Directory**, **DirectoryInfo**, **Path**, **DriveInfo**, **FileStream**, **StreamReader**, **StreamWriter**) являются статическими? В чем особенность статических методов классов?

### **Контрольные вопросы по теме: «Разработка пользовательского интерфейса»**

1. Что называется интерфейсом? Принципы разработки интерфейса, их характеристика.
2. Типы интерфейсов, их характеристика и типовой состав элементов.
3. Характеристика основных элементов пользовательского интерфейса, особенности их реализации.
4. Диалоговые окна, особенности их реализации. Модальность диалоговых окон.
5. Назначение и характеристика стандартных элементов управления, особенности их применения.

## **Задание на занятие №7.**

### **Разработка классов и использование свойств, методов и событий, организация событийного управления**

**Задание 1.** Разработать приложение для моделирования объектов в заданной предметной области. В этом приложении:

1. Создать класс, содержащий данные о некотором объекте (базовом) из заданной предметной области. Эти данные должны быть спрятаны от непосредственного доступа (инкапсулированы). Членами класса должны быть не менее 4-х свойств, 2-х методов, 1-го конструктора и 2-х событий.

Дополнительно создать классы, описывающие вспомогательные объекты, необходимые для моделирования.

2. В главной программе создать экземпляры созданных классов и продемонстрировать возможности изменения свойств объектов, использования их методов и обработки реализованных событий (разработав соответствующие методы обработки событий, объявленных в классах). При реализации событий необходимо обратить особое внимание на организацию событийного взаимодействия объектов.

**Задание 2.** Доработать предыдущее приложение, в котором:

1. Создать еще один класс путем наследования имеющегося класса. В новом классе переопределить хотя бы один из методов базового класса.

2. В главной программе создать экземпляры базового и дочернего классов (а также вспомогательных классов) и продемонстрировать их возможности по изменению свойств, использованию методов и обработки событий. Организовать взаимодействие созданных объектов между собой или с другими вспомогательными объектами с использованием пользовательских событий.

При выборе темы своей работы рекомендуется основываться прежде всего на знании соответствующей предметной области (или хотя бы знакомстве с ней). По согласованию с преподавателем возможно предложение своей темы.

#### **Примерные варианты предметных областей для заданий:**

1. Модель роста растения (цветка, дерева) с учетом погодных условий, смены дня и ночи (времен года), возможного влиянием человека.

2. Модель развития популяции рыбок в аквариуме.

3. Моделирование процесса ловли рыбы на удочку (или другую снасть).

4. Модель выполнения собакой команд кинолога.

5. Модель поведения собаки, охраняющей частный дом (дачу) при взаимодействии с хозяином и посторонним человеком.

6. Модель соревнований спортсменов (бегунов, пловцов, лыжников и т.д.) в эстафете с фиксацией результатов.

7. Модель забега спортсменов-бегунов на стадионе с отображением предварительных и итоговых результатов на табло.
8. Модель работы часов, в которых взаимодействующими объектами являются стрелки.
9. Модель работы часов с будильником во взаимодействии с человеком.
10. Модель работы кофемашины во взаимодействии с человеком.
11. Модель работы принтера при его взаимодействии с компьютером.
12. Модель работы вычислительной системы или сети при выполнении некоторой задачи.
13. Модель взаимодействия нескольких телефонов (смартфонов) через базовую станцию (АТС).
14. Модель солнечной системы (движения и взаимодействия планет).
15. Модель полета космического корабля вокруг Земли (возможно со стыковкой с космической станцией).
16. Модель движения электропоезда до конечной станции с остановками.
17. Модель движения электропоезда до конечной станции с остановками с однопутными участками между станциями.
18. Модель работы уличных фонарей с автоматическим включением в разных условиях.
19. Модель процесса прохождения автомобилем техосмотра (взаимодействие автомобилей с проверочными стендами).
20. Модель работы такси/автобуса (маршруты, посадка и высадка пассажиров, заправка топливом и пр.).
21. Модель проезда транспортными средствами регулируемого перекрестка.
22. Модель проезда транспортными средствами нерегулируемого перекрестка.
23. Модель проезда транспортными средствами железнодорожного проезда.
24. Модель функционирования автомобильной паромной переправы через реку.
25. Модель взаимодействия экскаватора и самосвалов в карьере при погрузке сыпучих грузов.
26. Модель работы автоматического шлагбаума на въезде на парковку ТЦ.
27. Модель полета самолета по маршруту (с возможным бомбометанием, десантированием, аэрофотосъемкой, дозаправкой, пожаротушением и т.д.).
28. Модель взлета и посадки самолетов в аэропорту.
29. Модель движения лодок (судов) по реке с их взаимодействием между собой и речной инфраструктурой.
30. Модель захода морских судов в порт.
31. Модель работы лифта (пассажирского, грузового) в многоэтажном доме при взаимодействии с жильцами этого дома.
32. Модель работы системы Умного дома в различных режимах и в разное время суток (возможно при взаимодействии с человеком).
33. Модель работы Умной теплицы в разное время суток.



34. Модель распространения вирусной инфекции между жителями населенного пункта.

**Примером** может служить следующее приложение, моделирующее работу фонарей уличного освещения (рис.7.1).

Перед моделированием необходимо выбрать тип фонарей (обычный или умный), а также задавая норму средней горизонтальной освещенности (норматив для пешеходных улиц: 4 или 6 лк), расстояние между опорами освещения (обычно 30-45 м) и высоту расположения светильников (обычно 6-9 м) определяется необходимая осветительная установка (обычно 125 или 250 Вт). При этом соотношение расстояния между опорами и высоты расположения светильников обычно составляет 5:1. Поэтому после ввода значений первых двух параметров остальные определяются автоматически.

Далее задается характер погоды на период моделирования (соотношение ясных и облачных дней, вероятность прихода дождя).

С целью повышения удобства тестирования в программе предусмотрена возможность изменения скорости смены дня и ночи, а соответственно и других процессов. Также предусмотрена и возможность появления на улице человека по команде пользователя.

Для непосредственного моделирования работы фонарей создаются:

1. Класс **«Окружающая среда»**, включающий следующие члены:

Поля и свойства:

- состояние погоды (ясно/облачно/пасмурно), определяется по воле случая в рамках заданного характера погоды;
- текущий уровень освещенности (определяет цвет неба, свойство только для чтения), постоянно меняется при смене дня и ночи.

Конструктор: создание объекта «Окружающая среда» с заданными параметрами и запуск метода смены дня и ночи.

Метод: периодическая смена дня и ночи (изменяется текущий уровень освещенности в зависимости от состояния погоды), при этом состояние погоды меняется случайным образом, а при переходе уровня освещенности через значение «2 лк» генерируются события, связанные с включением или выключением фонарей.

События: переход уровня освещенности через значение «2 лк» в одну и другую стороны (в ответ включаются или выключаются фонари).

2. Основной класс **«Фонарь»**, включающий следующие члены:

Поля и свойства:

- номер фонаря на участке улицы (определяется автоматически при создании объекта, свойство только для чтения);
- мощность осветительной установки (обычно 125 или 250 Вт), определяется задаваемыми в основной программе параметрами;
- состояние фонаря (включен/выключен/неисправен, свойство только для чтения).

Конструктор: создание объекта «Фонарь» с заданными параметрами.

Методы: включение/выключение фонаря (изменяется значение свойства состояния и меняется изображение фонаря, рис.7.1, 7.2), при включении

определяются условия для возможной генерации события выхода из строя фонаря.

Событие: выход из строя фонаря (при этом вызывается метод прибытия специалиста для ремонта фонаря из класса «Электрик»), может происходить только при включении фонаря (по воле случая).

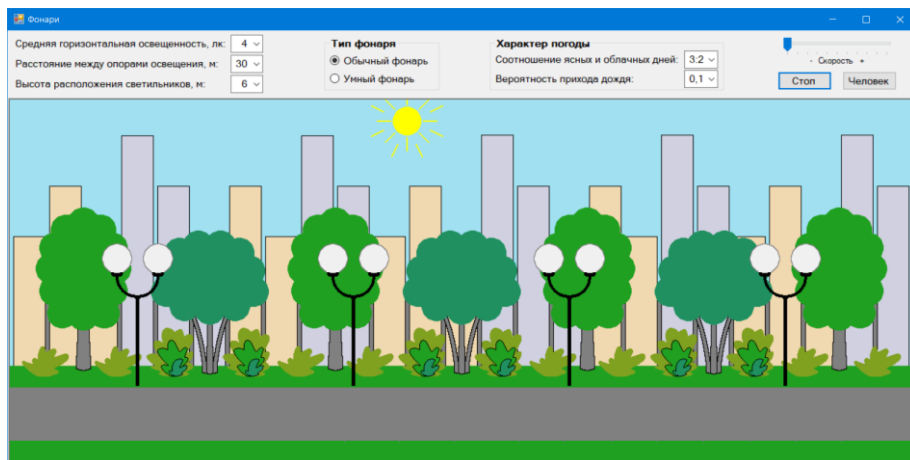


Рис.7.1. Выключенное состояние фонарей в дневное время

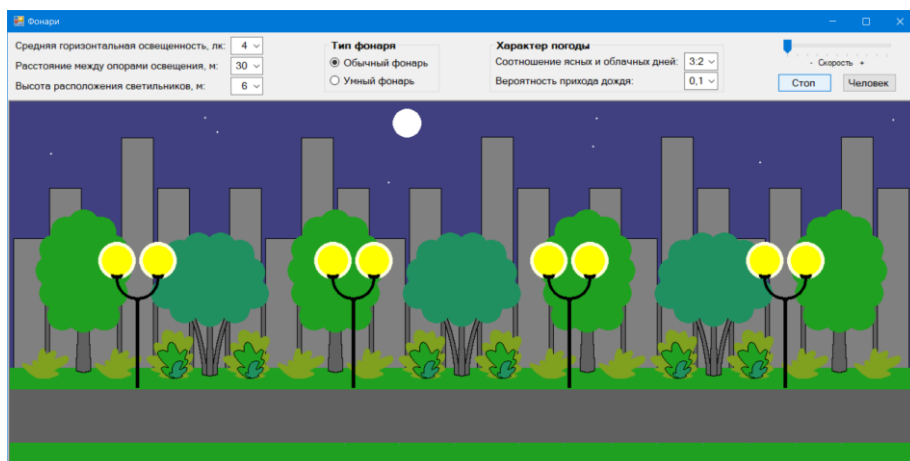


Рис.7.2. Включенное состояние фонарей в ночное время

3. Класс «Умный фонарь» (путем наследования на базе основного класса «Фонарь»), который содержит все члены родительского класса, но в нем:

- добавлены поле и свойство, фиксирующие количество человек в зоне действия фонаря;
- переопределены методы включения/выключения фонаря (добавлено условие наличия человека в зоне действия фонаря).

Умный фонарь отличается от обычного фонаря тем, что включается в темное время суток только при появлении человека в его зоне действия, а после выхода человека из этой зоны выключается (рис.7.3). Зона действия фонаря определяется как удвоенное расстояние между опорами фонарей.

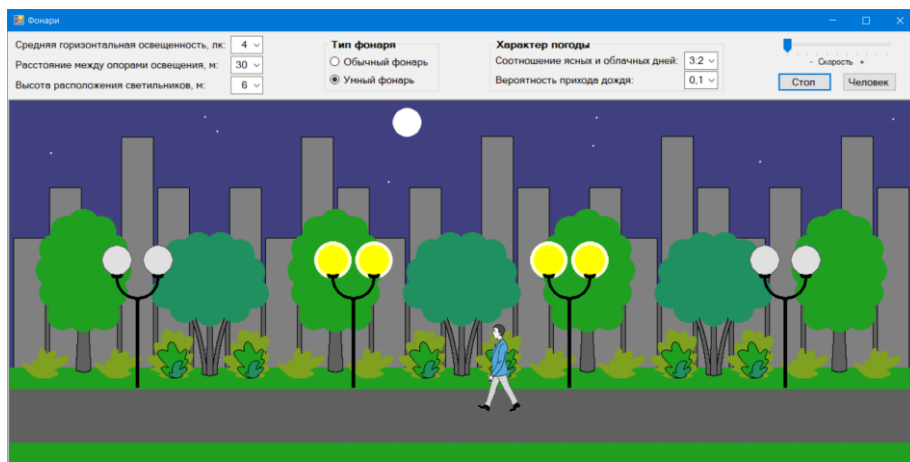


Рис.7.3. Включенное состояние отдельных фонарей в ночное время, в зоне действия которых находится человек

4. Вспомогательный класс «**Человек**», включающий следующие члены:

– Поля и свойства:

- цвет пиджака (светло-серый/синий/черный, определяется случайно, свойство только для чтения);
- цвет брюк (светло-серый/черный, определяется случайно, свойство только для чтения);
- координаты положения человека на улице (используется только одна координата, свойство только для чтения).

Конструктор: создание объекта «Человек» с заданными параметрами и запуск метода движения человека по улице.

Метод: движение человека по улице (изменяются координаты положения человека на улице), при этом после завершения пути генерируется событие и объект уничтожается.

Событие: выход человека за пределы участка улицы (в ответ в основной программе объект уничтожается).

Человек появляется на улице по воле случая или при необходимости по команде пользователя.

5. Класс «**Электрик**» (путем наследования на базе вспомогательного класса «Человек»), который содержит все члены родительского класса, но в нем:

- добавлены поле и свойство, фиксирующие номер фонаря, который требует ремонта;

– переопределены свойства, определяющие цвет пиджака цвет брюк (устанавливаются фиксированные значения элементов форменной одежды – темно синий);

– переопределен метод движения человека по улице (добавлена остановка для ремонта заданного фонаря).

Каждый электрик получает наряд на ремонт только одного фонаря, а этот ремонт выполняется только днем.

## **Контрольные вопросы по теме: «Объектно-ориентированное программирование»**

1. Объектно-ориентированный подход к разработке программ.
2. Основные принципы объектно-ориентированного программирования.
3. Объекты и их основные характеристики.
4. Класс. Определение и описание класса. Структура класса. Статические и экземплярные члены класса.
5. Поля класса. Битовые поля в классах.
6. Свойства. Свойства, доступные только для чтения и только для записи.
7. Методы класса. Перегрузка методов. Конструкторы, деструкторы.
8. Делегаты и события. Связанные и несвязанные делегаты. Свойства событий. Организация событийного управления для объектов, определенных пользователем.
9. В чем заключается смысл абстрагирования?
10. Инкапсуляция данных. Инкапсуляция и свойства объекта.
11. Смысл наследования. Одиночное и множественное наследование классов. Использование конструкторов и деструкторов при наследовании.
12. Полиморфизм. Перегруженные переменные и функции. Механизмы раннего и позднего связывания.
13. Производительность объектных программ.

