Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Институт повышения квалификации и переподготовки

Кафедра «Профессиональная переподготовка»

Е. И. Гридина

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ НА ЯЗЫКАХ ВЫСОКОГО УРОВНЯ

ПРАКТИКУМ
по одноименной дисциплине
для слушателей специальности переподготовки
1-40 01 73 «Программное обеспечение
информационных систем»
заочной формы обучения

Гомель 2018

УДК 004.921(075.8)
ББК 32.973-018я73
Г82

Рекомендовано кафедрой «Профессиональная переподготовка» ИПКиП ГГТУ им. П. О. Сухого
(*протокол № 9 от 25.05.2017 г.*)

Рецензент: доц. каф. «Информационные технологии» ГГТУ им. П. О. Сухого
канд. техн. наук, доц. *В. В. Комраков*

Гридина, Е. И.

Г82 Основы алгоритмизации и программирования на языках высокого уровня : практикум по одноим. дисциплине для слушателей специальности переподготовки 1-40 01 73 «Программное обеспечение информационных систем» заоч. формы обучения / Е. И. Гридина. – Гомель : ГГТУ им. П. О. Сухого, 2018. – 73 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

Данное издание представляет собой лабораторный практикум по дисциплине «Основы алгоритмизации и программирования на языках высокого уровня».

Практикум написан в соответствии с требованиями, предъявленными к оформлению методических пособий, доступным языком и содержит множество примеров, позволяющих в кратчайшие сроки овладеть основными принципами алгоритмизации и основ программирования на языке Си.

Издание адресовано слушателям ИПКиП специальности 1-40 01 73 «Программное обеспечение информационных систем».

УДК 004.921(075.8)
ББК 32.973-018.2я73

Содержание

Лабораторная работа № 1

Тема работы: Изучение основных понятий и приёмов работы со средой разработки. Программирование задач, содержащих алгоритмы линейной структуры.

Теоретические сведения по изучению темы:

1.Сущность алгоритма линейной структуры.
2.Типы данных и объявления данных.
3.Операции. Приоритет операций.
4.Функции getchar() и putchar().
5.Понятие буферизации.
6.Функция printf().
7.Функция scanf().
8.Оператор присваивания.

Изложение вопросов:

1. Под алгоритмом линейной структуры понимают алгоритм, включающий в себя прямую конечную последовательность действий Алгоритм линейной последовательности является наиболее простым.

2. В языке С различают следующие типы данных :

int - задает значения, к которым относятся целые числа , например -6, 0, 28 и т. д . Для 16-разрядных ЭВМ диапазон значений типа int занимает 16 бит. Для 32-разрядных - 32 бит или 4 байта.

short - в 16-разрядных системах short=int ; в 32-х-short=16 бит.

long - для 16-разрядных систем диапазон значений лежит от – 2147483648 до 2147483647; занимает переменная 32 бита; для 32-разрядных ЭВМ long = int.

char - задает значения, которые представляют различные символы. Например: w, y, 4, ! и т. д. Такими символами могут быть почти все знаки из кодовой таблицы ЭВМ .

Исключение составляют лишь специальные управляющие знаки, не имеющие графического изображения. И в 16-разрядных и в 32-разрядных ЭВМ данные типа char занимают в памяти 8 бит или 1 байт.

unsigned - в языке С можно задавать некоторые типы как unsigned . К этим типам относятся char, short, int, long.

Такое задание означает, что соответствующие переменные не будут иметь отрицательных значений. В результате они могут принимать большие положительные значения, чем переменные обычных типов. Например, задание unsigned int может принимать значения от 0 до 65535 при том же размере, что и int (16 бит).

Замечание : в случае типа int запись вида unsigned int a; можно записывать более сжато: unsigned a .

float - задает значения, к которым относятся вещественные числа, имеющие дробную часть. Например, 5.27; -1.58e+2; 3.61e-4. И в 16-ти и в 32-разрядных системах занимает 32 бита. Значения находятся в диапазоне от ±3.4e-38 до ±3.4e+38.

double - тип float с двойной точностью .

