

Министерство образования и науки  
Российской Федерации

Белгородский государственный технологический университет  
им. В.Г. Шухова  
Кафедра информационных технологий

Утверждено  
научно-методическим советом  
университета

## **ИНФОРМАТИКА**

Методические указания к выполнению  
лабораторных работ для студентов заочной формы обучения  
направлений бакалавриата  
09.03.02 - Информационные системы и технологии и

Белгород  
2021

УДК 007(07)  
ББК 32.81я7  
И74

Составители: ст. преп. С.Н. Рога  
ст. преп. А.Г. Смышляев  
ст. преп. А.В. Четвериков

Рецензент

**Информатика:** методические указания к выполнению

И74 лабораторных работ для студентов заочной формы обучения направлений бакалавриата 09.03.02 - Информационные системы и технологии и / сост: С.Н. Рога, А.Г. Смышляев, А.В. Четвериков. – Белгород: Изд-во БГТУ, 2021. – 55 с.

Методические указания составлены в соответствии с учебным планом и рабочей программой, предназначены для приобретения студентами базовых навыков в работе с персональным компьютером, алгоритмизации и программирования на языке C/C++ и содержат теоретический материал и задания к выполнению десяти лабораторных работ.

Методические указания предназначены для студентов заочной формы обучения направлений бакалавриата 09.03.02 - Информационные системы и технологии.

Данное издание публикуется в авторской редакции.

УДК 007(07)  
ББК 32.81я7

© Белгородский государственный  
технологический университет  
(БГТУ) им. В.Г. Шухова, 2021

**СОДЕРЖАНИЕ**

<b>СОДЕРЖАНИЕ .....</b>	<b>3</b>
<b>ПРАВИЛА ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ .....</b>	<b>4</b>
<b>ЛАБОРАТОРНАЯ РАБОТА №1 .....</b>	<b>5</b>
<b>ЛАБОРАТОРНАЯ РАБОТА № 2 .....</b>	<b>24</b>
<b>ЛАБОРАТОРНАЯ РАБОТА № 3 .....</b>	<b>29</b>
<b>БИБЛИОГРАФИЧЕСКИЙ СПИСОК.....</b>	<b>41</b>

## **ПРАВИЛА ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ**

По курсу информатики предусмотрено выполнение ряда лабораторных работ. Студент обязан перед выполнением каждой лабораторной работы самостоятельно ознакомиться с теоретическим материалом и по ее результатам предоставить отчет. Все отчеты о выполнении лабораторных работ оформляются в отдельной тетради. Допускается оформлять отчеты в печатном виде на листах формата А4.

Отчет к лабораторным работам должен содержать:

1. Заголовок лабораторной работы – номер работы, данные о студенте, слова «Выполнение» и «Защита», название и цель работы.
2. Содержание работы и индивидуальные задания.
3. Блок-схемы разработанных алгоритмов (при оформлении отчета в печатном виде рекомендуется использовать Microsoft Visio).
4. Тексты программ на языке C/C++.
5. Результаты тестирования программ.
6. Вывод о выполненной работе.

## ЛАБОРАТОРНАЯ РАБОТА №1

### РАБОТА В СРЕДЕ MICROSOFT VISUAL STUDIO 2010. РЕАЛИЗАЦИЯ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ СРЕДСТВАМИ ЯЗЫКА C/C++

*Цель работы:* получить навыки в создании, настройке и отладке консольных приложений на языке программирования C/C++ в среде Visual Studio; ознакомиться с основными библиотечными функциями ввода-вывода; получить навыки в составлении простейших циклических алгоритмов и реализации их средствами языка C/C++.

## КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

### Среда разработки программ Microsoft Visual Studio 2010

Среда Visual Studio 2010 позволяет работать с несколькими языками программирования, в т.ч. и с C/C++. Создаваемые приложения могут разрабатываться как в виде неуправляемого кода для платформы Win32, так и управляемого для платформы .Net. Далее рассмотрим процесс создания и настройки консольного приложения Win32 на языке C/C++.

### Создание решения

После запуска среды на экране появляется ее главное окно, в котором обычно отображается стартовая страница, как на рис. 1.1.

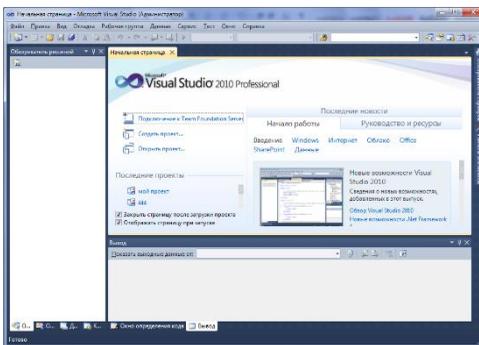


Рис. 1.1. Окно среды разработки программ Visual Studio 2010

Окно содержит заголовок, меню, панель инструментов, строку состояния, а также несколько дочерних окон, таких как *Обзорщик решений*, расположенный с левой стороны окна (рис. 1.1). В зависимости от настроек среды и пожеланий пользователя расположение и состав дочерних окон может отличаться от представленного на рисунке варианта. Стартовая страница, которую можно отключить в настройках среды (*Сервис | Параметры*), содержит ссылки создания и открытия проектов, открытия последних созданных проектов, а также запуска обучающих и справочных ресурсов Microsoft.

Для создания консольного приложения можно щелкнуть по ссылке *Создать проект* стартовой страницы или выполнить команду меню *Файл | Создать | Проект*. На экране появится диалоговое окно (рис. 1.2), в котором можно выбрать один из предлагаемых шаблонов для будущего приложения. В левой панели развернем узел *Visual C++*, а затем в средней части окна выберем пункт *Консольное приложение Win32*. Выбор в поле со списком версии платформы .Net Framework не имеет значения, т.к. приложение создается для платформы Win32. Далее в поле *Имя* в нижней части окна необходимо ввести имя нового проекта, в составе которого можно использовать как латиницу, так и кириллицу (например, «мой проект»). Одновременно автоматически заполняется поле *Имя решения*. В поле со списком *Расположение* можно указать каталог размещения проекта. Если установить флажок *Создать каталог для решения*, то в указанном месте будет создана новая папка с именем, совпадающим с именем решения. Нажатие *OK* закрывает это окно и запускает окно *Мастера приложений Win32*.

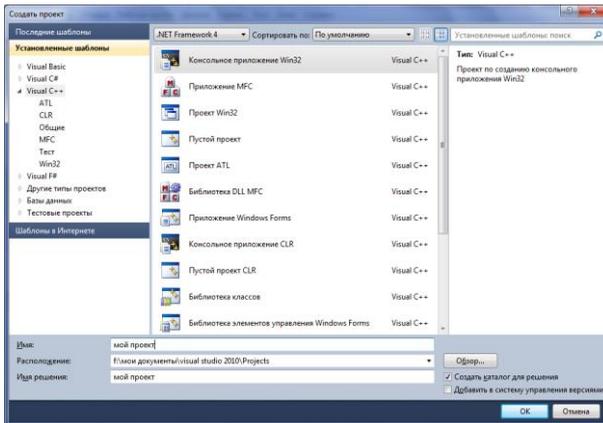


Рис. 1.2. Диалоговое окно *Создать проект*

В первом окне Мастера нажмем кнопку *Далее*. Во втором окне (рис. 1.3) устанавливаются параметры будущего приложения. Переключатель *Тип приложения* оставим в положении *Консольное приложение*. Далее снимем флажок *Предварительно скомпилированный заголовок* и установим флажок *Пустой проект*. Такие параметры означают, что приложение полностью будет создаваться самим программистом. Теперь нажмем кнопку *Готово*. Проект с именем *мой проект* создан.