## Задание на занятие №8.

### Работа с базами данных и организация взаимодействия с Microsoft Office

Разработать приложение для работы с базой данных в заданной предметной области, в котором необходимо предусмотреть:

1. Разграничение прав доступа специалистов к информации из базы данных в соответствии с должностными обязанностями.
2. Отображение информации из базы данных.
3. Элементы управления, необходимые для управления приложением.
4. Возможности для открытия и сохранения базы данных, ввода данных в таблицы с помощью форм, выполнения необходимых запросов, отображения и обработки информации.
5. Возможность импорта информации в базу данных и экспорта информации из нее (из таблицы и в таблицу MS Excel).
6. Возможность формирования необходимых отчетов (в формате MS Word), содержащих форматированный текст, таблицу с расчетами, графические изображения.

Номер наряда	Дата вызова	Срочность	Место вызова	Принял	Передал	Адрес	Открыт
1	12.01.2008	4	Дом	1	Бедюк А.С.	1 Кабельная 181 0 13	Открыт
2	12.01.2008	1	Дом	10	Вахт Л.К.	Айвазовского 25 1 2	Упав
3	12.01.2008	10	Дом	28	Зворыкина Н.Н.	Аносова 13 3 45	Темн
4	13.01.2008	1	Дом	31	Карпова З.А.	Баженова 2 2 69	Кров
5	13.01.2008	1	Дом	1	Мальшева О.Ю.	Вербная 87 3 112	Род
6	13.01.2008	5	Дом	3	Карпова З.А.	Гашека 119 0 25	Алле
7	13.01.2008	1	Дом	10	Вахт Л.К.	Дикого Алексея 125 0 33	Род
8	13.01.2008	3	Дом	28	Бедюк А.С.	Забелина 32 0 33	Пер
9	14.01.2008	8	Дом	3	Мальшева О.Ю.	Водопьянова 47 0 87	Тем
10	14.01.2008	1	Улица	12	Карпова З.А.	Губкина 88 2 45	Изб
11	14.01.2008	1	Дом	2	Олейникова Н.Я.	Защела 132 0 159	Отр
12	14.01.2008	3	Дом	11	Вахт Л.К.	Басманная Новая 55 0 54	Бол
13	14.01.2008	2	Дом	5	Бедюк А.С.	Самотечная 33 0 57	Ари
14	15.01.2008	10	Дом	31	Савельева Т.Д.	Жебрунова 10 0 28	Пер
15	15.01.2008	4	Дом	12	Карпова З.А.	Лавочкина 58 0 89	Зад
16	15.01.2008	1	Учреждение	14	Зворыкина Н.Н.	Инженерная 87 0 47	Отр
17	15.01.2008	1	Улица	12	Олейникова Н.Я.	Анненская 35 0 54	Эпи
18	15.01.2008	3	Улица	3	Зворыкина Н.Н.	Левитана 28 0 118	Выс
19	15.01.2008	1	Улица	7	Карпова З.А.	Ивовая 14 0 11	Без
20	16.01.2008	1	Учреждение	16	Карпова З.А.	Балтий 44 4 76	Пти

Рис.8.1. ИС подстанции скорой помощи, форма администратора

**Примером** такого приложения может служить информационная система (ИС) подстанции скорой помощи, которая включает формы администратора (рис.8.1) и оператора (рис.8.2), а также позволяет формировать отчеты «Карты вызова» (рис.8.3), экспортировать данные из базы данных в таблицы MS Excel (рис.8.4) и др.

The screenshot shows a software window titled "Вызов" (Call) with a blue header bar. Below the header is a navigation bar with buttons for back, forward, and search. The main area contains several input fields and buttons:

- Номер наряда** (Call number): A text box with "1" and a dropdown arrow.
- Дата вызова** (Call date): A date picker showing "12 января 2008 г.".
- Срочность** (Urgency): A text box with "4" and a dropdown arrow.
- Место вызова** (Call location): A dropdown menu showing "Дом".
- Номер бригады** (Ambulance team number): A text box with "1" and a dropdown arrow.
- Принял** (Accepted): A text box with "1" and a dropdown arrow.
- Передал** (Transferred): A text box with "Бедюк А.С.".
- Адрес** (Address): A text box with "1 Кабельная 181 0 13".
- Повод** (Reason): A text box with "Отравление пищевое".
- ФИО пациента** (Patient's name): A text box with "Собакин Ю.А.".
- Оказанные действия** (Actions taken): A text box containing a list of actions:
  - 1. Промывание желудка с помощью зонда.
  - 2. Симптоматическая медикаментозная помощь.
  - 3. Госпитализация.

On the right side of the form, there is a small image of an ambulance and several buttons:

- Сохранить** (Save)
- Распечатать** (Print)
- Печать шаблона** (Print template)
- Поиск** (Search)
- Войти как администратор** (Login as administrator)
- Выход** (Exit)

Рис.8.2. ИС подстанции скорой помощи, форма оператора

Подстанция\_СП.doc - Microsoft Word

Файл Правка Вид Вставка Формат Сервис Таблица Окно Справка

Обычный 14 Ж К Ч 68%

Станция скорой и неотложной медицинской помощи им. А.С.Пучкова

**Подстанция № 31**

ЮАО, Царицыно, ул. Веселая д.31.  
Тел.: 8-495-355-5514.

**Карта вызова**

1	Номер наряда	1
2	Дата вызова	12 января 2008 г.
3	Место вызова	Дом
4	Срочность	4
5	Принял	1
6	Передал	Бедюк А.С.
7	Адрес	1-я Кабельная, 181 0 13
8	Повод	Отравление пищевое
9	ФИО пациента	Собакин Ю.А.
10	Номер бригады	1
11	Оказанные действия	1. Промывание желудка с помощью зонда. 2. Симптоматическая медикаментозная помощь. 3. Госпитализация.

Подпись \_\_\_\_\_

Стр. 1 Разд 1 1/1 На 22см Ст 25 Кол 1 ЗАП ИСПР ВДП ЗАМ русский (Ро

Рис.8.3. ИС подстанции скорой помощи, отчет «Карта вызова»

Microsoft Excel - Подстанция\_СП.xls

Файл Правка Вид Вставка Формат Сервис Данные Окно Справка

Введите вопрос

Arial Cyr 10

Номер бригады

Номер бригады	Специализация	Номер подстанции
1	Педиатрическая	31
2	Педиатрическая	31
3	Врачебная выездная	31
4	По транспортировке рожениц, родильниц и гинекологических больных	31
5	По транспортировке рожениц, родильниц и гинекологических больных	31
6	Линейная	31
7	Линейная	31
8	Линейная	31
9	Линейная	31
10	Линейная	31
11	Линейная	31
12	Линейная	31
13	Врачебная выездная	31
14	Врачебная выездная	31
15	Врачебная выездная	31
16	Врачебная выездная	31
17	Врачебная выездная	31
18	Токсикологическая	31
19	Токсикологическая	31
20	По транспортировке соматических и инфекционных больных	31
21	По транспортировке соматических и инфекционных больных	31
22		

Подстанция \Бригады \МедПерсонал \Водители \Выво

Готово

Рис.8.4. ИС подстанции скорой помощи, экспорт данных в таблицу MS Excel

### Примерная тематика индивидуальных заданий:

1. Разработка БД врачей поликлиники.
2. Разработка БД магазина компьютерных игр.
3. Разработка БД для поставки и реализации товаров.
4. Разработка БД строительной фирмы.
5. Разработка БД фотосалона.
6. Разработка БД автосалона.
7. Разработка БД фирмы заказов продовольственных товаров.
8. Разработка БД книжного магазина.
9. Разработка БД страховой компании.
10. Разработка БД фонотеки.
11. Разработка БД турагентства.
12. Разработка БД фильмотеки.
13. Разработка БД студенческой библиотеки.
14. Разработка БД школьной библиотеки.
15. Разработка БД пункта проката фильмов.
16. Разработка БД спортивных соревнований.
17. Разработка БД управляющей компании (ЖКХ).
18. Разработка БД станции скорой помощи.
19. Разработка БД риэлтерской компании.
20. Разработка БД мебельного салона.



21. Разработка БД компьютерного магазина.
22. Разработка БД зоомагазина.
23. Разработка БД аптеки.
24. Разработка БД музея.
25. Разработка БД театральной кассы.
26. Разработка БД транспортной компании.
27. Разработка БД каршеринговой компании.
28. Разработка БД почтового отделения.
29. Разработка БД ломбарда.
30. Разработка БД отдела полиции.

# Справочные материалы по языку C++ в среде Visual Studio

1. Любые используемые в языке идентификаторы должны принадлежать к какому-либо **типу**. Для всех данных тип определяет способ их внутреннего (машинного) представления, т.е. определяет **размер памяти** для хранения переменной и **диапазон возможных значений**, а также **набор операций**, которые можно осуществлять над данными.

Таблица 9.1. Основные (базовые) типы данных

Тип C++ / тип CLI	Описание (размер в байтах)	Диапазон значений
<b>Целые типы</b>		
<b>bool</b> , System::Boolean	целочисленный тип, логическое значение (1 байт, зависит от платформы)	<b>true</b> (соответствует числу 1) или <b>false</b> (соответствует числу 0).
<b>char</b> , <b>signed char</b> , System::SByte	целочисленный тип, однобайтовый код символа в кодировке ASCII (1 байт)	от -128 до 127.
<b>unsigned char</b> , System::Byte	Короткое целое число без знака (1 байт)	от 0 до 255
<b>short</b> , <b>short int</b> , <b>signed short int</b> , System::Int16	Целое число (2 байта)	от -32 768 до +32 767
<b>unsigned short</b> , <b>unsigned short int</b> , System::UInt16	Целое число без знака (2 байта)	от 0 до 65 535
<b>int</b> , <b>signed int</b> , <b>signed</b> , <b>long int</b> , <b>long</b> , <b>signed long int</b> , System::Int32	Целое число (4 байта)	от -2 147 483 648 до +2 147 483 647
<b>unsigned int</b> , <b>unsigned</b> , <b>unsigned long</b> , <b>unsigned long int</b> , System::UInt32	Целое число без знака (4 байта)	от 0 до 4 294 967 295
<b>long long</b> , <b>long long int</b> , <b>signed long long</b> , <b>signed long long int</b> , System::Int64	Длинное целое число (8 байтов)	от -9 223 372 036 854 775 808 до +9 223 372 036 854 775 807
<b>unsigned long long</b> , <b>unsigned long long int</b> , System::UInt64	Длинное целое число без знака (8 байтов)	от 0 до 18 446 744 073 709 551 615

Вещественные типы		
<b>float</b> , System::Single	Число с плавающей запятой (4 байта: знак – 1 бит, порядок – 8 битов, мантисса – 23 бита), точность – 7-9 знаков	от $-3,402\ 823\ 47 \cdot 10^{38}$ до $-1,401\ 298 \cdot 10^{-45}$ для отрицательных величин и от $+1,401\ 298 \cdot 10^{-45}$ до $+3,402\ 823\ 47 \cdot 10^{38}$ для положительных величин
<b>double</b> , <b>long double</b> , System::Double	Число с плавающей запятой двойной точности (8 байтов: знак – 1 бит, порядок – 11 битов, мантисса – 52 бита), точность – 15-17 знаков	от $-1,797\ 693\ 134\ 862\ 315\ 7 \cdot 10^{308}$ до $-4,940\ 656\ 458\ 412\ 47 \cdot 10^{-324}$ для отрицательных величин и от $4,940\ 656\ 458\ 412\ 47 \cdot 10^{-324}$ до $1,797\ 693\ 134\ 862\ 315\ 7 \cdot 10^{308}$ для положительных величин
Раширенные типы		
<b>wchar_t</b> , System::Char	Расширенный символ (2 байта)	от 0 до 65 535
Структурный тип		
System::Decimal	Десятичное число с фиксированной запятой (16 байтов), точность – 29 значащих цифр	$\pm 79\ 228\ 162\ 514\ 264\ 337\ 593\ 543\ 950\ 335$ – как целое число; как десятичное число может содержать до 28 разрядов после запятой); Наименьшее ненулевое значение: $\pm 10^{-28}$
System::DateTime	Дата и время (8 байтов)	от 00:00:00 1 января 0001 года до 23:59:59 31 декабря 9999 года (значения времени измеряются в тактах, т.е. 100-наносекундных единицах)
Ссылочные типы (значения хранятся в динамической памяти)		
System::String	Строка (последовательность из нуля или более двухбайтовых символов в кодировке Unicode)	представляет собой неизменяемый объект в динамической памяти и рассматривается как массив символов. Любое изменение строки создает новый объект, при этом старый объект не уничтожается
System::Object	Ссылка на объект (4 байта – для 32-bit и 8 байт – для 64-bit платформ)	любая ссылка на объект

2. Оператор присваивания соответствует базовой алгоритмической конструкции «Процесс» и имеет следующий основной формат:

**переменная = выражение;**

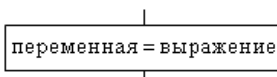


Рис.9.1. Алгоритмическая конструкция «Процесс»

3. Условный оператор **if** имеет две формы записи – сокращенную и полную (рис.9.2), каждая из которых может иметь однострочный или многострочный формат. Форматы записи этого оператора для различных форм имеют вид:

а) сокращенная форма:

– однострочный формат – **if (<условие>) <оператор>;**

– многострочный формат – **if (<условие>)**  
**<оператор>;**

б) полная форма:

– однострочный формат – **if (<условие>) <оператор\_1>; else <оператор\_2>;**

– многострочный формат – **if (<условие>)**  
**<оператор\_1>;**  
**else**  
**<оператор\_2>;**

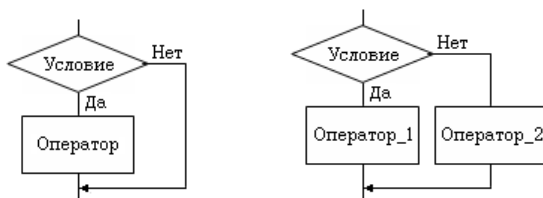


Рис.9.2. Алгоритмические конструкции Условного оператора:  
 сокращенная и полная

4. Оператор выбора варианта **Switch** реализует алгоритмическую конструкцию «Выбор варианта» (рис.9.3). Его формат записи имеет вид:

```

switch (<выражение>)
{ case <константное_выражение_1>:
    <операторы_1>;
    <оператор_перехода_1>;
  case <константное_выражение_2>:
    <операторы_2>;
    <оператор_перехода_2>;
  case <константное_выражение_K>:
    <операторы_K>;
    <оператор_перехода_K>;
  [default:
    <операторы_N>;
    <оператор_перехода_N>;]
}
```