Занимают в два раза больше места, чем тип float, то есть 64 бита.

Диапазон представления от ±1.7e-308 до ±1.7e+308.

enum – предназначен для описания объектов из некоторого заданного множества , например , {весна, лето, осень, зима}.

void – используется для обозначения величин , имеющих нулевую длину и не имеющих значения (не определенных).

В языке Си все данные должны быть объявлены раньше, чем будут использоваться. Объявление данных осуществляется в операторах объявления, которые определяют тип и список, состоящий из одной или более переменных этого типа . Например: int lower, upper, step; char ch , line[100]; enum tip _ year{весна, лето, осень, зима} a, b, c.

При объявлении переменная может быть инициализирована , например:char esc =’\\’; int i=0 ; int limit = MAXLINE+1; float eps = 1.0e-5.

К любой переменной при объявлении может быть применен квалификатор **const** для указания того, что ее значение далее в программе изменяться не будет . Например : const double e = 2.71828182845905; const char msg [] = “Предупреждение”;

Замечание: Реакция на попытку изменить переменную, помеченную квалификатором const оставляется обычно на усмотрение разработчиков компилятора.

Правила при объявлении переменных:

1. Внешние и статические переменные, по умолчанию инициализируются нулем .
2. Автоматические переменные, явным образом не инициализированные, содержат неопределенные значения (мусор).
3. В Си необходимо различать порядок вычисления операций и приоритет (старшинство) операций.

Порядок – это последовательность вычислений (слева направо, справа налево). Приоритет – это то, какая операция выполняется в выражении первой, какая второй и т. д. В табл. 1 перечислены все операции Си и Си++ с указанием их приоритета, начиная с высшего и кончая низшим, и показан порядок вычисления (слева направо или справа налево). Все операции, указанные внутри строк, имеют одинаковый приоритет, как указано в таблице 1.1.

Таблица 1.1 – Порядок вычисления операций

Значение	**Операция**	**Порядок вычисления**
1	**2**	**3**
Вызов функции	()	Слева направо
Индекс массива	[]	
Выделение элемента структуры, Объединения	.	
Выделение элемента(структуры, объединения)	.>	
Указатель на метод(объекты)	.*	
Указатель на метод(указатели)	.>*	
Постфиксный инкремент	++	Справа налево
Постфиксный декремент	--	
Операция для базы(не для 32-разр компиляции)	:>	
Доступ к области действия	::	
Логическое отрицание NOT	|	Справа налево
Поразрядное отрицание	~	
Унарный минус	-	
Унарный плюс	+	
Префиксный инкремент	++	
Префиксный декремент	--	
Определение адреса	&	
Обращение по адресу	*	
Размер объекта в байтах	sizeof	
Приведение типа	(тип)	
Динамическое выделение памяти	new	
Динамическое освобождение Памяти	delete	
Умножение	*	
Деление	/	Слева направо
Определение остатка	%	
Сложение	+	Слева направо
Вычитание	-	
Сдвиг влево	<<	Слева направо
Сдвиг вправо	>>	
Меньше , чем	<	Слева направо
Больше , чем	>	
Меньше или равно	<=	Слева направо
Больше или равно	>=	
Равно	==	Слева направо
Не равно	!-	
Поразрядное И	&	Слева направо
Поразрядное исключающие ИЛИ	^	
Поразрядное ИЛИ	|	
Логическое И	&&	Слева направо

Продолжение таблицы 1.1

Логическое ИЛИ	||	
Условная операция	?:	Справа налево
Присваивание	=	Справа налево
Умножить и присвоить	*=	
Разделить и присвоить	/=	
Разделить по модулю и присвоить	%=	
Сложить и присвоить	+=	
Вычесть и присвоить	-=	
Сдвинуть влево и присвоить	<<=	
Сдвинуть вправо и присвоить	>>=	
Поразрядное И и присвоить	&=	
Поразрядное исключающее ИЛИ и Присвоить	^=	
Поразрядное ИЛИ и присвоить	| =	
Запятая	,	Слева направо

4.Функция getchar() получает один символ , поступающий с устройства стандартного ввода-вывода (клавиатура) и передает его выполняющейся в данный момент программе.