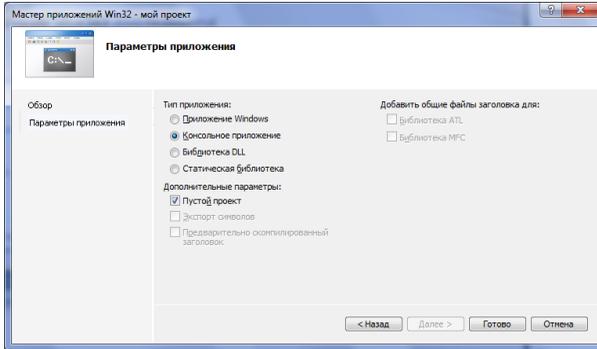


Рис. 1.3. Диалоговое окно Мастера приложений Win32

В среде Visual Studio проект всегда создается в составе *решения*, которое может содержать несколько проектов (в нашем случае один). В свою очередь проект может объединять несколько логически связанных файлов с исходным кодом, настройками конфигурации, ресурсами и т.д. Если перейти в папку, содержащую созданный проект, то мы увидим там ряд файлов:

- *мой проект.sln* – файл решения для созданной программы. Если выполнить двойной щелчок на этом файле, то произойдет запуск среды с открытием данного решения.
- *мой проект.suo* – файл настроек среды при работе с данным решением.
- *мой проект.sdf* – содержит данные, необходимые для работы компонента среды *Intellisense*.

Также в составе каталога с решением имеется подкаталог проекта, который также называется *мой проект* и содержит такие файлы:

- *мой проект.vcxproj* – файл проекта. Запуск этого файла также приведет к старту среды и открытию решения.

- *мой проект.vcxproj.filters* – файл с описанием фильтров, используемых *Обозревателем решения* для отображения разных групп файлов решения.
- *мой проект.vcxproj.user* – файл пользовательских настроек.

В дальнейшем в этом же каталоге будет размещаться файл с исходным текстом программы. Кроме того, в папке с решением будут созданы другие подкаталоги, о назначении которых будет рассказано ниже.

Состав созданного в виде пустого проекта решения отображается в *Обозревателе решения* (рис. 1.4) в виде набора папок.

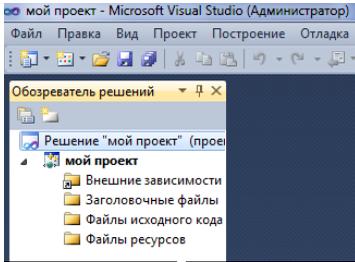


Рис. 1.4. Структура решения

Папка *Внешние зависимости* отображает файлы, не добавленные явно в проект, но использующиеся в файлах исходного кода, например включенные при помощи директивы *#include*. Остальные папки содержат соответственно имеющиеся в проекте заголовочные файлы, файлы исходного кода и ресурсов.

Созданный проект пока не содержит ни одного файла с исходным кодом. Для его добавления можно щелкнуть правой кнопкой мыши на папке *Файлы исходного кода* и выбрать команды *Добавить*, *Создать элемент* (рис. 1.5).

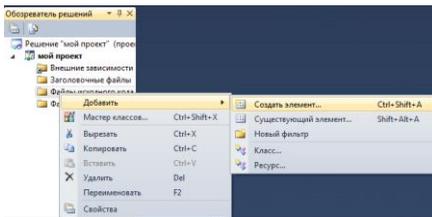


Рис. 1.5. Добавление в проект файла с исходным кодом



Если необходимо прекратить работу с текущим решением, то можно выполнить команду *Файл | Закреть решение* или открыть другое решение командой *Файл | Открыть*.

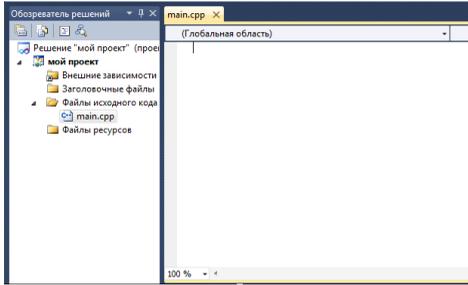


Рис. 1.7. Окно редактора кода

Теперь произведем настройку свойств созданного проекта. Для этого можно выполнить команду меню *Проект | Свойства* или нажать **Alt+F7**. В левой части диалогового окна (рис. 1.8) раскроем узел *Свойства конфигурации*. Конфигурации проекта определяют параметры компоновки приложения и свойства устанавливаются для каждой из них отдельно. Изначально каждый проект в решении Visual Studio имеет две конфигурации — *Debug* (Отладка) и *Release* (Выпуск). Каждая из них создается (автоматически) в отдельном каталоге. При использовании конфигурации *Debug* будет создаваться отладочная версия проекта, с помощью которой можно осуществлять отладку на уровне исходного кода. Конфигурация *Release* предназначена для окончательной сборки приложения. В ней отсутствует отладочная информация и при создании выходного файла (с расширением *.exe*) компиляция происходит с включенным режимом оптимизации кода, что уменьшает его объем (по сравнению с конфигурацией *Debug*). Для примера настроим некоторые свойства активной конфигурации *Debug*.

После раскрытия узла *Свойства конфигурации* выберем пункт *Общие*. Установим свойство *Набор символов* в значение *Использовать многобайтовую кодировку*. Данное свойство означает, что по умолчанию в программе будут использоваться строки и строковые константы в однобайтовой кодировке ANSI и соответствующие функции их обработки.

У проекта и его конфигураций имеется много других свойств. Например, раскрыв узел *C/C++* и выбрав пункт *Дополнительно*, можно установить значение свойства *Компилировать как*. Это свойство позволяет принудительно установить параметры компиляции независимо от расширения файла исходного кода. Свойство *Командная строка* позволяет увидеть с какими параметрами будет запускаться компилятор среды (*cl.exe*). Однако мы оставим эти свойства без изменения и нажмем *ОК*.

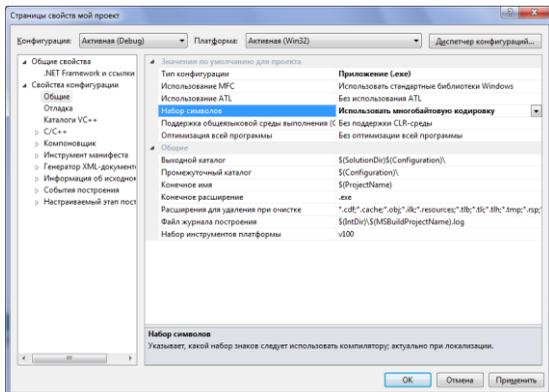


Рис. 1.8. Диалоговое окно определения свойств проекта

## Редактор кода

Среда Visual Studio включает в себя удобный редактор для набора исходного текста программы. Перечислим некоторые полезные возможности редактора кода:

- подсветка служебных слов языка и комментариев;
- разбиение кода на логические группы. Редактор автоматически объединяет фрагменты кода в группы, которые можно свернуть или развернуть воспользовавшись значками с символами «-» и «+» слева от кода (рис. 1.9). Примером таких логических групп кода может быть блок комментариев или составной оператор тела функции;
- автоматическое создание отступов в коде. Для улучшения читабельности текста программы редактор автоматически добавляет отступы при вводе содержимого составного и некоторых других операторов. Также автоматически



По-другому отобразить определение элемента можно, если установить на него курсор и нажать комбинацию клавиш **Ctrl+F12**. Если определение находится в этом же файле, то оно будет выделено. Если в другом, то он будет открыт в отдельной вкладке редактора кода и название элемента будет также выделено.

Технология автодополнения кода *IntelliSense* позволяет программисту не вводить полностью название функции или заголовочного файла, а выбрать их из списка, сформированного по первым введенным буквам. Например, если при наборе директивы препроцессора ввести `#include <`, то раскроется список всех доступных заголовочных файлов (рис. 1.9). Если вводить первые символы имени, то курсор начнет перемещаться по списку, сужая круг возможных вариантов. Для ввода выбранного варианта следует нажать **Tab** или **Enter**.

Для того, чтобы полностью не вводить название какой-либо функции или символической константы можно ввести первые несколько символов имени и нажать **Ctrl+Пробел**. Будет показан список возможных вариантов. После выбора одного из них и ввода открывающей круглой скобки появится всплывающая подсказка о параметрах функции.

## Средства ввода-вывода данных

Прежде, чем рассмотреть примеры простейших программ, вкратце остановимся на тех средствах ввода-вывода, которые можно использовать в консольных приложениях.