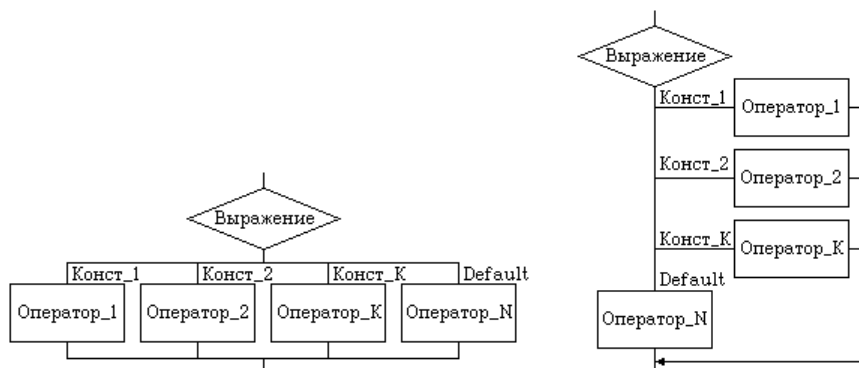


Рис.9.3. Алгоритмические конструкции **Выбора варианта**

5. Различают три типа циклов: «Цикл с параметром», «Цикл с предусловием» и «Цикл с постусловием» (рис.9.4):

а) формат записи оператора цикла с параметром:

```
for (<инициализация_счетчиков>; <условие>; <изменение_счетчиков>)
    <операторы_тела_цикла>;
```

б) формат записи операторов цикла с предусловием:

```
while (<условие>)
    <операторы_тела_цикла>;
```

в) формат записи оператора цикла с постусловием:

```
do
    <операторы_тела_цикла>;
while (<условие>;);
```

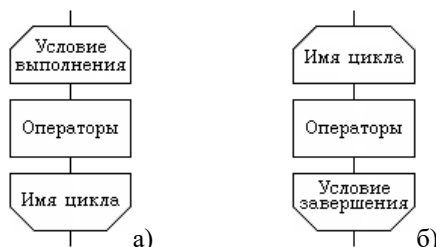


Рис.9.4. Алгоритмические конструкции **Цикла с параметром**, **Цикла с предусловием (а)**, **Цикла с постусловием (б)**

6. Операторы переходов в схемах алгоритмов обозначаются стрелкой, передающей управление. Используются следующие операторы переходов.

а) Оператор завершения текущего блока операторов (в цикле и в операторе выбора варианта). Формат записи этого оператора имеет вид:

**break;**

б) Оператор завершения текущей итерации цикла. Формат записи этого оператора имеет вид:

**continue;**

в) Оператор безусловной передачи управления оператору, помеченному меткой. Метка представляет собой уникальный идентификатор, располагающийся перед помечаемым оператором и отделяемый от него двоеточием. Формат записи этого оператора имеет вид:

**goto <Метка>;**

**...**

**<Метка>: <оператор>;**

г) Оператор завершения работы метода-функции и возврата к вызвавшему его оператору. При необходимости этот оператор обеспечивает возврат функцией своего результирующего значения. Формат записи этого оператора имеет вид:

**return [<выражение>;]**

7. При использовании подпрограмм можно разбивать программные коды на небольшие логически законченные блоки.

Для обозначения начала и конца программы или метода (подпрограммы) в схемах алгоритмов используется символ «Терминатор» (рис.9.5). Описание метода имеет следующий синтаксис:

**[модификаторы] Тип ИмяМетода([Список\_формальных\_параметров])  
{ Операторы\_тела\_метода }**

Для обозначения обращения к методу (подпрограмме) в схемах алгоритмов используется символ «Предопределенный процесс» (рис.9.6). Формат вызова метода имеет вид:

**ИмяМетода(аргумент1, аргумент2, ... аргументN);**

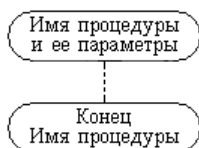


Рис.9.5. Использование символа «Терминатор» в схемах алгоритмов для обозначения начала и конца подпрограммы

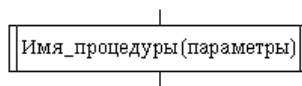


Рис.9.6. Использование символа «Предопределенный процесс» в схемах алгоритмов для обозначения обращения к подпрограмме

8. В программах могут использоваться массивы. **Массив** – это упорядоченная структура данных, содержащая набор переменных (объектов) одного типа, и имеющая имя. Массив может быть **одномерным** или **многомерным** (рис.9.7).

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	a)
--------	--------	--------	--------	--------	--------	--------	----

arrM[0, 0]	arrM[0, 1]	arrM[0, 2]	arrM[0, 3]	arrM[0, 4]	arrM[0, 5]	arrM[0, 6]	б)
arrM[1, 0]	arrM[1, 1]	arrM[1, 2]	arrM[1, 3]	arrM[1, 4]	arrM[1, 5]	arrM[1, 6]	
arrM[2, 0]	arrM[2, 1]	arrM[2, 2]	arrM[2, 3]	arrM[2, 4]	arrM[2, 5]	arrM[2, 6]	
arrM[3, 0]	arrM[3, 1]	arrM[3, 2]	arrM[3, 3]	arrM[3, 4]	arrM[3, 5]	arrM[3, 6]	
arrM[4, 0]	arrM[4, 1]	arrM[4, 2]	arrM[4, 3]	arrM[4, 4]	arrM[4, 5]	arrM[4, 6]	

Рис.9.7. Представление массива: а) одномерного и б) многомерного

Форматы объявления и инициализации массивов:

```
тип имяМассива[количество_элементов];
тип имяМассива[количество_элементов][количество_элементов]...;
тип имяМассива[ ] = { список_значений };
array<тип>^ имяМассива = gnew array<тип>(количество_элементов);
array<тип, кол-во_измерений>^ имяМассива = gnew
    array<тип, количество_измерений>(количество_элементов,
    количество_элементов...);
array<тип>^ имяМассива = { список_значений };
```

Для работы с массивами можно использовать свойства и методы класса **System::Array**.

9. В программах могут использоваться указатели и ссылки.

а) **Указатель** – это переменная, которая хранит адрес другой переменной определенного типа (для 32-х разрядных ЭВМ адресом является 4-байтное целое число без знака).

Форматы объявления и инициализации указателей:

```
тип *имяУказателя;
тип *имяУказателя = &переменная; // & – операция получения адреса
```

Указатели можно сохранять в массиве. Формат объявления массива указателей:

```
тип *имяМассива[количество_элементов];
```

б) **Ссылка** – это имя, применяемое как псевдоним для некоторой переменной, т.е. ссылка хотя и содержит адрес объекта, однако с синтаксической точки зрения ведет себя как объект.

Формат объявления ссылки:

**тип &имяСсылки = имяПеременной;**

в) **Динамическое выделение памяти.** Чтобы произвести увеличение массива во время выполнения программы необходимо сначала выделить достаточный объем памяти с помощью оператора **new**, перенести туда существующие элементы, и лишь затем добавить новые элементы.

Формат динамического выделения памяти под массив и ее освобождения:

```
указатель = new <тип>[<количество_элементов>]; // Выделение памяти  
... // Выполнение операций с массивом  
delete [ ] указатель; // Освобождение памяти  
указатель = nullptr; // Обнуление указателя
```

10. **Строка** представляет упорядоченную коллекцию символов, используемых для представления текста. Язык C++ поддерживает несколько типов строковых переменных:

- строки, представляемые в виде массивов символов (строки в стиле C), реализуют самый быстрый способ работы со строками, но менее удобны в работе;

- строки, представляемые библиотечным классом **string** (строки C++);

- строки, представляемые средой CLR (строки CLR), предоставляют самые широкие возможности и наиболее удобны в работе.

10.1. Строковый тип **System::String** в CLR предназначен для хранения строк текста переменной длины (коллекции объектов **System::Char**). Значения этого типа хранятся в динамической памяти и являются неизменяемыми (т.е. доступные только для чтения). А при попытке изменения значения строки, в действительности в динамической памяти создается новый ее экземпляр (с измененными данными). Максимальный размер такой строки в памяти 2 Гб или более 1 миллиарда символов.

Примеры объявления и инициализации строк (**System::String**):

```
String^ message1; // Объявление строковой переменной  
message1 = ""; // Инициализация строки  
  
// Объявление и инициализация пустой строки  
// с использованием встроенной константы.  
String^ message2 = System::String::Empty;  
  
// Объявление и инициализация строки  
String^ greeting = "Hello World!";  
String^ input = Console::ReadLine();
```



```
String^ surname = "Петров";      // Инициализация
String^ name = "Иван";          // объединением
String^ name = surname + name;   // строк
```

Для повышения производительности при выполнении многократных операций изменения строк используется класс **System::Text::StringBuilder**, который представляет изменяемые строки большого размера, и позволяет заново присваивать значения отдельным элементам строки без создания новой строки.

Примеры объявления и инициализации строк (**System::Text::StringBuilder**):

```
System::Text::StringBuilder sb = gcnew
    System::Text::StringBuilder("abcdef");
System::Text::StringBuilder sb1 = gcnew
    System::Text::StringBuilder(128);
System::Text::StringBuilder sb2 = gcnew
    System::Text::StringBuilder("abcdef", 16);
```

Для доступа к отдельным элементам строки и работы с ними может использоваться структура **System::Char**, которая предоставляет для этого свои поля и методы.

10.2. **Строки в стиле C** представляют собой массив символов (типа **char**), последний элемент которого содержит нулевой символ (**\0**). Так как каждый символ в такой строке занимает один байт, поэтому для размещения в памяти строке необходимо количество байт, на единицу превышающее количество символов в ней.

```
char st1[7]; // Строка содержит 6 символов +1
char st2[7] = {'S', 't', 'r', 'i', 'n', 'g', '\0'};
char st3[] = "String"; // Можно только при инициализации
```

Работа со строкой в стиле **C** может вестись как с массивом. Однако, такие строки нельзя ни присваивать, ни сравнивать при помощи обычных операций, необходимо использовать соответствующие функции стандартной библиотеки.

10.3. Библиотечный класс **string** в **C++** работает немного медленнее строк в стиле **C**, но предоставляет значительно больше удобств для работы со строками. В строках **C++** нет явного признака конца строки, как в **C**. Чтобы пользоваться этим типом, нужно включить в свою программу заголовочный файл **string** с помощью директивы **#include <string>**.

Примеры объявления и инициализации строк типа **string**:

```
string s1;
string s2 = "Hello";
string s3 = s2;
string s4(6, 'F'); // Будет содержать: "FFFFFFF"
```

Функции библиотечного класса **string** позволяют выполнять целый ряд операций над строками.

11. **Регулярное выражение** (regular expression) – это шаблон поиска строк, состоящий из последовательности символов – односимвольных или многосимвольных литералов, операторов или конструкций. Шаблон используется для поиска соответствий в некоторой строке или файле.

Использовать регулярные выражения можно с помощью методов класса **Regex**, которые позволяют выполнять следующие действия:

- Определить, встречается ли во входном тексте шаблон регулярного выражения, с помощью метода **IsMatch**.
- Извлечь из текста одно или все вхождения, соответствующие шаблону регулярного выражения, с помощью методов **Match** или **Matches**. Первый метод возвращает объект **Match**, предоставляющий сведения о совпадении в тексте. Второй метод возвращает коллекцию **MatchCollection**, в которую входят объекты **Match** для всех совпадений, найденных в проанализированном тексте.
- Заменить текст, соответствующий шаблону регулярного выражения, с помощью метода **Replace**.
- Создать массив строк, сформированный из частей входного текста, с помощью метода **Split**, т.е. разделить входную строку в заданных позициях.

12. Для ввода и вывода данных могут использоваться различные элементы управления:

- для ввода: **ListBox** (список), **ListView** (список объектов), **TreeView** (дерево), **NumericUpDown** (числовое поле со стрелками), **DateTimePicker** (выбор даты и времени), **MonthCalendar** (календарь) и др.;
- для вывода: **Label** (надпись), **ToolTip** (всплывающая подсказка), **PictureBox** (поле с рисунком) и др.;
- для ввода и вывода: **TextBox** (текстовое поле), **RichTextBox** (текстовое поле с расширенными возможностями), **ComboBox** (раскрывающийся список).

При этом данные могут подвергаться форматированию с использованием методов **String.Format** или **ToString**.

В схемах алгоритмах для обозначения операций ввода-вывода используется символ «Данные» (рис.9.8).

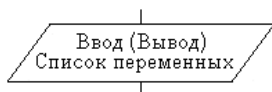


Рис.9.8. Использование символа «Данные» в схемах алгоритмов для обозначения операций ввода-вывода

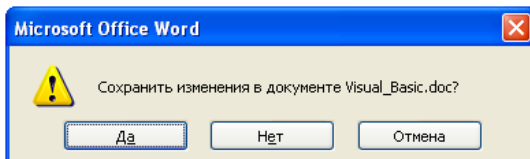


Рис.9.9. Пример окна сообщения **MessageBox**

13. Для вывода сообщений в стандартное окно **MessageBox** (рис.9.9) используется метод **Show** из класса **System.Windows.Forms.MessageBox**. При этом в окне задается текст сообщения, заголовок, набор кнопок, вид пиктограммы, кнопка по умолчанию и некоторые другие параметры.

**ReturnValue = MessageBox.Show([owner], text, [caption], [buttons], [icon], [defaultButton], [options], [helpFilePath, navigator, param]);**

где:

- **owner** – окно, которому принадлежит данное модальное диалоговое окно;
- **text** – текст, отображаемый в окне сообщения;
- **caption** – текст, отображаемый в заголовке окна сообщения;
- **buttons** – определяет набор кнопок в окне сообщения;
- **icon** – определяет вид пиктограммы в окне сообщения;
- **defaultButton** – определяет кнопку по умолчанию в окне сообщения;
- **options** – определяет вариант отображения окна сообщения и способ отображения текста в нем;
- **helpFilePath, navigator, param** – определяют путь и имя файла справки, способ доступа к нему и числовой идентификатор его раздела;
- **ReturnValue** – возвращаемое значение, позволяющее определить нажатую пользователем кнопку в окне сообщения.

14. При работе с файлами принято выделять файлы **последовательного доступа, произвольного доступа и бинарные**.

Типовая процедура работы с файлом включает: **открытие файла, операции с файлом и закрытие файла**.

При открытии и сохранении файлов могут использоваться управляющие элементы **OpenFileDialog** и **SaveFileDialog** из пространства имен **System::Windows::Forms** (стандартные диалоговые окна открытия и сохранения файлов, рис.9.10), которые позволяют просматривать папки и выбирать файлы для открытия или сохранения, что упрощает работу пользователя. Вначале необходимо настроить свойства диалогового окна, а затем производится его вызов с помощью метода **ShowDialog()**.

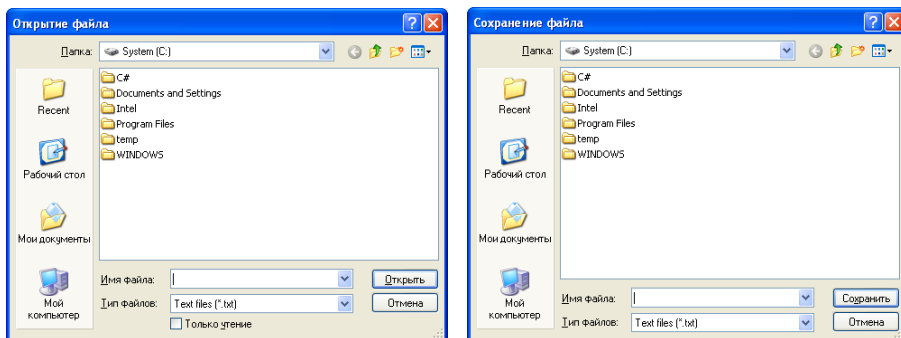
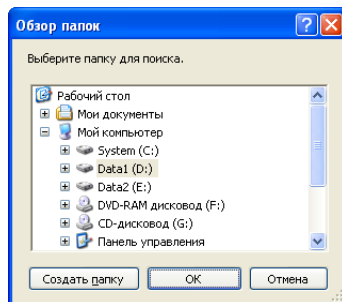


Рис.9.10. Примеры диалоговых окон открытия (а) и сохранения (б) файлов

Для просмотра и выбора папок для дальнейшей работы используется управляющий элемент **FolderBrowserDialog** из пространства имен **System::Windows::Forms** (стандартное диалоговое окно выбора папок, рис.9.11). Перед вызовом диалогового окна с помощью метода **ShowDialog()** также необходимо настроить свойства этого окна.

Рис.9.11. Диалоговое окно обзора папок



Для непосредственной работы с файлами и каталогами в Visual C++ используются свойства и методы классов **File**, **Directory**, **Path**, **FileInfo**, **DirectoryInfo**, **DriveInfo**, **FileStream**, **StreamReader**, **StreamWriter**, **BinaryReader**, **BinaryWriter**, **TextReader**, **TextWriter** из пространства имен **System::IO** .Net Framework.

При записи в файл возможны два способа: **перезапись** и **дозапись**. Второй способ является более универсальным.

15. **Структуры** представляют собой особую разновидность классов (определяемый пользователем тип). В различных языках программирования структуры используются как контейнеры для множества поименованных данных разного типа, называемых **полями**.

Определение структуры выполняется с помощью ключевого слова **struct**:

```
[модификатор] struct ИмяСтруктуры {
    тип1 поле1;
    тип2 поле2;
    ...;
};
```

Формат создания переменной типа структуры:

```
ИмяСтруктуры ИмяПеременной;
```

Инициализация полей структуры может быть выполнена при объявлении переменной типа структуры путем задания начальных значений полей структуры. При этом инициализирующие значения указываются в фигурных скобках и разделяются запятыми.

```
ИмяСтруктуры ИмяПеременной {
    значениеПоля1, // Начальные
    значениеПоля2, // значения
    ...           // полей
};
```

16. В программах можно создавать собственные наборы именованных констант, называемые **перечислениями**. Объявление этого типа данных осуществляется с помощью ключевого слова **enum**:

```
enum class ИмяПеречисления { Константа1, Константа2, ... };
```

Для работы с перечислениями можно использовать методы класса **System::Enum**.

17. При использовании объектно-ориентированного программирования объявление класса выполняется с помощью ключевого слова **class**:

```
[модификатор_доступа] class [<имя_класса>] [: <список_родителей>] {  
    <тело_класса>  
} [<объявления_объектов>;];
```

Объявить переменную типа класса (объект) можно и отдельно, используя название класса в качестве типа данных.

```
<имя_класса> <имя_объекта>;
```

или

```
<имя_класса>^ <имя_объекта> = gnew <Имя_класса>();
```

Члены класса определяются внутри этого класса следующим образом:

а) Определение закрытого поля и открытого свойства, реализующих инкапсуляцию данных:

```
private:
```

```
<тип_поля> <имя_поля>; // Объявление закрытого поля
```

```
public:
```

```
property <тип_свойства> <имя_свойства> { // Определение свойства
```

```
<тип> get() { // Специальный метод чтения значения свойства
```

```
    return <имя_поля>; // Возвращение значения поля
```

```
}
```

```
void set(<тип> value) { // Специальный метод сохранения значения  
    // свойства
```

```
    if (<условие_значения_value_в_поле>)
```

```
        <имя_поля> = value; // Сохранение значения в поле
```

```
}
```

```
}
```

б) Определение методов внутри класса осуществляется так же, как и определение обычного метода (функции) (см. п.6).

в) Объявление делегата выглядит как прототип функции, которому предшествует ключевое слово **delegate**. Оно определяет имя ссылочного типа объекта делегата, а также сигнатуру функции (список параметров и тип

возвращаемого функцией значения), которая может быть ассоциирована с делегатом.

**[модификатор] delegate <тип> <имя\_делегата>(список\_параметров);**

г) Событие является открытым членом ссылочного класса, создается на основе делегата и объявляется с использованием ключевого слова **event** и имени класса делегата.

**event <имя\_делегата>^ <имя\_события>;**

Событие может порождаться (или генерироваться) только классом-владельцем.

**<имя\_события>(список\_параметров);**

Подписка на событие обеспечивает его связь с функцией-обработчиком. Добавление обработчика для некоторого события какого-либо объекта можно выполнить следующим образом:

**<объект\_класса-издателя>-><имя\_события> += gspnew  
<имя\_делегата>(<объект\_класса\_обработчиков>,  
&<имя\_класса\_обработчиков>::<имя\_метода\_обработчика\_события>);**

В программах пользователей можно организовать событийное управление не только применительно к стандартным элементам управления, но и использовать события объектов, определенных пользователями. Для этого необходимо:

- 1) Объявить делегат.
- 2) Создать класс-издателя, содержащий:
  - событие, соответствующее объявленному делегату;
  - метод, включающий логику порождения события и собственно его генерацию.
- 3) Определить метод обработки данного события со списком параметров и типом возвращаемого значения, определенными делегатом (определение этого метода можно разместить внутри класса-подписчика или другого класса).
- 4) Создать объект класса-издателя, содержащий объявление и генерацию события, а затем выполнение подписки на это событие и связывание его с методом-обработчиком (инструкция, обеспечивающая подписку, и метод-обработчик могут быть размещены в отдельном классе-подписчике).

д) При наследовании используется следующий формат объявления класса:

**class <Производный\_класс>:  
[<Спецификатор\_доступа>] <Базовый\_класс> {  
    <Объявления\_членов\_класса>;  
} [<Объявления\_переменных\_через\_запятую>;]**

Для указания, что класс не может использоваться в качестве базового класса (т.е. наследоваться), применяется ключевое слово **sealed** в объявлении класса, например:

```
public sealed class A { ... }
```

Для указания, что класс может использоваться только в качестве базового класса и нельзя создать экземпляр этого класса, используется ключевое слово **abstract** в объявлении класса, например:

```
public abstract class B { ... }
```

# Приложение А.

## Перечень типовых задач

### Общие требования к решению задач.

Разработать приложение для решения поставленной задачи с использованием оконного интерфейса. В этом приложении предусмотреть:

- возможность различных (естественных для конкретной задачи) вариантов ввода исходных данных (с клавиатуры, из файла, методом случайной генерации) по выбору пользователя;
- отображение введенных исходных данных в естественном виде;
- наличие необходимых элементов управления для выбора режимов работы приложения и выполнения заданных функций;
- отображение полученных результатов (в том числе и промежуточных) в естественном виде;
- возможность (при необходимости, по выбору пользователя) сохранения исходных данных и полученных результатов в файле на диске (с задаваемым пользователем именем);
- корректную обработку возможных ошибочных ситуаций.

Для разработанного приложения подготовить необходимые контрольные примеры и файлы для его тестирования.

## 1. Массивы

### 1.1. Одномерные массивы

#### А

1. Дан массив натуральных чисел. Найти сумму элементов, кратных данному  $K$ .
2. В целочисленной последовательности есть нулевые элементы. Создать массив из номеров этих элементов.
3. Дана последовательность целых чисел  $a_1, a_2, \dots, a_n$ . Выяснить, какое число встречается раньше – положительное или отрицательное.
4. Дана последовательность действительных чисел  $a_1, a_2, \dots, a_n$ . Выяснить, будет ли она возрастающей.
5. Дана последовательность натуральных чисел  $a_1, a_2, \dots, a_n$ . Создать массив из четных чисел этой последовательности. Если таких чисел нет, то вывести сообщение об этом факте.
6. Дана последовательность чисел  $a_1, a_2, \dots, a_n$ . Указать наименьшую длину числовой оси, содержащую все эти числа.
7. Дана последовательность действительных чисел  $a_1, a_2, \dots, a_n$ . Заменить все ее члены, большие данного  $Z$ , этим числом. Подсчитать количество замен.
8. Последовательность действительных чисел оканчивается нулем. Найти количество членов этой последовательности.
9. Дан массив действительных чисел, размерность которого  $N$ . Подсчитать, сколько в нем отрицательных, положительных и нулевых элементов.



10. Даны действительные числа  $a_1, a_2, \dots, a_n$ . Поменять местами наибольший и наименьший элементы.
11. Даны целые числа  $a_1, a_2, \dots, a_n$ . Вывести на печать только те числа, для которых выполняется условие  $a_i \leq i$ .
12. Даны натуральные числа  $a_1, a_2, \dots, a_n$ . Указать те, у которых остаток от деления на  $M$  равен  $L$  ( $0 \leq L \leq M - 1$ ).
13. В заданном одномерном массиве поменять местами соседние элементы, стоящие на четных местах, с элементами, стоящими на нечетных.
14. При поступлении в вуз абитуриенты, получившие «двойку» на первом экзамене, ко второму не допускаются. В массиве  $A[n]$  записаны оценки экзаменуемых, полученные на первом экзамене. Подсчитать, сколько человек не допущено ко второму экзамену.
15. Дана последовательность чисел, среди которых имеется один нуль. Вывести на печать все числа, включительно до нуля.
16. В одномерном массиве размещены: в первых элементах значения аргумента, в следующих – соответствующие им значения функции. Напечатать элементы этого массива в  $n$  параллельных столбцов (аргумент и значения функции).
17. Пригодность детали оценивается по размеру  $B$ , который должен соответствовать интервалу  $(A - \delta, A + \delta)$ . Определить, имеются ли в партии из  $N$  деталей бракованные. Если да, то подсчитать их количество, иначе выдать отрицательный ответ.
18. У вас есть доллары. Вы хотите обменять их на рубли. Есть информация о стоимости купли-продажи в банках города. В городе  $N$  банков. Составьте программу, определяющую, какой банк выбрать, чтобы выгодно обменять доллары на рубли.
19. Дан целочисленный массив с количеством элементов  $n$ . Напечатать те его элементы, индексы которых являются степенями двойки (1, 2, 4, 8, 16, ...).

## Б

20. Дан одномерный массив  $A[N]$ .  
Найти:  $\max(a_2, a_4, \dots, a_{2k}) + \min(a_1, a_3, \dots, a_{2k-1})$ .
21. Дана последовательность действительных чисел  $a_1, a_2, \dots, a_n$ . Указать те ее элементы, которые принадлежат отрезку  $[c, d]$ .
22. Дана последовательность целых положительных чисел. Найти произведение только тех чисел, которые больше заданного числа  $M$ . Если таких нет, то выдать сообщение об этом.
23. Последовательность  $a_1, a_2, \dots, a_n$  состоит из нулей и единиц. Поставить в начало этой последовательности нули, а затем единицы.
24. Даны действительные числа  $a_1, a_2, \dots, a_n$ . Среди них есть положительные и отрицательные. Заменить нулями те числа, величина которых по модулю больше максимального числа ( $|a_i| > \max\{a_1, a_2, \dots, a_n\}$ ).
25. Даны действительные числа  $a_1, a_2, \dots, a_n$ .  
Найти  $\max(a_1 + a_{2n}, a_2 + a_{2n-1}, \dots, a_n + a_{n+1})$ .
26. В последовательности действительных чисел  $a_1, a_2, \dots, a_n$  есть только положительные и отрицательные элементы. Вычислить произведение

отрицательных элементов  $P_1$  и произведение положительных элементов  $P_2$ . Сравнить модуль  $P_2$  с модулем  $P_1$  и указать, какое из произведений по модулю больше.

27. Дан массив действительных чисел. Среди них есть равные. Найти первый максимальный элемент массива и заменить его нулем.

28. Дана последовательность действительных чисел  $a_1 \leq a_2 \leq \dots \leq a_n$ . Вставить действительное число  $b$  в нее так, чтобы последовательность осталась неубывающей.

29. Даны целые положительные числа  $a_1, a_2, \dots, a_n$ . Найти среди них те, которые являются квадратами некоторого числа  $m$ .

30. Дана последовательность целых чисел  $a_1, a_2, \dots, a_n$ . Образовать новую последовательность, выбросив из исходной те члены, которые равны  $\min(a_1, a_2, \dots, a_n)$ .

31. У прилавка магазина выстроилась очередь из  $p$  покупателей. Время обслуживания  $i$ -того покупателя равно  $t_i$  ( $i = 1, \dots, n$ ). Определить время  $S_i$  пребывания  $i$ -го покупателя в очереди.

32. Секретный замок для сейфа состоит из 10 расположенных в ряд ячеек, в которые надо вставить игральные кубики. Но дверь открывается только в том случае, когда в любых трех соседних ячейках сумма точек на передних гранях кубиков равна 10. (Игральный кубик имеет на каждой грани от 1 до 6 точек.) Напишите программу, которая разгадывает код замка при условии, что два кубика уже вставлены в ячейки.

33. В массиве целых чисел с количеством элементов  $n$  найти наиболее часто встречающееся число. Если таких чисел несколько, то определить наименьшее из них.