Языки C и C++ лишены встроенных средств ввода-вывода, который осуществляется с помощью библиотечных функций. Языку C++ в наследство от языка C досталась стандартная библиотека ввода-вывода, прототипы функций которой содержатся в заголовочном файле *stdio.h*. Для того чтобы использовать эти функции, в начале программы необходимо разместить директиву препроцессора `#include <stdio.h>`.

Вывести информацию в стандартный поток вывода (на экран) можно с помощью функции форматированного вывода *printf*, которая может иметь переменное число параметров. Первым параметром является управляющая строка, которая содержит компоненты трех типов: обычные символы, которые просто копируются в стандартный выходной поток; спецификации преобразования, каждая из которых вызывает вывод на экран очередного аргумента из последующего списка; управляющие символьные константы. После управляющей строки могут размещаться через запятую аргументы, представляющие собой какие-

либо выражения. Например, для вывода значения целочисленной переменной можно использовать следующий фрагмент:

```
int a=5;
printf("a= %d\n",a);
```

В результате такого вызова функции первые три символа управляющей строки (включая и пробел) без изменений выводятся на экран. Затем вместо спецификации преобразования *%d* выводится значение переменной *a*. Далее в выходной поток выводится управляющий символ *'\n'*, что обеспечивает перевод курсора в начало следующей строки.

Для ввода данных можно использовать функцию *scanf*, которая имеет сходный набор параметров. В управляющей строке могут размещаться спецификации преобразования, а также и пробелы и символы табуляции. Аргументы должны быть указателями на переменные соответствующих типов. Для этого перед именем переменной записывается символ *&* (операция взятия адреса переменной). Например, для ввода целочисленной и вещественной переменных можно использовать следующий фрагмент:

```
int x;
double y;
scanf("%d%f",&x,&y);
```

Кроме стандартной библиотеки ввода-вывода языка C можно использовать инструменты стандартной библиотеки языка C++. Средства ввода-вывода для консольных приложений описаны в заголовочном файле *iostream*. Для их использования в начале текста программы размещаются следующие инструкции:

```
#include <iostream>
using namespace std;
```

В отличие от заголовочных файлов языка C, заголовочные файлы стандартной библиотеки C++ не имеют расширения. Все идентификаторы стандартной библиотеки определены в пространстве имен *std*. Такой подход позволяет использовать одни и те же названия в разных библиотеках. Для доступа к объектам необходимо перед их именами размещать квалификатор *std::* (например, *std::cout*) или указывать оператор *using*. Во втором случае ниже указанной строки можно использовать все имена без квалификаторов.

Заголовочный файл *iostream* содержит описание классов для управления вводом и выводом данных, а также определения стандартных объектов-поточков ввода с клавиатуры (*cin*) и вывода на

экран (*cout*). Также в файле определены (перегружены) для разных типов данных операции помещения в поток << и чтения из потока >>. Например, для ввода с клавиатуры значений двух целых переменных и вывода их суммы можно использовать следующий фрагмент:

```
int x,y;
cout<<"Введите два числа"<<'\\n';
cin>>x>>y;
cout<<"Сумма чисел= "<<x+y<<endl;
```

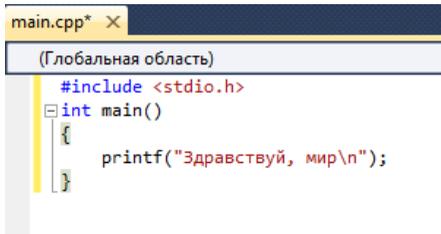
В приведенном примере для перевода строки при выводе используются два альтернативных варианта: вывод символьной константы с управляющим символом '\\n' или применение манипулятора *endl*.

### Пример составления и отладки простейшей программы

Для примера в созданном ранее файле *main.cpp* разместим программу, которая выводит на экран фразу *Здравствуй, мир* (рис. 1.11). Первая строка программы содержит директиву препроцессора, включающую в текст программы содержимое заголовочного файла *stdio.h*. Далее идет определение функции *main*, которая обязательно должна быть в программе и с которой начинается ее выполнение. Согласно стандарта C++ функция *main* должна быть типа *int* и возвращать системе значения нуля в случае успешного завершения с помощью оператора:

```
return 0;
```

Фактически же достижение при выполнении программы закрывающей фигурной скобки эквивалентно применению данного оператора, поэтому его можно не указывать.



```
main.cpp* X
(Глобальная область)
#include <stdio.h>
int main()
{
    printf("Здравствуй, мир\\n");
}
```

Рис. 1.11. Пример программы

Для того, чтобы запустить набранную программу на выполнение можно использовать команду *Отладка* | *Начать отладку* или нажать **F5**. На экране появится окно, в котором будет сказано, что проект устарел и нужно выполнить его построение (т.е. компиляцию и компоновку). После нажатия кнопки *Да* можно увидеть появление сведений о ходе построения во вкладке *Вывод* информационного окна. Если в тексте будут обнаружены синтаксические ошибки, то в это окно о них выводятся сообщения с указанием номера ошибки. Если вызвать контекстное меню сообщения и выполнить команду *Найти в коде*, то курсор переместится в место, где ошибка обнаружена.

В нашем случае ошибок нет и произойдет запуск программы в конфигурации, установленной по умолчанию (*Debug*). При этом окно консоли откроется и тут же закроется. Для того, чтобы выполнить задержку программы после вывода изменим ее следующим образом:

```
#include <stdio.h>
#include <conio.h>
int main()
{
    printf("Здравствуй, мир\n");
    getch();
}
```

В данном случае мы добавили в программу вызов функции *getch*, прототип которой определен в файле *conio.h*, где описаны функции ввода-вывода для консольного терминала. Данная функция ждет нажатия любой клавиши и возвращает ее код. После повторного запуска проекта мы увидим результаты работы функции вывода (рис. 1.12).

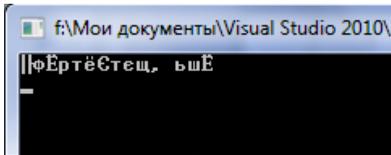


Рис. 1.12. Результат работы программы

Однако в окне консоли мы не увидим слов *Здравствуй, мир*. Это происходит потому, что текст программы был набран в кодировке win-1251 (кодировка CP1251), а по умолчанию в окне консоли используется кодировка CP866. Для ее изменения используем функцию *setlocale*, которая изменяет так называемую схему локализации

или *локаль*. Локаль определяет кроме кодировки символы валюты, систему мер и другие параметры. Прототип функции определен в файле *locale.h*. У нее два параметра. Первый определяет локализуемую категорию. В данном случае это кодировка символов, поэтому укажем константу LC\_STYPE (можно также LC\_ALL – все категории). В качестве второго параметра должна быть указана строка с названием локали. В нашем случае это *Russian* или *rus*. Однако поскольку программа будет выполняться под управлением русифицированной операционной системы, то можно записать в качестве названия пустую строку. Это означает, что будет применена локаль, используемая по умолчанию в данной системе. Таким образом, окончательный вариант программы выглядит так:

```
#include <stdio.h>
#include <conio.h>
#include <locale.h>
int main()
{
    setlocale(LC_STYPE, "");
    printf("Здравствуй, мир\n");
    getch();
}
```

Теперь рассмотрим такое средство отладки программ, как пошаговое выполнение. В Visual Studio имеется два варианта пошагового выполнения программы: с заходом внутрь функций (**F11**) и с обходом (**F10**). Чтобы не заходить внутрь библиотечных функций, нажмем **F10**. Далее при необходимости производится построение и слева от первого оператора появляется желтая стрелка, показывающая текущий оператор. Далее нажатием клавиши **F10** выполняем строки программы. При наличии в программе переменных их значения можно отслеживать во вкладках информационного окна *Видимые, Локальные, Контрольные значения 1*.

После выполнения последней строки в редакторе в отдельной вкладке открывается файл *crtexe.c*, в который передается управление. Данный файл содержит функцию *mainCRTStartup*, которая запускается перед функцией *main* и предназначена для инициализации библиотеки времени выполнения языка C – *C Run-Time Library (CRT)*. После выхода из функции *main* данная функция также должна завершить свою работу. Для того чтобы не выполнять ее операторы по отдельности, можно нажать комбинацию клавиш **Shift+F11** (шаг с выходом). Другим вариантом является прекращение отладки (**Shift+F5**).