34. Каждый солнечный день улитка, сидящая на дереве, поднимается вверх на 2 см, а каждый пасмурный день опускается вниз на 1 см. В начале наблюдения улитка находится в  $A$  см от земли на  $B$ -метровом дереве. Имеется 30-элементный массив, содержащий сведения о том, был ли соответствующий день наблюдения пасмурным или солнечным. Написать программу, определяющую местоположение улитки к концу 30-го дня наблюдения.

35. Дан целочисленный массив с количеством элементов  $n$ . «Сожмите» массив, выбросив из него каждый второй элемент (дополнительный массив при этом не использовать).

36. Задан массив, содержащий несколько нулевых элементов. Сжать его, выбросив эти элементы.

37. Задан массив с количеством элементов  $N$ . Сформируйте два массива: в первый включите элементы исходного массива с четными номерами, а во второй – с нечетными.

38. Дана последовательность целых чисел  $a_1, a_2, \dots, a_n$ . Указать пары чисел  $a_i, a_j$ , таких, что  $a_i + a_j = m$ .

39. Даны целые числа  $a_1, a_2, \dots, a_n$ . Наименьший член этой последовательности заменить целой частью среднего арифметического всех членов, остальные члены оставить без изменения. Если в последовательности несколько наименьших членов, то заменить последний по порядку.

40. Даны целые числа  $a_1, a_2, \dots, a_n$  и  $b_1, b_2, \dots, b_n$ . Преобразовать последовательность  $b_1, b_2, \dots, b_n$  по правилу: если  $a_i \leq 0$ , то  $b_i$  увеличить в 10 раз, иначе  $b_i$  заменить нулем ( $i = 1, 2, \dots, n$ ).

41. Даны действительные числа  $a_1, a_2, \dots, a_n$ . Требуется умножить все члены последовательности  $a_1, a_2, \dots, a_n$  на квадрат ее наименьшего члена, если  $a_k \geq 0$ , и на квадрат ее наибольшего члена, если  $a_k \leq 0$  ( $1 \leq k \leq n$ ).

42. Даны координаты  $n$  точек на плоскости:  $(X_1, Y_2), \dots, (X_n, Y_n)$  ( $n \leq 30$ ). Найти номера пары точек, расстояние между которыми наибольшее (считать, что такая пара единственная).

43. Дана последовательность из  $n$  различных целых чисел. Найти сумму ее членов, расположенных между максимальным и минимальным значениями (в сумму включить и оба этих числа).

44. Японская радиокompания провела опрос  $N$  радиослушателей по вопросу: «Какое животное Вы связываете с Японией и японцами?» Составить программу получения  $k$  наиболее часто встречающихся ответов и их долей (в процентах).

45. Дан массив, состоящий из  $n$  натуральных чисел. Образовать новый массив, элементами которого будут элементы исходного, оканчивающиеся па цифру  $k$ .

46. Дан массив целых чисел. Найти в этом массиве минимальный элемент  $N$  и максимальный элемент  $M$ . Получить в порядке возрастания все целые числа из интервала  $(N; M)$ , которые не входят в данный массив.

47. Дано действительное число  $x$  и массив  $A[n]$ . В массиве найти два члена, среднее арифметическое которых ближе всего к  $x$ .

48. Даны две последовательности  $a_1, a_2, \dots, a_n$  и  $b_1, b_2, \dots, b_m$  ( $m < n$ ). В каждой из них члены различны. Верно, ли что все члены второй последовательности входят в первую последовательность?

49. Напишите программу, входными данными которой является возраст  $n$  человек. Программа подсчитывает количество людей, возраст которых находится в интервале 10 лет, а именно:

<..> человек имеет возраст в диапазоне 0-10 лет;

<..> человек имеет возраст в диапазоне 10-20 лет и т.д.

## В

50. В одномерном массиве все отрицательные элементы переместить в начало массива, а остальные – в конец с сохранением порядка следования. Дополнительный массив заводить не разрешается.

51. В одномерном массиве с четным количеством элементов  $(2N)$  находятся координаты  $N$  точек плоскости. Они располагаются в следующем порядке:  $x_1, y_1, x_2, y_2, x_3, y_3$ , и т.д. Определить минимальный радиус круга с центром в начале координат, который содержит все точки.

52. В одномерном массиве с четным количеством элементов  $(2N)$  находятся координаты  $N$  точек плоскости. Они располагаются в следующем порядке:  $x_1, y_1, x_2, y_2, x_3, y_3$ , и т.д. Определить кольцо с центром в начале координат, которое содержит все точки.

53. В одномерном массиве с четным количеством элементов ( $2N$ ) находятся координаты  $N$  точек плоскости. Они располагаются в следующем порядке:  $x_1, y_1, x_2, y_2, x_3, y_3$ , и т.д. ( $x_i, y_i$  – целые). Определить номера точек, которые могут являться вершинами квадрата.

54. В одномерном массиве с четным количеством элементов ( $2N$ ) находятся координаты  $N$  точек плоскости. Они располагаются в следующем порядке:  $x_1, y_1, x_2, y_2, x_3, y_3$ , и т.д. Определить номера точек, которые могут являться вершинами равнобедренного треугольника.

55. Задан целочисленный массив размерности  $N$ . Есть ли среди элементов массива простые числа? Если да, то вывести номера этих элементов.

56. Дан одномерный массив чисел. Найти количество различных чисел этого массива.

57. Дан одномерный массив чисел, среди элементов которого есть одинаковые. Создать новый массив из различных элементов исходного массива.

58. Дан массив из  $n$  четырехзначных натуральных чисел. Вывести на экран только те, у которых сумма первых двух цифр равна сумме двух последних.

59. Даны две последовательности целых чисел  $a_1, a_2, \dots, a_n$  и  $b_1, b_2, \dots, b_m$ . Все члены последовательностей – различные числа. Найти, сколько членов первой последовательности совпадают с членами второй последовательности.

60. Даны два упорядоченные массива  $A$  и  $B$ . Образовать из элементов этих массивов упорядоченный массив  $C$ .

61. В массиве  $A$  каждый элемент равен 0, 1, 2. Переставить элементы массива так, чтобы сначала располагались все нули, затем все единицы, и, наконец все двойки.

62. Дан массив  $A$ . Найти длину самой длинной последовательности подряд идущих элементов массива, равных нулю.

63. Дан целочисленный массив  $A$  и число  $M$ . Найти такое подмножество подряд идущих элементов массива, сумма значений элементов, которых равна  $M$ .

64. Даны два целочисленные массива. Определить, можно ли в первом из них выбрать такие  $k$  идущих подряд элементов  $X_j, X_{j+1}, \dots, X_{k-1}$ , чтобы  $X_j = Y_1, X_{j+1} = Y_2, \dots, X_{k-1} = Y_k$ .

65. Найти длину самой длинной «пилообразной» (зубьями вверх) последовательности подряд идущих чисел  $X_k < X_{k+1} > X_{k+2} > \dots > X_{k+m} < X_{k+m+1} < \dots < X_n$ .

66. Дан массив  $A$ . Найти отрезок массива максимальной длины, в котором первое число равно последнему, второе – предпоследнему и т.д. Вывести этот отрезок и его длину.

67. Дан массив  $A$ . Циклически сдвинуть элементы массива на  $K$  элементов вправо (влево).

68. На плоскости  $n$  точек заданы своими координатами и также дана окружность радиуса  $R$  с центром в начале координат. Указать множество всех треугольников с вершинами в заданных точках, пересекающихся с

окружностью; множество всех треугольников, содержащихся внутри окружности.

69. В одномерном массиве с четным количеством элементов ( $2N$ ) находятся координаты  $N$  точек плоскости. Они располагаются в следующем порядке:  $x_1, y_1, x_2, y_2, x_3, y_3$ , и т.д. Найти номера самых удаленных друг от друга точек и наименее удаленных друг от друга точек.

70. В одномерном массиве с четным количеством элементов ( $2N$ ) находятся координаты  $N$  точек плоскости. Они располагаются в следующем порядке:  $x_1, y_1, x_2, y_2, x_3, y_3$ , и т.д. Определить три точки, которые являются вершинами треугольника, для которого разность числа точек вне его и внутри является минимальной.

## 1.2. Сортировка массивов

71. Заданы два одномерных массива с различным количеством элементов и натуральное число  $k$ . Объединить их в один массив, включив второй массив между  $k$ -м и  $(k+1)$ -м элементами первого, не используя дополнительный массив.

72. Даны два массива  $a_1 \leq a_2 \leq \dots \leq a_n$  и  $b_1 \leq b_2 \leq \dots \leq b_n$ . Образовать из них новый массив так, чтобы он тоже был неубывающим (дополнительный массив не использовать).

73. **Сортировка выбором.** Дан массив чисел  $a_1, a_2, \dots, a_n$ . Требуется переставить его элементы так, чтобы они были расположены по убыванию. Для этого в массиве, начиная с первого, выбирается наибольший элемент и ставится на первое место, а первый – на место наибольшего. Затем, начиная со второго, эта процедура повторяется. Написать алгоритм сортировки выбором.

74. **Сортировка обменами.** Дан массив чисел  $a_1, a_2, \dots, a_n$ . Требуется переставить его элементы в порядке возрастания. Для этого сравниваются два соседних элемента  $a_i$  и  $a_{i+1}$ . Если  $a_i > a_{i+1}$ , то делается перестановка. Так продолжается до тех пор, пока все элементы не станут расположены в порядке возрастания. Составить алгоритм сортировки, подсчитывая при этом количество перестановок.

75. **Сортировка вставками.** Дан массив чисел  $a_1, a_2, \dots, a_n$ . Требуется переставить его элементы в порядке возрастания. Делается это следующим образом. Пусть  $a_1, a_2, \dots, a_i$  – упорядоченная последовательность, т.е.  $a_1 \leq a_2 \leq \dots \leq a_i$ . Берется следующее число  $a_{i+1}$  и вставляется в последовательность так, чтобы новая последовательность была также возрастающей. Процесс производится до тех пор, пока все элементы от  $i+1$  до  $n$  не будут перебраны.

76. **Сортировка Шелла.** Дан массив  $n$  действительных чисел. Требуется упорядочить его по возрастанию. Делается это следующим образом: сравниваются два соседних элемента  $a_i$  и  $a_{i+1}$ . Если  $a_i \leq a_{i+1}$ , то продвигаются на один элемент вперед. Если  $a_i > a_{i+1}$ , то производится перестановка и сдвигаются на один элемент назад. Составить алгоритм этой сортировки.

77. Пусть даны неубывающая последовательность действительных чисел  $a_1 \leq a_2 \leq \dots \leq a_n$  и действительные числа  $b_1 \leq b_2 \leq \dots \leq b_m$ . Требуется указать те места, на которые нужно вставлять элементы последовательности  $b_1, b_2, \dots, b_m$

в первую последовательность так, чтобы новая последовательность оставалась возрастающей.

78. Даны дроби  $p_1/q_1, p_2/q_2, \dots, p_n/q_n$  ( $p_i, q_i$  – натуральные). Составить программу, которая приводит эти дроби к общему знаменателю и упорядочивает их в порядке возрастания.

79. **Алгоритм фон Неймана.** Упорядочить массив  $a_1, a_2, \dots, a_n$  по неубыванию с помощью алгоритма сортировки слияниями: каждая пара соседних элементов сливается в одну группу из двух элементов (последняя группа может состоять из одного элемента), затем каждая пара соседних двухэлементных групп сливается в одну четырехэлементную группу и т.д. При каждом слиянии новая укрупненная группа упорядочивается.

### 1.3. Двумерные массивы

Сформировать квадратную матрицу порядка  $n$  по заданному образцу:

80.

1	2	3	...	n
n	n-1	n-2	...	1
1	2	3	...	n
n	n-1	n-2	...	1
...	...	...	...	...
n	n-1	n-2	...	1

( $n$  – четное).

81.

0	0	0	...	0	0	1
0	0	0	...	0	2	0
0	0	0	...	3	0	0
...	...	...	...	...	...	...
0	n-1	0	...	0	0	0
n	0	0	...	0	0	0

82.

n	0	0	...	0	0	0
0	n-1	0	...	0	0	0
0	0	n-2	...	0	0	0
...	...	...	...	...	...	...
0	0	0	...	0	2	0
0	0	0	...	0	0	1

83.

1×2	0	0	...	0	0	0
0	2×3	0	...	0	0	0
0	0	3×4	...	0	0	0
...	...	...	...	...	...	...
0	0	0	...	0	(n-1)×n	0
0	0	0	...	0	0	n×(n+1)

84.

1	1	1	...	1	1	1
1	0	0	...	0	0	1
1	0	0	...	0	0	1
...	...	...	...	...	...	...
1	0	0	...	0	0	1
1	1	1	...	1	1	1

85.

1	1	1	...	1	1	1
2	2	2	...	2	2	0
3	3	3	...	3	0	0
...	...	...	...	...	...	...
n-1	n-1	0	...	0	0	0
n	0	0	...	0	0	0

86.

1	1	1	...	1	1	1
0	1	1	...	1	1	0
0	0	1	...	1	0	0
...	...	...	...	...	...	...
2	3	4	...	n-1	n	0
1	2	3	...	n-2	n-1	n

87.

1	0	0	...	0	0	1
1	1	0	...	0	1	1
1	1	1	...	1	1	1
...	...	...	...	...	...	...
n-1	n	0	...	0	0	0
n	0	0	...	0	0	0

88.

n	0	0	...	0	0	0
n-1	n	0	...	0	0	0
n-2	n-1	n	...	0	0	0
...	...	...	...	...	...	...
2	3	4	...	n-1	n	0
1	2	3	...	n-2	n-1	n

89.

1	2	3	...	n-2	n-1	n
2	3	4	...	n-1	n	0
3	4	5	...	n	0	0
...	...	...	...	...	...	...
n-1	n	0	...	0	0	0
n	0	0	...	0	0	0

90.

1	0	0	...	0	0	n
0	2	0	...	0	n-1	0
0	0	3	...	n-2	0	0
...	...	...	...	...	...	...
0	2	0	...	0	n-1	0
1	0	0	...	0	0	n

91.

1	2	3	...	n-2	n-1	n
2	1	2	...	n-3	n-2	n-1
3	2	1	...	n-4	n-3	n-2
...	...	...	...	...	...	...
n-1	n-2	n-3	...	2	1	2
n	n-1	n-2	...	3	2	1

92. Построить квадратную матрицу порядка  $2n$  :

n				n				
1	1	...	1	2	2	...	2	}
1	1	...	1	2	2	...	2	
...	...	...	...	...	...	...	...	
1	1	...	1	2	2	...	2	
3	3	...	3	4	4	...	4	}
3	3	...	3	4	4	...	4	
...	...	...	...	...	...	...	...	
3	3	...	3	4	4	...	4	

93. Дано действительное число  $x$ .  
Получить квадратную матрицу  
порядка  $n+1$  :

1	x	x <sup>2</sup>	...	x <sup>n-2</sup>	x <sup>n-1</sup>	x <sup>n</sup>
x	0	0	...	0	0	x <sup>n-1</sup>
x <sup>2</sup>	0	0	...	0	0	x <sup>n-2</sup>
...	...	...	...	...	...	...
x <sup>n-1</sup>	0	0	...	0	0	x
x <sup>n</sup>	x <sup>n-1</sup>	x <sup>n-2</sup>	...	x <sup>2</sup>	x	1

94. Даны действительные числа  
 $a_1, a_2, \dots, a_n$ . Получить квадратную  
матрицу порядка  $n$  :

a <sub>1</sub>	a <sub>2</sub>	a <sub>3</sub>	...	a <sub>n-2</sub>	a <sub>n-1</sub>	a <sub>n</sub>
a <sub>2</sub>	a <sub>3</sub>	a <sub>4</sub>	...	a <sub>n-1</sub>	a <sub>n</sub>	a <sub>1</sub>
a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>	...	a <sub>n</sub>	a <sub>1</sub>	a <sub>2</sub>
...	...	...	...	...	...	...
a <sub>n-1</sub>	a <sub>n</sub>	a <sub>1</sub>	...	a <sub>n-4</sub>	a <sub>n-3</sub>	a <sub>n-2</sub>
a <sub>n</sub>	a <sub>1</sub>	a <sub>2</sub>	...	a <sub>n-3</sub>	a <sub>n-2</sub>	a <sub>n-1</sub>

95. Получить матрицу:

1	2	3	...	9	10
0	1	2	...	8	9
0	0	1	...	7	8
...	...	...	...	...	...
0	0	0	...	0	1

96. Получить матрицу:

1	0	...	0	1
0	1	...	1	0
...	...	...	...	...
0	1	...	1	0
1	0	...	0	1

97. Составить программу, которая заполняет квадратную матрицу порядка  $n$  натуральными числами  $1, 2, 3, \dots, n^2$ , записывая их в нее «по спирали». Например, для  $n = 5$  получаем следующую матрицу:

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

98. Дана действительная квадратная матрица порядка  $2n$ . Получить новую матрицу, переставляя ее блоки размера  $n \times n$  по часовой стрелке, начиная с блока в левом верхнем углу.

99. Дана действительная квадратная матрица порядка  $2n$ . Получить новую матрицу, переставляя ее блоки размера  $n \times n$  крест-накрест.

100.

101.

Дан линейный массив  $x_1, x_2, \dots, x_{n-1}, x_n$ . Получить действительную квадратную матрицу порядка  $n$ :

$x_1$	$x_2$	$\dots$	$x_{n-1}$	$x_n$	1	1	$\dots$	1	1
$x_1^2$	$x_2^2$	$\dots$	$x_{n-1}^2$	$x_n^2$	$x_1$	$x_2$	$\dots$	$x_{n-1}$	$x_n$
$x_1^3$	$x_2^3$	$\dots$	$x_{n-1}^3$	$x_n^3$	$x_1^2$	$x_2^2$	$\dots$	$x_{n-1}^2$	$x_n^2$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$x_1^n$	$x_2^n$	$\dots$	$x_{n-1}^n$	$x_n^n$	$x_1^{n-1}$	$x_2^{n-1}$	$\dots$	$x_{n-1}^{n-1}$	$x_n^{n-1}$

102.

103.

Получить квадратную матрицу порядка  $n$ :

1	2	$\dots$	$n-1$	$n$	0	0	0	$\dots$	0	0
$n+1$	$n+2$	$\dots$	$2n-1$	$2n$	0	1	0	$\dots$	0	0
$2n+1$	$2n+2$	$\dots$	$3n-1$	$3n$	0	0	2	$\dots$	0	0
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$(n-1)n+1$	$(n-1)n+2$	$\dots$	$nn-1$	$nn$	0	0	0	$\dots$	0	$n-1$

104. Магическим квадратом порядка  $n$  называется квадратная матрица размера  $n \times n$ , составленная из чисел  $1, 2, \dots, n^2$  так, что суммы по каждому столбцу, каждой строке и каждой из двух больших диагоналей равны между собой. Построить такой квадрат. Пример магического квадрата порядка 3:

6	1	8
7	5	3
2	9	4

105. Вычислить сумму и число положительных элементов матрицы  $A[N, N]$ , находящихся над главной диагональю.

106. Дана вещественная матрица  $A$  размера  $n \times m$ . Определить  $k$  – количество «особых» элементов массива  $A$ , считая его элемент особым, если он больше суммы остальных элементов его столбца.

107. Задана квадратная матрица. Переставить строку с максимальным элементом на главной диагонали со строкой с заданным номером  $m$ .



108. Дана матрица  $B[N, M]$ . Найти в каждой строке матрицы максимальный и минимальный элементы и поменять их с первым и последним элементом строки соответственно.

109. Дана целая квадратная матрица  $n$ -го порядка. Определить, является ли она магическим квадратом, т.е. такой, в которой суммы элементов во всех строках и столбцах одинаковы.

110. Элемент матрицы назовем седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце или, наоборот, является наибольшим в своей строке и наименьшим в своем столбце. Для заданной целой матрицы размером  $n \times m$  напечатать индексы всех ее седловых точек.

111. Дана вещественная матрица размером  $n \times m$ . Переставляя ее строки и столбцы, добиться того, чтобы наибольший элемент (или один из них) оказался в верхнем левом углу.

112. Определить, является ли заданная целая квадратная матрица  $n$ -го порядка симметричной (относительно главной диагонали).

113. Дана целочисленная квадратная матрица. Найти в каждой строке наибольший элемент и поменять его местами с элементом главной диагонали.

114. Упорядочить по возрастанию элементы каждой строки матрицы размером  $n \times m$ .

115. Задана матрица размером  $n \times m$ . Найти максимальный по модулю элемент матрицы. Переставить строки и столбцы матрицы таким образом, чтобы найденный элемент был расположен на пересечении  $k$ -й строки и  $k$ -го столбца.

116. Дана квадратная матрица  $A[N, N]$ . Записать на место отрицательных элементов матрицы нули, а на место положительных – единицы. Вывести на печать нижнюю треугольную матрицу в общепринятом виде.

117. Дана действительная матрица размером  $n \times m$ , все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы элемента с найденным значением.

118. Дана действительная квадратная матрица порядка  $N$  ( $N$  – нечетное), все элементы которой различны. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении этих диагоналей.

119. Для заданной квадратной матрицы сформировать одномерный массив из ее диагональных элементов. Найти след матрицы, суммируя элементы одномерного массива. Преобразовать исходную матрицу по правилу: четные строки разделить на полученное значение, нечетные оставить без изменения.

120. Задана квадратная матрица. Получить транспонированную матрицу, т.е. матрицу, где столбцы и строки меняются местами.

121. Квадратная матрица, симметричная относительно главной диагонали, задана верхним треугольником в виде одномерного массива. Восстановить исходную матрицу и напечатать по строкам.

122. Задана матрица порядка  $n$  и число  $k$ . Разделить элементы  $k$ -й строки на диагональный элемент, расположенный в этой строке.
123. Для целочисленной квадратной матрицы найти число элементов, кратных  $k$ , и наибольший из полученных результатов.
124. Найти наибольший и наименьший элементы прямоугольной матрицы и поменять их местами.
125. Дана прямоугольная матрица. Найти строку с наибольшей и наименьшей суммой элементов. Вывести на печать найденные строки и суммы их элементов.
126. В данной действительной квадратной матрице порядка  $n$  найти сумму элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.
127. В данной действительной квадратной матрице порядка  $n$  найти наибольший по модулю элемент. Получить квадратную матрицу порядка  $n-1$  путем отбрасывания из исходной матрицы строки и столбца, на пересечении которых расположен элемент с найденным значением.
128. Дана действительная квадратная матрица порядка  $n$ . Преобразовать матрицу по правилу: строку с номером  $n$  сделать столбцом с номером  $n$ , а столбец с номером  $n$  – строкой с номером  $n$ .
129. Пусть дана действительная матрица размером  $n \times m$ . Требуется преобразовать матрицу: поэлементно вычесть последнюю строку из всех строк, кроме последней.
130. Определить номера тех строк целочисленной матрицы  $A[N, K]$ , которые совпадают с массивом  $D[K]$ . Если таких строк нет, выдать соответствующее сообщение.
131. Определить наименьший элемент каждой четной строки матрицы  $A[M, N]$ .
132. Расположить столбцы матрицы  $D[M, N]$  в порядке возрастания элементов  $k$ -ой строки ( $1 \leq k \leq M$ ).
133. Определить номера строк матрицы  $R[M, N]$ , хотя бы один элемент которых равен  $s$ , и элементы этих строк умножить на  $d$ .
134. Матрица  $A[N, M]$  ( $M$  кратно 4) разделена по вертикали на две половины. Определить сумму элементов каждого нечетного столбца левой половины и сумму элементов каждого четного столбца правой половины матрицы  $A$ .
135. Дана квадратная целочисленная матрица порядка  $n$ . Сформировать результирующий одномерный массив, элементами которого являются строчные суммы тех строк, которые начинаются с  $k$  идущих подряд положительных чисел.
136. Дана матрица  $A$ . В каждой строке матрицы найти элемент с минимальным значением, затем среди этих значений найти максимальное значение. Напечатать элементы строки, в которой расположено найденное значение, и ее номер.
137. Дан двумерный массив  $A$ , каждый элемент которого равен 0, 1, 5 или 11. Подсчитать в нем количество четверок  $(A_{j,k}, A_{j,k+1}, A_{j+1,k}, A_{j+1,k+1})$  в каждой из которых все элементы различные.

138. Дан двумерный массив  $A$ . Каждая строка массива упорядочена по не возрастанию. Найти числа, одновременно присутствующие во всех строках массива.

139. Дан двумерный массив  $A$ . Заменить нулями элементы массива, стоящие в строках или столбцах, где имеются нули.

140. «Тестирование коллектива». Пусть целочисленная матрица размером  $n \times m$  содержит информацию об учениках некоторого класса из  $n$  человек. В первом столбце проставлена масса (кг), во втором – рост (см), в третьем – успеваемость (средний балл) и т.д. (используйте свои дополнительные показатели). Ученик называется *среднестатистическим* по  $k$ -му параметру (*уникальным* по  $k$ -му параметру), если на нем достигается минимум (максимум) модуля разности среднего арифметического чисел из  $k$ -го столбца и значения  $k$ -го параметра этого ученика. Ученик называется самым уникальным (самым средним), если он уникален (является среднестатистическим) по самому большому количеству параметров. По данной матрице определить самых уникальных учеников и самых средних.

141. Лабиринт задан квадратной матрицей  $A$ .  $A_k = 0$ , если клетка «проходима»;  $A_k = 1$ , если клетка «непроходима». Начальное положение путника задается в проходимой клетке  $A = 0$ . Путник может перемещаться из одной проходимой клетки в другую, если они имеют общую сторону. Путник выходит из лабиринта, когда попадает в граничную клетку. Может ли путник выйти из лабиринта? Если может, то напечатать путь от выхода (в виде координат точек на маршруте) до начального положения путника. Путь должен иметь минимальную длину.

## 2. Подпрограммы

Во всех задачах этого раздела необходимо использовать подпрограммы.

### 2.1. Нерекурсивные подпрограммы

**A**

142. Треугольник задан координатами своих вершин. Составить программу вычисления его площади.

143. Составить программу нахождения наибольшего общего делителя и наименьшего общего кратного двух натуральных чисел:

$$(НОК(A, B) = \frac{A \times B}{НОД(A, B)}).$$

144. Составить программу нахождения наибольшего общего делителя четырех натуральных чисел.

145. Составить программу нахождения наименьшего общего кратного трех натуральных чисел.

146. Написать программу нахождения суммы большего и меньшего из 3-х чисел.

147. Вычислить площадь правильного шестиугольника со стороной  $a$ , используя подпрограмму вычисления площади треугольника.

148. На плоскости заданы своими координатами  $n$  точек. Составить программу, определяющую между какими из пар точек самое большое расстояние (координаты точек занести в массив).

149. Проверить, являются ли данные три числа взаимно простыми.

150. Написать программу вычисления суммы факториалов всех нечетных чисел от 1 до 9.

151. Даны две дроби  $A/B$  и  $C/D$  ( $A, B, C, D$  – натуральные числа). Составить программу:

- деления дроби на дробь;
- умножения дроби на дробь;
- сложения этих дробей.

Ответ должен быть несократимой дробью.

152. На плоскости заданы своими координатами  $n$  точек. Создать матрицу, элементами которой являются расстояние между каждой парой точек.

153. Даны числа  $X, Y, Z, T$  – длины сторон четырехугольника. Вычислить его площадь, если угол между сторонами длиной  $X$  и  $Y$  – прямой.

154. Сформировать массив  $X(n)$ ,  $n$ -й член которого определяется формулой  $X(n) = 1 / n!$ .

155. Составить программу вычисления суммы факториалов всех четных чисел от  $m$  до  $n$ .

156. Заменить отрицательные элементы линейного массива их модулями, не пользуясь стандартной функцией вычисления модуля. Подсчитать количество произведенных замен.

157. Дан массив  $A(n)$ . Сформировать массив  $B(m)$ , элементами которого являются большие из рядом стоящих в массиве  $A$  чисел. Например, массив  $A$  состоит из элементов 1, 3, 5, -2, 0, 4, 0. Элементами массива  $B$  будут 3, 5, 4.

158. Дан массив  $A(n)$  ( $n$  – четное). Сформировать массив  $B(m)$ , элементами которого являются средние арифметические соседних пар рядом стоящих в массиве  $A$  чисел. Например, массив  $A$  состоит из элементов 1, 3, 5, -2, 0, 4, 0, 3. Элементами массива  $B$  будут 2; 1,5; 2; 1,5.

159. Дано простое число. Составить функцию, которая будет выводить следующее за ним простое число.

160. Составить функцию для нахождения наименьшего натурального делителя  $k$  ( $k \neq 1$ ) любого заданного натурального числа  $n$ .

## Б

161. Дано натуральное число  $N$ . Составить программу формирования массива, элементами которого являются цифры числа  $N$ .

162. Составить программу, определяющую в каком из заданных двух чисел больше цифр.

163. Заменить данное натуральное число на число, которое получается из исходного записью его цифр в обратном порядке. Например, дано число 156, нужно получить 651.