## Операторы цикла языка C/C++

В языке C/C++ имеются три оператора цикла:

### 1. Оператор цикла *for*.

Данный оператор используется в основном в тех случаях, когда в программе необходимо организовать выполнение цикла с параметром, т.е. цикла в котором некоторая переменная изменяется от начального до конечного значения с заданным шагом. Синтаксис этого оператора следующий:

```
for (<выражение_1>; <выражение_2>; <выражение_3>)
    <оператор>
```

Тело цикла составляет либо один оператор, либо несколько операторов, заключенных в фигурные скобки { ... } (составной оператор).

*Выражение\_1* описывает инициализацию цикла (обычно присваивает начальное значение управляющей переменной).

*Выражение\_2* – проверка условия завершения цикла. Если оно истинно, то выполняется оператор тела цикла.

*Выражение\_3* вычисляется после каждой итерации (обычно изменяет на каждом шаге значение управляющей переменной).

Пример:

```
for (int i=1; i<=10; i++)
    printf("i=%d\n", i);
```

Этот фрагмент программы осуществляет вывод на экран десяти значений переменной *i* от 1 до 10.

В отличие от некоторых других языков программирования, здесь в цикле *for* параметр может иметь любой тип и изменяться с произвольным шагом.

Любое из трех выражений может отсутствовать, но точки с запятой их разделяющие опускать нельзя

```
for ( ; ; ) // Бесконечный цикл
```

Выражения 1 и 3 могут состоять из нескольких выражений, объединенных операцией запятой.

Пример:

Написать программу, вычисляющую значение функции  $z = \ln(x)/\sin(y)$  при  $x \in [1; 1.5]$  изменяющимся с шагом  $h_1 = 0,1$  и  $y \in [1; 2]$  и изменяющимся с шагом  $h_2 = 0,2$ .

```
/*Включение в текст программы заголовочного файла стандартной
библиотеки ввода-вывода.*/
```

```
#include <stdio.h>
```

*/\*Включение в текст программы заголовочного файла с прототипами математических функций:\*/*

```
#include <math.h>
int main()
{
    double x,y,z;
    for (x=1,y=1;y<=2;x=x+0.1, y=y+0.2)
    {
        z=log(x)/sin(y);
        printf("x=%5.2f y=%5.2f z=%5.2f\n",x,y,z);
    }
}
```

## 2. Оператор цикла *while*.

Данный оператор реализует цикл с предусловием. Его синтаксис:

```
while (<выражение>
    <оператор>
```

Если выражение в скобках истинно, то выполняется оператор тела цикла, который может быть и составным.

*Пример:*

Реализовать предыдущий пример, с применением оператора *while*.

```
#include <stdio.h>
#include <math.h>
int main()
{
    double x=1,y=1,z;
    while (y<=2)
    {
        z=log(x)/sin(y);
        printf("x=%5.2f y=%5.2f z=%5.2f\n",x,y,z);
        x+=0.1;
        y+=0.2;
    }
}
```

## 3. Оператор цикла *do-while*.

Данный оператор реализует цикл с постусловием. Его синтаксис:

```
do
    <оператор>
while (<выражение>);
```

Если выражение в скобках истинно, то выполняется оператор тела цикла.

Пример:

Реализовать предыдущий пример, с применением оператора *do-while*.

```
#include <stdio.h>
#include <math.h>
int main()
{
    double x=1,y=1,z;
    do
    {
        z=log(x)/sin(y);
        printf("x=%5.2f y=%5.2f z=%5.2f\n",x,y,z);
        x+=0.1;
        y+=0.2;
    }
    while (y<=2);
}
```

## СОДЕРЖАНИЕ РАБОТЫ

1. Ознакомиться с теоретическим материалом.
2. В среде Visual Studio 2010 создать решение (консольное приложение). Настроить его свойства по аналогии с примером, рассмотренным в теоретических сведениях. В составе решения составить программу, которая выводит на экран ФИО студента, выполняющего работу и номер группы. Также программа должна содержать описание двух целочисленных переменных, которые вводятся с клавиатуры, а затем их сумма выводится на экран. Использовать сначала средства ввода-вывода языка C, затем C++.
3. При наборе программы отработать использование основных возможностей редактора кода.
4. Произвести отладку программы в обычном и пошаговом режимах. В отчет внести текст программы, а также скриншоты информационного окна после построения и при пошаговом выполнении программы (со значениями локальных переменных) и окна консоли с результатом работы программы.
5. Выбрать алгоритм, составить его блок-схему и программу с использованием оператора цикла *for* для вычисления и вывода на экран в точках  $x_i=a+i\cdot h$ ,  $i=0,1,2,\dots,n$ ,  $h=(b-a)/n$  промежутка  $[a,b]$  значений функции  $y=f(x)$ , указанной в варианте задания (см. ниже). Также программа должна определять наибольшее и среднее значение функции. Предусмотреть проверку

вычисляемых значений аргумента на принадлежность области допустимых значений. Ввод исходных данных  $(a, b, n)$  осуществлять с клавиатуры.

6. Составить аналогичные блок-схему и программу, но с использованием оператора цикла *while* или *do-while* на выбор.
7. Создать новое решение, в которое в виде отдельных проектов включить программы, созданные при выполнении пунктов 5 и 6. В отчет внести обе блок-схемы и программы, а также результаты их тестирования.

## ВАРИАНТЫ ЗАДАНИЯ

$$1. y = \frac{x}{x^2 - 1} + \log_3(x + 2),$$

$$x \in [2; 3], n = 10$$

$$2. y = \frac{x^3}{(x+1)(x+2)} + \frac{\arcsin(1-x)}{\sqrt[3]{1-\ln x}},$$

$$x \in [1; 2], n = 10$$

$$3. y = \frac{\sin x}{1 - \cos x} \cdot \frac{\operatorname{tg}^3(\ln(1-x))}{|1 + x \cdot e^{-x}|},$$

$$x \in [-1; -0,5], n = 5$$

$$4. y = \frac{-\arccos(1-x)}{\sqrt[4]{x^3 - 1}} + (2-x) \cos^2|x|,$$

$$x \in [1,5; 2], n = 5$$

$$5. y = \frac{\cos^2 x}{1 + \sin x} - \ln^2\left(\frac{x}{\sqrt[3]{x-1}}\right),$$

$$x \in [2; 3], n = 10$$

$$6. y = \frac{x^3 e^{x-1}}{x^3 - |x|} - \log_2(\sqrt{x} - x),$$

$$x \in [0,2; 0,8], n = 6$$

$$7. y = \frac{\sqrt[3]{x + \sin x}}{x^2 - x^4} \cdot \arcsin^2 \sqrt[4]{3-x},$$

$$x \in [2; 3], n = 10$$

$$8. y = \frac{x^5 + e^{-2|x|}}{\sqrt[4]{9-x^2}} \cdot \operatorname{tg}^3|\cos^2 x|,$$

$$x \in [1; 2], n = 10$$

$$9. y = \frac{\sin x + \frac{1}{x}}{\sqrt[3]{\operatorname{tg}^2\left(-\frac{x^3}{x^2-4}\right)}} + 2^{|x-1|},$$

$$x \in [1; 2], n = 5$$

$$10. y = \frac{\sin^2|x| + 3^{\frac{1}{x-1}}}{\sqrt[6]{x^4-16}} \cdot \sqrt{1-\ln x},$$

$$x \in [2,2; 2,6], n = 4$$

$$11. y = \frac{\sqrt[4]{8x^2 - 6x + 1}}{\arctg \sqrt{2x + 1}} + 2^{\sin x / |x|},$$

$$x \in [1; 2], n = 10$$

$$12. y = \frac{\ln^3(x-1)^2 + \cos^2(2x)}{\sqrt[6]{x^2 - 5x + 6}} \cdot \sin \frac{3^{x^2-1}}{2},$$

$$x \in [3,5; 4], n = 5$$

$$13. y = \sqrt[3]{\log_2(1-x)} + \frac{\operatorname{tg}(1+1/x)}{\sqrt{|x|-2}},$$

$$x \in [-2; -1], n = 10$$

$$14. y = \frac{1}{x} \log_3(4-x^2) + \frac{\sin(\cos x)}{e^{|x|-1}},$$

$$x \in [0,5; 1,5], n = 10$$

$$15. y = \frac{\sqrt[4]{|x|+1}}{\sin^2 \frac{x}{2} - 1} + 2^{\sqrt{x+1}},$$

$$x \in [0; 1], n = 10$$

$$16. y = \sqrt{\frac{1}{x}(x^2-1)} \cdot \cos^2 \frac{|x|}{3} + \lg \frac{1}{x+1},$$

$$x \in [1; 2], n = 10$$

$$17. y = \frac{x^3 + \sin(3|x|-1)}{1 - \cos^2 x} - \log_2(3^x - 9),$$

$$x \in [3; 4], n = 10$$

$$18. y = \frac{\sin^2 \sqrt[3]{x}}{x} + e^{-\sqrt{x^2-6x+8}},$$

$$x \in [1; 2], n = 10$$

$$19. y = (1+x)^{\sin \sqrt{x}} \cdot 2^{\cos^2\left(\frac{x}{x-2}\right)},$$

$$x \in [0; 1], n = 10$$

$$20. y = \frac{-x^2}{(2x+2)(2x-3)} + \frac{\log_2(\sqrt{x}-1)}{\sin 2x},$$

$$x \in [2; 3], n = 10$$

$$21. y = 2^{|x|} \cdot \ln |\sin x^4| - \cos^2 \sqrt{4-x^2},$$

$$x \in [1; 2], n = 10$$

$$22. y = \left[ \cos \left( e^{\sqrt{|x|-2}} + x^3 \right) \right]^{2x} - \frac{|x|}{x - \sqrt{x}},$$

$$x \in [2; 3], n = 10$$

$$23. y = e^{x^2-1} + \frac{x \cdot \sin \frac{1}{x}}{\sqrt[4]{9-\sqrt{x}}},$$

$$x \in [1; 2], n = 10$$

$$24. y = \frac{x}{\cos(x-\pi/2)\sin^2(x-\pi/2)} + e^{\sqrt{x}-|x|},$$

$$x \in [2; 3], n = 10$$

$$25. \quad y = \frac{2^{x^2} + \sqrt{16 - x^2}}{\sqrt[3]{x-2}} + \operatorname{tg}^2\left(\frac{x}{x+2}\right),$$

$$x \in [3; 4], n = 10$$

$$26. \quad y = \frac{1 + \log_2(\sin 2x)}{1 - 2x} + \frac{\sqrt[2]{|x| - 1}}{x^3 - 27},$$

$$x \in [1; 1.5], n = 5$$

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что в среде Visual Studio называется решением? Чем решение отличается от проекта?
2. Как создать решение?
3. Как в готовое решение добавляется еще один проект?
4. Как в состав проекта добавляются новые файлы?
5. Какие типы файлов входят в состав решение, проекта?
6. Каким образом в среде Visual Studio настраиваются свойства проекта?
7. Перечислите основные возможности редактора кода в среде Visual Studio?
8. Что такое конфигурация проекта? Как ее можно изменить?
9. Как можно произвести локализацию создаваемого консольного приложения?
10. Каким образом в среде Visual Studio производится отладка создаваемого приложения? Какие средства отладки вы знаете?
11. Формат записи оператора цикла *for*.
12. Формат записи оператора цикла *while*.
13. Формат записи оператора цикла *do-while*.
14. Каким образом можно включить несколько операторов в тело цикла?
15. Может ли управляющая переменная в цикле *for* быть вещественной?
16. Допустима ли форма записи цикла *for*, в которой отсутствует условие выхода? Если да, то сколько раз выполнится такой оператор?
17. Отличия оператора цикла *while* от *do-while*.

## ЛАБОРАТОРНАЯ РАБОТА № 2

### ОБРАБОТКА ОДНОМЕРНЫХ МАССИВОВ

*Цель работы:* приобрести практический опыт использования одномерных массивов.

#### КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Массивы являются примером структурных (составных) типов данных. Массив – это индексированный набор переменных определенного типа, имеющий общее для всех своих элементов имя. В зависимости от количества индексов, определяющих положение элемента, массивы подразделяются на одномерные, двумерные и т.д.

Пример использования одномерного массива:

```
#include<stdio.h>
#define N 3
int main()
{
    // Описание переменных:
    int list[N], i;
    // Присваивание элементам массива значений:
    list[0] = 421;
    list[1] = 53;
    list[2] = 1806;
    // Вывод элементов массива на экран:
    printf("Элементы массива: \n ");
    for(i=0; i<N; i++)
        printf("%d-й элемент: %d \n ",i+1,list[i]);
}
```

Результаты работы данной программы будут иметь вид:

Элементы массива:

1-й элемент: 421

2-й элемент: 53

3-й элемент: 1806

Выражение `int list[N]` объявляет `list` как массив переменных типа `int`, с объемом памяти, выделяемым для трех целых переменных (так как `N` равно 3). Индексы массива всегда целые и начинаются с нуля. К первой переменной массива можно обращаться как `list[0]`, ко второй – как `list[1]`,

к третьей – как *list[2]*. В общем случае описание любого массива имеет следующий вид:

*<тип> <имя>[размер]*

Наряду с непосредственным присваиванием, существуют и другие способы ввода элементов массива. Например, элементы массива можно ввести с клавиатуры:

```
int a[10],i;
printf("Введите 10 элементов массива:");
for(i=0; i<10; i++)
{
    printf("%d-й элемент --> ",i+1);
    scanf("%d", &a[i]);
}
```

Другим способом является непосредственная инициализация массива:

```
int a[10]={1,2,3,4,5,6,7,8,9,10};
```

## СОДЕРЖАНИЕ РАБОТЫ

Выбрать алгоритм, составить его блок-схему и программу для решения выбранного варианта задания. Исходный массив может быть введен с клавиатуры или инициализирован при описании.

Исходные и результирующие массивы вывести на экран в виде:

$x_0 \ x_1 \ x_2 \ x_3 \ x_4$

$x_5 \ x_6 \ x_7 \ x_8 \ x_9$

$x_{10} \ x_{11} \ x_{12} \ x_{13} \ x_{14}$

$x_{15} \ x_{16} \ x_{17} \ x_{18} \ x_{19}$

## ВАРИАНТЫ ЗАДАНИЯ

1. В заданном массиве  $X$ , состоящем из 20 элементов, определить и вывести на экран первый отрицательный элемент и его порядковый номер, а затем заменить его произведением предшествующих значений. Если все элементы положительны, выдать соответствующее сообщение.
2. Задан целочисленный массив  $X$  из 20 элементов. Вывести на экран все группы идущих подряд одинаковых элементов. Выдать соответствующее сообщение, если таких групп элементов в массиве нет.

3. Задан массив  $X$  из 20 элементов и число  $N$  ( $N < 20$ ). Не прибегая к сортировке, определить и вывести на экран  $N$  наибольших элементов массива.
4. В заданном целочисленном массиве  $X$ , состоящем из 20 элементов и упорядоченном по неубыванию, определить и вывести на экран те элементы, которые можно представить суммой двух других элементов.
5. Задан целочисленный массив  $X$  из 20 элементов. Определить и вывести на экран те элементы, делителем которых является хотя бы один из других элементов.
6. В заданном массиве  $X$ , состоящем из 20 элементов, определить и вывести на экран количество положительных, отрицательных и равных нулю элементов. Если положительных элементов больше (меньше), чем отрицательных, то заменить нулями нужное число положительных (отрицательных) элементов, чтобы их количество совпадало.
7. Задан целочисленный массив  $X$  из 20 элементов. Из этого массива переписать в массив  $Y$  ту последовательность, которая образует арифметическую прогрессию как минимум из пяти членов. Выдать соответствующее сообщение, если таких последовательностей нет.
8. Задан целочисленный массив  $X$  из 20 элементов. Переписать в массив  $Y$  те элементы исходного массива, которые строго больше двух своих соседей. Элементы массива  $Y$  не должны повторяться.
9. Задан целочисленный массив  $X$  из 20 элементов. Из этого массива переписать в массив  $Y$  той же размерности подряд все отрицательные элементы, а оставшиеся места заполнить единицами. Расположить элементы образованного массива в порядке убывания.
10. Задан целочисленный массив  $X$  из 20 элементов. Определить, можно ли из его положительных элементов составить строго возрастающую последовательность.
11. Задан целочисленный массив  $X$  из 20 элементов, содержащий как четные, так и нечетные числа. Из этого массива переписать в массив  $Y$  подряд первые пять различных четных элементов. Если таких элементов менее пяти, заполнить оставшиеся позиции в массиве суммой нечетных элементов массива  $X$ .
12. Задан целочисленный массив  $X$  из 20 элементов, содержащий группы подряд идущих одинаковых элементов. Поменять местами первую и последнюю группы массива.