164. Даны натуральные числа  $K$  и  $N$ . Составить программу формирования массива  $A$ , элементами которого являются числа, сумма цифр которых равна  $K$  и которые не больше  $N$ .

165. Даны три квадратных матрицы А, В, С n-го порядка. Вывести на печать ту из них, норма которой наименьшая. Нормой матрицы считать максимум из абсолютных величин ее элементов.

166. Два натуральных числа называются «дружественными», если каждое из них равно сумме всех делителей (кроме его самого) другого (например, числа 220 и 284). Найти все пары «дружественных» чисел, которые не больше заданного числа N.

167. Два простых числа называются «близнецами», если они отличаются друг от друга на 2 (например, 41 и 43). Напечатать все пары «близнецов» из отрезка  $[n, 2n]$ , где  $n$  – заданное натуральное число больше 2.

168. Написать программу вычисления суммы 
$$\frac{p}{q} = 1 - \frac{1}{2} + \frac{1}{3} - \dots + \frac{(-1)^{n+1}}{n}$$

для заданного числа  $n$ . Дробь  $p/q$  должна быть несократимой ( $p, q$  – натуральные).

169. Написать программу вычисления суммы  $1 + 1/2 + 1/3 + \dots + 1/n$  для заданного числа  $n$ . Результат представить в виде несократимой дроби –  $p/q$  ( $p, q$  – натуральные).

170. Натуральное число, в записи которого  $n$  цифр, называется числом Амстронга, если сумма его цифр, возведенная в степень  $n$ , равна самому числу. Найти все эти числа от 1 до  $k$ .

171. Написать программу, которая находит и выводит на печать все четырехзначные числа вида  $\overline{abcd}$ , для которых выполняется:  $a, b, c, d$  – разные цифры;  $\overline{ab} - \overline{cd} = a + b + c + d$ .

172. Найти все простые натуральные числа, не превосходящие  $n$ , двоичная запись которых представляет собой палиндром, т.е. читается одинаково слева направо и справа налево.

173. Найти все натуральные  $n$ -значные числа, цифры в которых образуют строго возрастающую последовательность (например, 1234, 5789).

174. Найти, все натуральные числа, не превосходящие заданного  $n$ , которые делятся на каждую из своих цифр.

175. Составить программу для нахождения чисел из интервала  $[M; N]$ , имеющих наибольшее количество делителей.

176. Для последовательности  $a_1 = 1, a_{n+1} = a_n + \frac{1}{1 + a_n}$  составить

программу печати  $k$ -го члена в виде обыкновенной несократимой дроби. Например,  $a_2 = 3/2, a_3 = 19/10$ .

177. Дано натуральное число  $n$ . Выяснить, можно ли представить  $n$  в виде произведения трех последовательных натуральных чисел.

178. На части катушки с автобусными билетами номера шестизначные. Составить программу, определяющую количество счастливых билетов на катушке, если меньший номер билета –  $N$ , больший –  $M$  (билет является счастливым, если сумма первых трех цифр его номера равна сумме последних трех цифр).

179. Написать программу, определяющую сумму  $n$ -значных чисел, содержащих только нечетные цифры. Определить также, сколько четных цифр в найденной сумме.

180. Из заданного числа вычли сумму его цифр. Из результата вновь вычли сумму его цифр и т.д. Через сколько таких действий получится ноль?

181. Составить программу разложения данного натурального числа на простые множители. Например,  $200 = 2^3 \times 5^2$ .

182. Дано натуральное число  $n$ . Найти все меньшие  $n$  числа Мерсена. Простое число называется числом Мерсена, если оно может быть представлено в виде  $2^p - 1$ , где  $p$  – тоже простое число. Например,  $31 = 2^5 - 1$  – число Мерсена.

183. Дано четное число  $n > 2$ . Проверить для него гипотезу Гольдбаха: каждое четное  $n$  представляется в виде суммы двух простых чисел.

## В

184. Реализовать набор подпрограмм для выполнения следующих операций над обыкновенными дробями вида  $p/q$  ( $p$  – целое,  $q$  – натуральное): а) сложение; б) вычитание; в) умножение; г) деление; д) сокращение дроби; е) возведение дроби в степень  $n$  ( $n$  – натуральное); ж) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше).

1) Дан массив  $A$  – массив обыкновенных дробей. Найти сумму всех дробей и вычислить их среднее арифметическое, результаты представить в виде несократимых дробей.

2) Дан массив  $A$  – массив обыкновенных дробей. Отсортировать его в порядке возрастания.

185. Реализовать набор подпрограмм для выполнения следующих операций над векторами: а) сложение; б) вычитание; в) скалярное умножение векторов; г) умножение вектора на число; д) нахождение длины вектора.

1) Дан массив  $A$  – массив векторов. Отсортировать его в порядке убывания длин векторов.

2) С помощью датчика случайных чисел сгенерировать  $2N$  целых чисел.  $N$  пар этих чисел задают  $N$  точек координатной плоскости. Вывести номера тройки точек, которые являются координатами вершин треугольника с наибольшим углом.

186. Реализовать набор подпрограмм для выполнения следующих операций над натуральными числами в  $P$ -ичной системе счисления ( $2 \leq P \leq 9$ ): а) сложение, вычитание, умножение, деление; б) перевод из десятичной системы счисления в  $P$ -ичную; в) перевод из  $P$ -ичной системы счисления в десятичную; г) функция проверки правильности записи числа в  $P$ -ичной системе счисления; д) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше).

1) Возвести число в степень (основание и показатель степени записаны в  $P$ -ичной системе счисления). Ответ выдать в  $P$ -ичной и десятичной системах счисления.

2) Дан массив А – массив чисел, записанных в Р-ичной системе счисления. Отсортировать его в порядке убывания. Ответ выдать в Р-ичной и десятичной системах счисления.

187. Реализовать набор подпрограмм для выполнения следующих операций над натуральными числами в шестнадцатеричной системе счисления: а) сложение; б) вычитание; в) умножение; г) деление; д) перевод из двоичной системы счисления в шестнадцатеричную; е) перевод из шестнадцатеричной системы счисления в десятичную; ж) функция проверки правильности записи числа в шестнадцатеричной системе счисления; з) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше).

1) Возвести число в степень (основание и показатель степени записаны в шестнадцатеричной системе счисления). Ответ выдать в шестнадцатеричной и десятичной системах счисления.

2) Дан массив А – массив чисел, записанных в шестнадцатеричной системе счисления. Отсортировать его в порядке убывания. Ответ выдать в шестнадцатеричной и десятичной системах счисления.

## 2.2. Рекурсивные подпрограммы

188. Найдите сумму цифр заданного натурального числа.

189. Подсчитать количество цифр в заданном натуральном числе.

190. Описать функцию  $C(m, n)$ , где  $0 \leq m \leq n$ , для вычисления биномиального коэффициента  $C_n^m$  по следующей формуле:

$$C_n^0 = C_n^n = 1; \quad C_n^m = C_{n-1}^m + C_{n-1}^{m-1} \quad \text{при } 0 < m < n.$$

191. Описать рекурсивную логическую функцию  $\text{Simm}(S, i, j)$ , проверяющую, является ли симметричной часть строки S, начинающаяся i-м и заканчивающаяся j-м ее элементами.

192. Составить программу вычисления НОД двух натуральных чисел.

193. Составить программу нахождения числа, которое образуется из данного натурального числа при записи его цифр в обратном порядке. Например, для числа 1234 получаем ответ 4321.

194. Составить программу перевода данного натурального числа в Р-ичную систему счисления ( $2 \leq P \leq 9$ ).

195. Дана символьная строка, представляющая собой запись натурального числа в Р-ичной системе счисления ( $2 \leq P \leq 9$ ). Составить программу перевода этого числа в десятичную систему счисления.

196. Составить программу вычисления суммы:  $1! + 2! + 3! + \dots + n!$  ( $n \leq 15$ ). Тип результата значения функции – Long.

197. Составить программу вычисления суммы:  $2! + 4! + 6! + \dots + n!$  ( $n \leq 16, n - \text{четное}$ ). Тип результата значения функции – Long.

## 3. Обработка строк

А

198. Дана строка, заканчивающаяся точкой. Подсчитать, сколько в ней слов.

199. Дана строка, содержащая английский текст. Найти количество слов, начинающихся с буквы b.
200. Дана строка. Подсчитать в ней количество вхождений букв g, k, t.
201. Дана строка. Определить, сколько в ней символов \*, ;, :.
202. Дана строка, содержащая текст. Найти длины самого короткого и самого длинного слов.
203. Дана строка символов, среди которых есть двоеточие (:). Определить, сколько символов ему предшествует.
204. Дана строка, содержащая текст, заканчивающийся точкой. Вывести на экран слова, содержащие три буквы.
205. Дана строка. Преобразовать ее, удалив каждый символ \* и повторив каждый символ, отличный от \*.
206. Дана строка. Определить, сколько раз входит в нее группа букв abc.
207. Дана строка. Подсчитать количество букв k в последнем ее слове.
208. Дана строка. Подсчитать, сколько различных символов встречаются в ней. Вывести их на экран.
209. Дана строка. Подсчитать самую длинную последовательность подряд идущих букв a.
210. Дана строка символов, среди которых есть одна открывающаяся и одна закрывающаяся скобка. Вывести на экран все символы, расположенные внутри этих скобок.
211. Имеется строка, содержащая буквы латинского алфавита и цифры. Вывести на экран длину наибольшей последовательности цифр, идущих подряд.
212. Дан набор слов, разделенных точкой с запятой (;). Набор заканчивается двоеточием (:). Определить, сколько в нем слов, заканчивающихся буквой a.
213. Дана строка. Указать те слова, которые содержат хотя бы одну букву k.
214. Дана строка. Найти в ней те слова, которые начинаются и оканчиваются одной и той же буквой.
215. В строке заменить все двоеточия (:) точкой с запятой (;). Подсчитать количество таких замен.
216. В строке удалить символ двоеточие (:) и подсчитать количество удаленных символов.
217. В строке между словами вставить вместо пробела запятую и пробел.
218. Удалить часть символьной строки, заключенной в скобки (вместе со скобками).
219. Определить, сколько раз в строке встречается заданное слово.
220. В строке имеется одна точка с запятой (;). Подсчитать количество символов до точки с запятой и после нее.
221. Дана строка из n символов. Преобразовать ее, заменив точками все двоеточия (:), встречающиеся среди первых  $n/2$  символов, и все восклицательные знаки, встречающиеся среди символов, стоящих после  $n/2$  символов.



222. Строка содержит одно слово. Проверить, будет ли оно читаться одинаково справа налево и слева направо (т.е. является ли оно палиндромом).
223. В записке слова зашифрованы – каждое из них записано наоборот. Расшифровать сообщение.
224. Проверить, одинаковое ли число открывающихся и закрывающихся скобок в данной строке.
225. Строка, содержащая произвольный русский текст, состоит не более чем из 200 символов. Написать, какие буквы и сколько раз встречаются в этом тексте. Ответ должен приводиться в грамматически правильной форме, например: а – 25 раз, к – 3 раза и т.д.
226. Упорядочить данный массив английских слов по алфавиту.
227. Даны две строки А и В. Составьте программу, проверяющую, можно ли из букв, входящих в А, составить В (буквы можно использовать не более одного раза и можно переставлять). Например, А: ИНТЕГРАЛ; В: АГЕНТ – составить можно; В: ГРАФ – нельзя.
228. Строка содержит произвольный русский текст. Проверить, каких букв в нем больше: гласных или согласных.
229. Двумерный массив  $n \times m$  содержит некоторые буквы русского алфавита, расположенные в произвольном порядке. Написать программу, проверяющую, можно ли из этих букв составить данное слово S. Каждая буква массива используется не более одного раза.
230. Результаты вступительных экзаменов представлены в виде списка из N строк, в каждой строке которого записаны фамилия студента и отметки по каждому из M экзаменов. Определить количество абитуриентов, сдавших вступительные экзамены: а) только на «отлично»; б) на «хорошо» и «отлично».
231. Составить программу преобразования натуральных чисел, записанных в римской нумерации, в десятичную систему счисления.
232. Из заданной символьной строки выбрать те символы, которые встречаются в ней только один раз, в том порядке, в котором они встречаются в тексте.
233. В строковом массиве хранятся фамилии и инициалы учеников класса. Требуется напечатать список класса с указанием для каждого ученика количества его однофамильцев.
234. Дано число в двоичной системе счисления. Проверить правильность ввода этого числа (в его записи должны быть только символы 0 и 1). Если число введено неверно, повторить ввод. При правильном вводе перевести число в десятичную систему счисления.
235. Дана строка символов. В каждой подстроке, заключенной в квадратные скобки поменять местами только цифры так, чтобы они стали упорядоченными по возрастанию. Остальные символы оставить на своих местах.
236. Дана строка символов. Проверить, является ли эта строка синтаксически правильной записью целого числа (десятичного или 16-ричного).
237. Дана строка символов. Проверить является ли эта запись синтаксически правильной записью вещественного числа.

## Б

238. Дана строка, содержащая текст, записанный строчными русскими буквами. Получить в другой строке тот же текст, записанный заглавными буквами.

239. Дана строка, содержащая произвольный текст. Выяснить чего в нем больше: русских букв или цифр.

240. Дана строка, содержащая текст на русском языке. Выяснить, входит ли данное слово в указанный текст, и если да, то сколько раз.

241. Дана строка, содержащая текст на русском языке. В предложениях некоторые из слов записаны подряд несколько раз (предложение заканчивается точкой или восклицательным знаком). Получить в новой строке отредактированный текст, в котором удалены подряд идущие вхождения слов в предложениях.

242. Дана строка, содержащая текст, набранный заглавными русскими буквами. Провести частотный анализ текста, т.е. указать (в процентах), сколько раз встречается та или иная буква.

243. Дана строка, содержащая текст на русском языке. Определить, сколько раз встречается в ней самое длинное слово.

244. Дана строка, содержащая произвольный текст. Проверить, правильно ли в нем расставлены круглые скобки (т.е. находится ли правее каждой открывающей скобки закрывающая, и левее закрывающей – открывающая).

245. Дана строка, содержащая текст на русском языке. Составить в алфавитном порядке список всех слов, встречающихся в этом тексте.

246. Дана строка, содержащая текст на русском языке. Определить, сколько раз встречается в нем самое короткое слово.

247. Дана строка, содержащая текст на русском языке и некоторые два слова. Определить, сколько раз они входят в текст и сколько раз они входят непосредственно друг за другом.

248. Дана строка, содержащая текст на русском языке. Выбрать из него только те символы, которые встречаются в нем только один раз, в том порядке, в котором они встречаются в тексте.

249. Дана строка, содержащая текст и арифметические выражения вида:  $a + b$ ,  $a - b$ ,  $a * b$  и  $a / b$ . Выписать из нее все арифметические выражения и вычислить их значения.

250. Дана строка, содержащая текст на русском языке и некоторая буква. Найти слово, содержащее наибольшее количество указанных букв.

251. Дана строка, содержащая текст на русском языке и некоторая буква. Подсчитать, сколько слов начинается с указанной буквы.

252. Дана строка, содержащая текст на русском языке. Найти слово, встречающееся в каждом предложении, или сообщить, что такого слова нет.

253. Дана строка, содержащая текст, включающий русские и английские слова. Подсчитать, каких букв в тексте больше – русских или латинских.

254. Дана строка, содержащая произвольный текст. Сколько слов в тексте? Сколько цифр в тексте?

255. Дана строка, содержащая текст, включающий русские и английские слова. Получить новую строку, заменив в исходной строке все заглавные буквы строчными и наоборот.

256. Дана строка, содержащая зашифрованный русский текст. Каждая буква заменяется следующей за ней в алфавите буквой (буква я заменяется буквой а). Получить в новом файле расшифровку данного текста.

257. Даны две строки  $f_1$  и  $f_2$ . Строка  $f_1$  содержит произвольный текст. Слова в тексте разделены пробелами и знаками препинания. Строка  $f_2$  содержит не более 30 слов, которые разделены запятыми. Эти слова образуют пары: каждое второе является синонимом первого. Заменить в строке  $f_1$  те слова, которые можно, их синонимами. Результат поместить в новую строку.

258. Дана строка. Удалить из нее все лишние пробелы, оставив между словами не более одного. Если строка начиналась и/или заканчивалась пробелами, то их тоже нужно удалить. Результат поместить в новую строку.

259. Дана строка и некоторое слово. Напечатать те предложения строки, которые содержат данное слово.

260. Дана строка. Напечатать в алфавитном порядке все слова из данной строки, имеющие заданную длину  $n$ .

261. Дана строка, содержащая текст на русском языке. Подсчитать количество слов, начинающихся и заканчивающихся на одну и ту же букву.

262. Дана строка символов. Проверить является ли эта строка синтаксически правильной записью арифметического выражения.

263. Дан текст, содержащий  $N$  строк. Каждая строка заканчивается точкой. Длина строки  $\leq 60$ . Выровнять строки так, чтобы каждая строка имела длину 60. Строка не должна начинаться и заканчиваться пробелами. Выравнивание строк проводить равномерно вставляя дополнительные пробелы в тех местах, где они уже имеются. Вставлять все требуемые пробелы в одном месте не допускается.

## 4. Регулярные выражения

**Внимание.** Во всех задачах этого раздела предварительно нужно подготовить исходную текстовую строку (по тематике задачи) размером не менее 1-2 страниц.

264. Из заданной текстовой строки вывести все слова длиной в  $n$  символов.

265. Из заданной текстовый строки вывести все слова длиной от  $n$  до  $m$  символов.

266. Из заданной текстовой строки вывести все слова, начинающиеся с символа  $c1$  и заканчивающиеся символом  $c2$ .

267. Из заданной текстовой строки вывести все слова, в которых первый и последний символы совпадают. Учесть однобуквенные слова.

268. Из заданной текстовой строки вывести все слова, в которых первый и последний символы разные. Учесть однобуквенные слова.

269. Из заданной текстовой строки вывести все слова, в которых первый и последний символы гласные/согласные. Учесть однобуквенные слова.
270. Из заданной текстовой строки вывести все слова, содержащие пары подряд идущих одинаковых букв.
271. Из заданной текстовой строки вывести все сложные слова, пишущиеся через дефис.
272. В заданной текстовой строке найдите и удалите все повторяющиеся (подряд) слова и словосочетания. Вывести полученный текст и отдельно удаленные слова и словосочетания.
273. Из заданной текстовой строки, содержащей список литературы (выполненный по стандарту библиографической записи), вывести для каждой записи авторов, город, издательство, год издания и объем страниц.
274. Из заданной текстовой строки, включающей кроме обычного тематического текста фамилии и инициалы людей (Иванов И.И.) с датами их рождения (01.01.2000), вывести все фамилии и инициалы людей с датами их рождения.
275. Из заданной текстовой строки, включающей кроме обычного тематического текста химические формулы веществ, вывести все химические формулы указанных веществ.
276. Из заданной текстовой строки вывести все числа (не путать с цифрами). Числа могут быть целыми или записаны в формате с фиксированной запятой (т.е. в научном формате), а также содержать разделители групп разрядов. Лишние нули слева или справа удалять.
277. Из заданной текстовой строки вывести все числа (не путать с цифрами). Числа могут быть целыми или записаны в формате с плавающей запятой, а также содержать разделители групп разрядов. Лишние нули слева или справа удалять.
278. В заданной текстовой строке найдите все числа с разделителями групп разрядов и удалите эти разделители. Предусмотрите и обратную процедуру.
279. В заданной текстовой строке, содержащей арифметические выражения (каждое выражение – в новой строке), выполнить проверку их корректности (знаки операций и расстановку скобок), а также посчитать результат. Вывести выражения и их результаты. Если выражение некорректно, то вместо результата указать характер ошибки.
280. Из заданной текстовой строки вывести все предложения, содержащие значения времени (в различных форматах, включая национальные настройки). Предложения с некорректными значениями времени не выводить.
281. Из заданной текстовой строки вывести все предложения, содержащие значения даты (в различных форматах, включая национальные настройки). Предложения с некорректными значениями даты не выводить.
282. В заданной текстовой строке замените все даты, представленные в американском формате, на привычный нам российский формат, и наоборот. Некорректные даты не выводить.

283. Из заданной текстовой строки вывести все телефонные номера, заданные в международном формате. Считать, что код страны и города может содержать от трёх до пяти цифр, а городской номер – от 7 до 5 цифр соответственно. Некорректные телефонные номера не выводить.

284. Из заданной текстовой строки вывести все телефонные номера выбранного сотового оператора.

285. Из заданной текстовой строки вывести все российские автомобильные номера (с кодом региона). Некорректные автомобильные номера не выводить.

286. Из заданной текстовой строки вывести все номера банковских карт. Для каждой карты укажите платежную систему, к которой она относится, и наименование выпустившего ее банка.

287. Из заданной текстовой строки вывести все паспортные данные (серия и номер паспорта, дата его выдачи) указанных в тексте людей.

288. Из заданной текстовой строки вывести все IPv4-адреса (IPv6) в точно-десятичной нотации. Некорректные IP-адреса не выводить.

289. Из заданной текстовой строки вывести все имена указанных там файлов. Использовать полные имена файлов.

290. Из заданной текстовой строки, включающей полные имена файлов, вывести имена указанных там файлов (только имя и расширение), содержащихся в задаваемой папке (на некотором диске).

291. Из заданной текстовой строки вывести все гиперссылки. Некорректные гиперссылки при выводе обозначить.

292. Из заданной текстовой строки вывести все адреса электронной почты. Некорректные адреса при выводе обозначить.

293. Из заданной текстовой строки вывести все значения HTML-цвета, заданного как **#ABCDEF**, то есть # и затем 6 шестнадцатеричных символов.

294. Из заданной текстовой строки (например, HTML-документа) вывести все URL, ведущие на картинки (файлы с расширениями **png, jpg, bmp, svg**). Это тэги вида:

```
</img>
```

```

```

или с дополнительными атрибутами (они необязательные), например:

```
.
```

295. Из заданной текстовой строки вывести все смайлики и их количество. Смайликом будем считать последовательность символов, удовлетворяющую условиям:

- первым символом является либо ; (точка с запятой), либо : (двоеточие) ровно один раз;

- далее может идти символ - (минус) сколько угодно раз (в том числе символ минус может идти ноль раз, т.е. отсутствовать);

- в конце обязательно идет некоторое количество (не меньше одной) одинаковых скобок из следующего набора: (, ), [, ];

- внутри смайлика не может встречаться никаких других символов.

## **5. Работа со списками**

296. Информация о студентах, зачисленных на данную специальность, включает в себя:

- фамилию;
- имя;
- отчество;
- год и месяц рождения;
- домашний адрес;
- изучаемый иностранный язык.

Сформировать списки студенческих групп с учетом следующих требований:

- количество групп – 2;
- количество студентов в группе, не более 10;
- количество студентов, изучающих разные иностранные языки должно быть примерно одинаково в каждой группе.

Списки групп должны быть упорядочены по алфавиту и размещены в текстовом файле.

297. Составить программу для начисления стипендии студентам по результатам экзаменационной сессии. Информация о результатах сессии включает в себя:

- фамилию;
- имя;
- отчество;
- номер группы;
- экзаменационные оценки.

Количество экзаменационных оценок не менее 3 и не более 5. Стипендия начисляется студентам, сдавшим все экзамены в сессию, по следующим правилам. Студенты, сдавшие все экзамены на «отлично» получают надбавку равную 100%; студенты, сдавшие экзамены на «хорошо» и «отлично» – 50%; а студенты, сдавшие экзамены на «хорошо», – 25%. Стипендия не начисляется студентам, имеющим в сессию более двух удовлетворительных оценок.

Исходный список студентов каждой группы разместить в отдельном текстовом файле. Список студентов каждой группы, получивших стипендию, вывести на экран, упорядочив его по алфавиту.

298. Составить программу для определения проходного балла на заданную специальность по результатам вступительных экзаменов и формирования списка студентов, зачисленных на эту специальность. Информация о студентах включает в себя:

- фамилию, имя, отчество;
- оценки, на вступительных экзаменах (не более четырех);
- дополнительный балл за участие в олимпиадах.

Список студентов должен быть упорядочен по алфавиту и размещен в текстовом файле.

## **Приложение Б.**

### **Содержание отчета по выполненному заданию**

Отчет по выполненным заданиям оформляется в текстовом редакторе MS Word в соответствии с ГОСТ Р 2.105-2019, он должен включать:

**Титульный лист** (оформленный по правилам, принятым в Университете).

**Оглавление** (сформированное автоматически средствами MS Word).

**1. Постановка задачи** (расширенная), включая:

- требования к элементам пользовательского интерфейса, таблица используемых элементов управления с установленными значениями свойств;
- требования к организации ввода-вывода данных в программе, включая работу с файлами (если она предусматривается);
- требования к реакции программы на нестандартные, в том числе критические, действия пользователя;
- прочие требования к программе.

**2. Схемы алгоритмов** (выполненные аккуратно, в строгом соответствии с ГОСТ 19.701-90, средствами MS Word, MS Visio или др.).

**3. Текст программы** (с использованием структурного оформления кода, правильного переноса операторов и комментирования).

**4. Результаты тестирования программы:**

- таблица с набором тестовых исходных данных, вводимых разными способами, и соответствующих результатов;
- содержание файла(ов) с исходными данными (если они используются);
- несколько наиболее интересных скриншотов работы программы;
- содержание файла результатов (если он формируется).

**Выводы.**

**Примечания:** 1. В отчете по объектно-ориентированному программированию дополнительно нужно добавить пункт «**Логическая структура программы**», содержащий: используемые объекты и их члены (поля, свойства, методы, события), их назначение и выполняемые действия, взаимосвязи и взаимодействие составных частей. Описание логической структуры программы выполняется с учетом схем алгоритмов и текста программы, а также необходимых скриншотов работы программы.

2. В отчете по приложению для работы с базой данных дополнительно нужно добавить пункт «**Схема базы данных**», содержащий собственно схему базы данных системы и структуры ее таблиц с указанием ключевых полей.

# Оглавление

Задание на занятие №1. Изучение интегрированной среды разработки Microsoft Visual Studio .....	3
Контрольные вопросы по теме: «Интегрированная среда разработки Microsoft Visual Studio» .....	8
Задание на занятие №2. Разработка линейной программы «Простой калькулятор» .....	9
Контрольные вопросы по теме: «Основные элементы программирования» .....	14
Задание на занятие №3. Разработка программы с разветвлениями «Решение квадратного уравнения» .....	15
Контрольные вопросы по теме: «Строки» .....	16
Контрольные вопросы по теме: «Регулярные выражения» .....	16
Контрольные вопросы по теме: «Ввод-вывод» .....	17
Контрольные вопросы по теме: «Операторы управления и циклы» .....	17
Задание на занятие №4. Разработка программы с разветвлениями «АРМ оператора обменного пункта» .....	19
Контрольные вопросы по теме: «Операторы управления и циклы» .....	20
Задание на занятие №5. Разработка простой циклической программы «Расчет значения $\exp(x)$ » .....	22
Контрольные вопросы по теме: «Операторы управления и циклы» .....	24
Задание на занятие №6. Разработка программы обработки массива .....	26
Контрольные вопросы по теме: «Массивы» .....	28
Контрольные вопросы по теме: «Подпрограммы как методы» .....	28
Контрольные вопросы по теме: «Указатели» .....	29
Контрольные вопросы по теме: «Структуры» .....	29
Контрольные вопросы по теме: «Перечисления» .....	29
Контрольные вопросы по теме: «Организация работы с файлами» .....	29
Контрольные вопросы по теме: «Разработка пользовательского интерфейса» .....	30
Задание на занятие №7. Разработка классов и использование свойств, методов и событий, организация событийного управления .....	31
Контрольные вопросы по теме: «Объектно-ориентированное программирование» .....	36
Задание на занятие №8. Работа с базами данных и организация взаимодействия с Microsoft Office .....	37
Справочные материалы по языку C++ в среде Visual Studio .....	42
Приложение А. Перечень типовых задач .....	56
1. Массивы .....	56
1.1. Одномерные массивы .....	56
1.2. Сортировка массивов .....	61
1.3. Двумерные массивы .....	62



2. Подпрограммы.....	67
2.1. Нерекурсивные подпрограммы .....	67
2.2. Рекурсивные подпрограммы.....	71
3. Обработка строк .....	71
4. Регулярные выражения .....	75
5. Работа со списками.....	78
Приложение Б. Содержание отчета по выполненному заданию.....	79



Волков А.И.

Программирование (Visual C++). Практикум по освоению методов и приемов разработки программ. 2-е изд. – М.: МТУСИ, 2020.– 82 с.

Подписано в печать \_\_. \_\_.20. Формат \_\_×\_\_ / \_\_. Бумага офсетная №2.  
Ризография. Усл.-печ.л. 5,13. Уч.-изд.л. 4,56. Изд. №\_\_. Тираж 100 экз.  
Заказ \_\_\_\_ . Бесплатно.

Московский технический университет связи и информатики.  
111024, Москва, ул. Авиамоторная, 8а.