13. Задан целочисленный массив  $X$  из 20 элементов. Определить максимальное количество идущих подряд и упорядоченных по возрастанию положительных чисел.
14. Задан целочисленный массив  $X$  из 20 элементов. Определить сумму элементов, имеющих четные индексы и являющихся нечетными числами. Если таковых нет, увеличить на единицу все элементы с четными индексами и вывести на экран результирующий массив.
15. Задан массив  $X$  натуральных чисел из 20 элементов. Удалить из него элементы, являющиеся удвоенными нечетными числами.
16. Задан массив  $X$  натуральных чисел из 20 элементов. Переписать в массив  $Y$  элементы, дающие при делении на 7 остаток 1, 2 или 5. Элементы массива  $Y$  упорядочить по неубыванию. В случае отсутствия таких элементов выдать соответствующее сообщение.
17. Задан целочисленный массив  $X$  из 20 элементов. Переписать в массив  $Y$  те элементы, которые равны сумме предшествующих им значений.
18. Задан действительный массив  $X$  из 20 элементов, содержащий 10 положительных и 10 отрицательных чисел. Переставить элементы массива так, чтобы положительные и отрицательные числа чередовались.
19. Задан целочисленный массив  $X$  из 20 элементов, среди которых есть повторяющиеся. Переписать в массив  $Y$  только неповторяющиеся элементы исходного массива.
20. Задан целочисленный массив  $X$  из 20 элементов, среди которых есть повторяющиеся. Записать в массив  $Y$  по одному элементу из каждой группы одинаковых значений исходного массива.
21. Задан целочисленный массив  $X$  из 20 элементов. Получить массив  $Y$ , в который переписать те положительные элементы массива  $X$ , которые расположены между двумя отрицательными. Если таких элементов нет, вывести соответствующее сообщение. Элементы массива  $Y$  не должны повторяться.
22. Задан целочисленный массив  $X$  из 20 элементов. Переписать в массив  $Y$  наибольшую по длине возрастающую последовательность исходного массива.
23. Задан целочисленный массив  $X$  из 20 элементов, среди которых есть повторяющиеся. Определить наименьший и наибольший элементы массива. Если они встречаются несколько раз, то оставить их по одному экземпляру, заменив остальные вхождения средним арифметическим наибольшего и наименьшего элементов.

24. Задан целочисленный массив  $X$  из 20 элементов, содержащий как положительные, так и отрицательные значения. Переставить элементы в массиве так, чтобы в начале располагались все положительные элементы, а затем все отрицательные. Порядок следования элементов в этих группах должен остаться прежним.
25. Задан целочисленный массив  $X$  из 20 элементов. Получить массив  $Y$ , в который записать те из элементов исходного массива в порядке следования, которые образуют наиболее длинную возрастающую последовательность.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение массива.
2. Как производится доступ к отдельным элементам массива?
3. Что такое указатель? Как он описывается?
4. Что общего у понятий массива и указателя?
5. Как с помощью указателя обратиться к элементу массива?
6. Какие способы заполнения массива значениями вы знаете?
7. Как определяется символьный массив?
8. Что представляет собой строка символов в языке C?
9. Каково внутреннее представление строковых констант?
10. Чем ограничен размер строки символов в языке C?
11. Какие ограничения накладываются на индексы массивов?
12. Может ли быть индекс массива равен значению его размерности?

## ЛАБОРАТОРНАЯ РАБОТА № 3

### ОБРАБОТКА ДВУМЕРНЫХ МАССИВОВ. ФАЙЛОВЫЙ ВВОД-ВЫВОД. ПРИМЕНЕНИЕ ИТЕРАТИВНЫХ И РЕКУРСИВНЫХ ФУНКЦИЙ

*Цель работы:* ознакомиться с организацией двумерных массивов в языке C/C++; приобрести практические навыки в файловом вводе-выводе данных; ознакомиться с организацией передачи параметров в функции по ссылке; получить навыки описания рекурсивных функций.

#### КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

##### Организация многомерных массивов

В языке C/C++ многомерные (в частности двумерные) массивы – это массивы, элементами которых в свою очередь являются другие массивы. Например, конструкция:

```
int mass[3][5];
```

описывает массив из трех элементов, каждым из которых является массив из пяти элементов целого типа. Фактически это матрица (3×5). Доступ к элементам осуществляется указанием двух индексов:

```
mass[0][1]=25; // присвоить 25 элементу, находящемуся в  
// 1-й строке и 2-м столбце
```

Для работы с двумерными массивами, как правило, требуется применение вложенных циклов. В качестве примера рассмотрим ввод элементов массива с клавиатуры:

```
int a[5][5], i, j;  
for(i=0; i<5; i++)  
    for(j=0; j<5; j++)  
        scanf("%d", &a[i][j]);
```

Двумерные массивы можно инициализировать при описании так же как и одномерные. Например:

```
int a[2][3]={  
    {1,2,3},  
    {4,5,6}  
}
```

## Доступ к файлам

Ввод-вывод на верхнем уровне в языке Си осуществляется через **потоки**. Поток является файлом или физическим устройством (например, принтером или монитором), которое управляется с помощью указателей на объект типа FILE (определенный в *stdio.h*). Структура FILE содержит различную информацию о потоке, включая текущую позицию потока, указатель на соответствующие буферы и индикаторы ошибки или конца файла. Открывается поток с помощью функции *fopen*:

```
FILE *fopen(char *pathname, char *type);
```

Строка *pathname* – это путь к файлу и его имя. Строка *type* – разрешенный тип доступа (например "w" – для записи, "r" – для чтения). Функция возвращает указатель на структуру типа FILE, который в дальнейшем передается другим функциям ввода-вывода. Нулевое значение указателя (*NULL*) говорит об ошибке открытия файла.

Для закрытия потока используется функция *fclose*:

```
int fclose(FILE *fp);
```

В дополнение к потокам, создаваемым вызовом *fopen*, три предопределенных (стандартных) потока автоматически открываются всякий раз, когда начинается выполнение программы:

Имя предопределенного потока	Тип	Режим	Описание
stdin	ввод	Текстовый	Стандартный ввод
stdout	вывод	Текстовый	Стандартный вывод
stderr	вывод	Текстовый	Стандартная ошибка

Имя предопределенного потока допустимо указывать в любых функциях ввода-вывода вместо имени пользовательского потока. Закрыть предопределенный поток нельзя. Его можно только переопределить с помощью функции *freopen*, имеющей следующий прототип:

```
FILE *freopen(char *pathname, char *type, FILE *fp);
```

Для чтения данных из файла используется функция *fscanf*, являющаяся аналогом функции *scanf*, за исключением того, что первым параметром является указатель на файл. Например,

```
fscanf(fr, "%d", &a);
```

Для записи данных в файл используется функция *fprintf*, являющаяся аналогом функции *printf*, за исключением того, что первым параметром является указатель на файл. Например,

```
fprintf(fw, "y = %d", y);
```

*Пример:* написать программу, считывающую из файла *in.txt* элементы матрицы  $A(5 \times 5)$  и записывающую их в файл *out.txt* в виде транспонированной матрицы.

```
#include <stdio.h>
#define N 5 // размерность матрицы
int main(void)
{
    int i, j, a[N][N];
    FILE *fp; // указатель на файловую структуру
    fp=fopen("in.txt", "r"); // открываем файл in.txt
                               // для чтения
    /* файл in.txt должен существовать в текущем каталоге и
    содержать 25 элементов матрицы a. Далее проверяется корректность
    открытия файла*/
    if(fp)
    {
        for(i=0; i<N; i++)
            for(j=0; j<N; j++)
                fscanf(fp, "%d", &a[i][j]); // считываем
                                               // элементы матрицы
        fclose(fp); // закрываем файл
        fp=fopen("out.txt", "w"); // открываем файл out.txt
                                   // для записи
        // записываем элементы массива a в виде
        // транспонированной матрицы в файл out.txt:
        for(i=0; i<N; i++)
        {
            for(j=0; j<N; j++)
                fprintf(fp, "%5d", a[j][i]);
            fprintf(fp, "\n");
        }
        fclose(fp); // закрываем файл
    } else printf("Входной файл отсутствует\n");
}
```

## Описание функций

Определение функции состоит из ее заголовка и составного оператора тела функции. В заголовке указывается тип функции, ее имя и в скобках список формальных параметров (если они есть). Параметры любых типов (кроме массивов) в обычном случае передаются в функции по значению. Подразумевается, что функция может возвращать результат своей работы, объявленного при ее описании типа, через свое имя (если она не типа *void*). Функция может возвращать значение любого типа, кроме массивов и функций. Но функция также может возвращать указатель на объект любого типа, в том числе и на массив или функцию.

Однако в некоторых случаях, например при необходимости возврата нескольких значений, применяется передача параметров по ссылке. Для этого имеются две возможности:

### 1. Ссылочные параметры.

*Пример:* функция инкремента целого числа:

```
void inc(int &x)
{
    x++;
}
```

.....  
*// Вызов функции:*

```
int x=1;
inc(x);
```

### 2. Указатели.

*Пример:* функция инкремента целого числа:

```
void inc(int *x)
{
    (*x)++;
}
```

.....  
*// Вызов функции:*

```
int x=1;
inc(&x);
```

Поскольку имя массива является константным указателем, то он *всегда* передается в функцию по ссылке. В случае одномерного массива в функцию фактически передается адрес первого элемента. Если даже указать при описании такого параметра размерность массива, то она будет игнорироваться, так как нет никакого контроля за тем, выходит ли при обращении к элементу его вычисленный адрес за границы массива.

Поэтому при описании в качестве параметра функции одномерного массива возможны разные варианты записи: с указанием или без указания размерности массива или непосредственно в виде указателя на базовый тип элементов (см. пример ниже).

Двумерный массив представляет собой массив, элементами которого являются одномерные массивы (строки). При вызове функции, имеющей в качестве параметра двумерный массив, будет передаваться указатель на первую строку матрицы. В связи с этим также возможны разные варианты записи параметров: с указанием обеих размерностей массива или только одной (второй, т.е. количества элементов в строке), а также в виде указателя на одномерный массив, размер которого совпадает со второй размерностью матрицы (см. пример ниже).

Если среди параметров, передаваемых в функцию по ссылке, имеются такие, которые не должны изменяться в процессе ее работы, перед ними при описании можно указывать модификатор *const*.

Как было сказано выше, функция в качестве результата может возвращать указатель на функцию. Но также верным является то, что такой указатель может быть параметром функции. Общий синтаксис описания указателя на функцию следующий:

`<тип ф-ии> (*<имя указателя>)(<список типов парам-в ф-ии>)`

Например, если имеется функция:

```
int sum(int a, int b)
{
    return a+b;
}
```

то указатель, ссылающийся на нее можно описать следующим образом:

```
int (*pf)(int, int)=&sum;
```

*Пример:* дана целочисленная матрица  $A(5 \times 5)$ . Определить массив  $X$  из пяти элементов, равных элементам побочной диагонали матрицы  $A$ . Элементы матрицы по выбору пользователя либо вводятся с клавиатуры, либо генерируются случайным образом.

```
#include <stdio.h>
#include <stdlib.h> //Содержит прототипы функций для
                  //работы с ГСЧ
#include <conio.h>
#include <time.h> //Содержит прототипы функций для
                  //работы с временем
#include <locale.h>
```

```

#define N 5
// Функция ввода элементов матрицы с клавиатуры:
void input_matr(int a[N][N])
{
    int i,j;
    printf("Введите элементы матрицы A: \n");
    for (i=0;i<N;i++)
        for (j=0;j<N;j++)
            scanf("%d",&a[i][j]);
}
// Функция заполнения матрицы с помощью ГСЧ:
void input_matr_rand(int a[][N])
{
    int i,j;
    srand((unsigned)time(NULL)); //Инициализация ГСЧ
//Функция time возвращает текущее время в секундах
    for (i=0;i<N;i++)
        for (j=0;j<N;j++)
            a[i][j]=rand()%50; // Значения элементов от 0 до 49
}
// Функция вычисления элементов массива X:
int *mas_x(int (*a)[N], int *x,
           void (*pfunc)(int [N][N]))
{
    int i,j;
    // Вызов через указатель одной из двух функций ввода элементов a:
    pfunc(a);
    for (i=0, j=N-1;i<N;i++, j--)
        x[i]=a[i][j];
    return x;
}
// Функция вывода на экран элементов матрицы a и массива x:
void output(const int x[], const int a[][N] )
{
    int i,j;
    printf("Матрица A:\n");
    for (i=0;i<N;i++)
    {
        for (j=0;j<N;j++)
            printf("%5d",a[i][j]);
        printf("\n");
    }
}

```

```

}
printf("Массив X:\n");
for (i=0;i<N;i++)
    printf("%5d",x[i]);
printf("\n");
}
int main()
{
    setlocale(LC_CTYPE,"");
    int a[N][N], x[N], c;
    void (*pfunc)(int [N][N]);
    do
    {
        printf("Ввод матрицы:\n1 - с клавиатуры\n2- ГСЧ\n");
        scanf("%d",&c);
    }
    while (c!=1 && c!=2);
    // Присваивание указателю адреса одной из функций:
    switch (c)
    {
        case 1: pfunc=&input_matr;
                break;
        case 2: pfunc=&input_matr_rand;
    }
    output(mas_x(a,x,pfunc),a);
    _getch();
}

```

## Рекурсивные функции

Функция называется рекурсивной, если она вызывает саму себя (прямая рекурсия), или вызывает другую функцию, которая в свою очередь вызывает первую (косвенная рекурсия).

Глубина рекурсии должна быть конечной. При выполнении очередного рекурсивного вызова система создает в стеке новые экземпляры всех автоматических переменных функции и ее параметров. Поэтому при большой глубине рекурсии возможно переполнение стека и аварийное завершение работы программы.

Также следует аккуратно обращаться в рекурсивных функциях с глобальными переменными, так как их изменение отразится во всех последующих вызовах.

*Пример:* составить рекурсивную функцию вычисления факториала:

```
int factorial_recurs(int n)
{
    if (n==1 || n==0)    return 1;
    else return n*factorial_recurs(n-1);
}
```

## СОДЕРЖАНИЕ РАБОТЫ

Выбрать алгоритм, составить его блок-схему и программу для решения выбранного варианта задания. Программа должна по выбору пользователя осуществлять ввод исходной матрицы с клавиатуры или из файла. Для этого программа должна содержать две соответствующие функции, указатель на одну из которых необходимо передавать в функцию для вычисления элементов массива  $X$ . Данная функция должна вызывать через указатель одну из функций ввода элементов матрицы, производить вычисление элементов массива  $X$  в соответствии с заданием и возвращать указатель на этот массив. Кроме того, программа должна содержать функцию для вывода на экран и в файл исходной матрицы и результирующего массива, а также рекурсивную функцию определения в соответствии с заданием величины  $Y$ .

В программе не должно быть глобальных переменных.

## ВАРИАНТЫ ЗАДАНИЯ

1. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен сумме элементов соответствующей строки верхней треугольной матрицы. Определить величину  $Y$ , как наибольший из элементов массива  $X$ .
2. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен произведению элементов соответствующего столбца нижней треугольной матрицы. Определить величину  $Y$ , как сумму положительных элементов массива  $X$ .
3. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен наибольшему элементу соответствующей строки матрицы. Определить величину  $Y$ , как наименьший из положительных элементов массива  $X$ .
4. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен 1, если произведение элементов

соответствующего столбца больше нуля и  $-1$  в противном случае. Определить величину  $Y$ , как количество повторений  $1$  среди элементов массива  $X$ .

5. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен наименьшему из элементов соответствующего столбца матрицы. Определить величину  $Y$ , как количество нечетных элементов, расположенных перед наибольшим из элементов массива  $X$ .
6. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен  $1$ , если количество положительных элементов в соответствующей строке больше количества отрицательных и  $-1$  в противном случае. Определить величину  $Y$ , как количество четных элементов в первой строке матрицы  $A$ .
7. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен среднему арифметическому наибольшего и наименьшего из элементов соответствующего столбца матрицы. Определить величину  $Y$ , как сумму элементов, расположенных перед наименьшим элементом массива  $X$ .
8. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен  $1$ , если в соответствующей строке элемент главной диагонали больше элемента побочной и  $-1$  в противном случае. Определить величину  $Y$ , как количество нечетных элементов в первом столбце матрицы  $A$ .
9. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен  $1$ , если элементы упорядочены по возрастанию или по убыванию и  $-1$  в противном случае. Определить величину  $Y$ , как среднее арифметическое наибольшего и наименьшего элемента главной диагонали матрицы  $A$ .
10. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен первому отрицательному элементу соответствующей строки матрицы или нулю, если все элементы строки положительны. Определить величину  $Y$ , как индекс первого отрицательного элемента массива  $X$ .
11. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен  $1$ , если все элементы соответствующей строки положительны и  $-1$  в противном случае. Определить величину  $Y$ , как  $1$ , если элементы первой строки матрицы образуют арифметическую прогрессию и  $0$  в противном случае.
12. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен соответствующему элементу столбца  $s$

наибольшей среди других столбцов суммой положительных элементов. Определить величину  $Y$ , как 1, если элементы массива  $X$  образуют последовательность Фибоначчи ( $f_1 = f_2 = 1$ ,  $f_i = f_{i-1} + f_{i-2}$  для  $i > 2$ ) и 0 в противном случае.

13. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен 1, если наименьший элемент соответствующей строки положителен и  $-1$  в противном случае. Определить величину  $Y$ , как наибольший из индексов элементов массива  $X$ , равных 1.
14. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен сумме элементов соответствующей строки, если они все либо положительны либо отрицательны, и нулю в противном случае. Определить величину  $Y$ , как сумму элементов массива  $X$ , расположенных после первого нулевого элемента.
15. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен среднему арифметическому элементов строки и столбца, на пересечении которых находится соответствующий элемент побочной диагонали. Определить величину  $Y$ , как произведение четных элементов, расположенных после наименьшего элемента массива  $X$ .
16. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен количеству вхождений в соответствующую строку наибольшего из элементов матрицы. Определить величину  $Y$ , как количество нулевых элементов массива  $X$ .
17. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен сумме элементов того столбца, в котором находится первый положительный элемент соответствующей строки, и нулю, если все элементы строки неположительны. Определить величину  $Y$ , как количество отрицательных элементов, расположенных перед наибольшим элементом массива  $X$ .
18. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен 1, если в соответствующем столбце есть возрастающая подпоследовательность из трех элементов и нулю в противном случае. Определить величину  $Y$ , как произведение нечетных элементов, расположенных перед первым встретившимся четным элементом массива  $X$ .
19. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен 1, если наименьший из элементов

соответствующей строки совпадает с наименьшим элементом матрицы и  $-1$  в противном случае. Определить величину  $Y$ , как сумму четных элементов первой строки матрицы, расположенных после первого нечетного элемента

20. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен элементам того столбца, в котором находятся наибольший и наибольший по модулю элементы матрицы, или элементам побочной диагонали, если они находятся в разных столбцах. Определить величину  $Y$ , как произведение отрицательных элементов массива  $X$ , расположенных после первого положительного элемента.
21. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен сумме элементов соответствующей строки, предшествующих первому в ней отрицательному элементу. Определить величину  $Y$ , как количество повторений наименьшего элемента в первой строке матрицы.
22. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен 1, если сумма модулей элементов соответствующего столбца больше наибольшего по модулю элемента матрицы и  $-1$  в противном случае. Определить величину  $Y$ , как сумму положительных элементов первой строки матрицы, расположенных после первого нулевого элемента.
23. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен произведению элементов соответствующего столбца, расположенных за первым в нем отрицательным элементом. Определить величину  $Y$ , как количество отрицательных элементов первой строки матрицы, имеющих нечетные номера столбцов.
24. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен 1, если количество четных элементов в соответствующей строке больше количества нечетных и  $-1$  в противном случае. Определить величину  $Y$ , как произведение положительных элементов, расположенных после наибольшего элемента первой строки матрицы.
25. Дана матрица  $A(5 \times 5)$ . Определить массив  $X$  из 5 элементов, каждый из которых равен 1, если наибольший по модулю элемент соответствующей строки совпадает с наименьшим по модулю элементом побочной диагонали и  $-1$  в противном случае. Определить величину  $Y$ , как сумму элементов первой строки матрицы, расположенных между наибольшим и наименьшим элементом.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каковы в языке C/C++ принципы размещения в памяти многомерных массивов? Как производится их описание?
2. Как производится обращение к элементам многомерного массива?
3. Какими способами можно произвести заполнение многомерного массива элементами?
4. Как осуществляется файловый ввод-вывод в языке C?
5. В каком файле определены прототипы функций ввода-вывода верхнего уровня?
6. Какие функции осуществляют открытие и закрытие файла?
7. Какие функции предназначены для форматированного ввода-вывода данных?
8. В чем заключается различие в принципах передачи в функцию параметров по значению и по ссылке?
9. Какие вы знаете способы передачи параметров по ссылке в языке C/C++?
10. Каким образом передаются в функции массивы?
11. Возможен ли возврат функцией таких типов данных, как структуры и объединения?
12. Возможен ли возврат функцией массива?
13. Назовите преимущества и недостатки рекурсивных функций по сравнению с итеративными.
14. В каком случае задача может иметь рекурсивное решение?
15. Каков механизм вызова рекурсивной функции?
16. Какие условия должны выполняться при описании рекурсивных функций?
17. Как описываются функции с косвенной рекурсией?

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Информатика: базовый курс / под ред. С.В. Симоновича. – СПб.: Питер, 2008. – 640 с.
2. Информатика: учеб. / под ред. Н.В. Макаровой. – 3-е изд. – М.: Финансы и статистика, 2007. – 768 с.
3. *Могилев, А.В.* Информатика: учеб. пособие/ А.В. Могилев, Н.И. Пак, Е.К. Хеннер. – М.: Академия, 2004. – 848 с.
4. *Острейковский, В.А.* Информатика: учеб. / В.А. Острейковский. – М.: Высшая школа, 2007.– 511 с.
5. *Демидович, Е.М.* Основы алгоритмизации и программирования. Язык СИ: учеб. пособие/ Е.М. Демидович. – СПб.: БХВ-Петербург, 2006. – 439 с.
6. *Костюкова, Н.И.* Язык Си и особенности работы с ним: учеб. пособие/ Н.И. Костюкова, Н.А. Калинина. – М.: БИНОМ. Лаборатория знаний, 2006. – 205 с.
7. *Павловская, Т.А.* С/С++. Программирование на языке высокого уровня: учеб./ Т.А. Павловская. – СПб.: Питер, 2009.– 432 с.
8. *Подбельский, В.В.* Курс программирования на языке Си: учеб./ В.В. Подбельский, С.С. Фомин. – М.: ДМК Пресс, 2012.– 384 с.
9. *Скляров, В.А.* Программирование на языках СИ и СИ++: учеб. пособие/ В.А. Скляров. – М.: Высшая школа, 1999. – 288 с.