Функция putchar () получает один символ , поступающий из программы и пересылает его для вывода на экран .

Функция getchar() аргументов не имеет. Она получает символ и возвращает его программе. Функция putchar() имеет один аргумент, который представляет собой символ, выводимый на печать. Аргументом функции putchar() могут быть:

- одиночный символ, включая знаки, представляемые управляющими последовательностями;
- переменная, значением которой является одиночный символ;
- функция, значением которой является одиночный символ;

Пример правильного указания функции putchar() :

```
putchar('s');
putchar('\n');
putchar('\007');
putchar(ch);
putchar(genchar())';
```

Пример:

```
#include<stdio.h>
void main (void)
{
  putchar(getchar());
}
```

5. Буфер – это некоторая область оперативной памяти, организуемая временно для помещения в нее информации. Если буфер заполнится, содержимое его передается по назначению, и процесс буферизации начинается снова.

Функции стандартного ввода-вывода – это функции с буферизацией, то есть не сразу осуществляют ввод-вывод, а через буфер.

6. Функции printf() и scanf() дают возможность взаимодействовать с программой на уровне стандартного ввода-вывода.

Обычно функции printf() и scanf() работают во многом одинаково – каждая использует управляющую строку и список аргументов. Вначале рассмотрим работу функции printf(),а затем scanf().

Функция printf() осуществляет форматный вывод на устройство стандартного вывода (дисплей). Общий вид обращения к функции printf() следующий:

printf(управляющая строка, аргумент 1, аргумент 2,…);

где аргумент 1, аргумент 2 и т.д. - выводимые параметры, в качестве которых могут быть:

- переменные;
- константы;
- выражения, которые вычисляются перед выводом.

Управляющая строка – строка символов, показывающая, как должны быть выведены параметры. Управляющая строка обрамляется кавычками.

Пример:

printf(“%d женщин съели %d плиток шоколада.\n”, number, shok);

Управляющая строка содержит информацию двух различных видов:

-символы, выводимые текстуально;

-идентификаторы данных, которые называются также спецификациями преобразования.

Правило: каждому аргументу из списка, следующего за управляющей строкой, должна соответствовать одна спецификация преобразования.

Замечание: поскольку символ % используется в функции printf() в спецификации преобразования, то если нужно вывести сам символ %, просто надо написать два символа % подряд.

Пример: printf(“Только %d %% стряпни Анны было съедено.\n”,pe);

Спецификации преобразования могут содержать внутри себя модификаторы. Они помещаются между знаком % и символом, определяющим тип преобразования.

Каждому типу выводимой информации соответствует своя спецификация преобразования. Спецификации формата приведины в таблице 1.2.

Таблица 1.2 – Спецификации формата

Формат	Тип выводимой информации
%d	десятичное целое число
%c	один символ
%s	строка символов
%e	число с плавающей точкой, экспоненциальная запись
%f	число с плавающей точкой, десятичная запись
%g	число с плавающей точкой в формате %e или %f
%u	десятичное целое число без знака
%o	восьмеричное целое число без знака
%x	шестнадцатеричное целое число без знака
%p	указатель в формате xxxx:yyyy, где xxxx-сегмент, yyyy – смещение, а x и y шестнадцатеричные цифры

7.Функция scanf() во многом идентична функции printf(), с той лишь разницей, что она осуществляет форматное чтение с устройства стандартного ввода (клавиатуры). Так же, как и функция printf() она имеет управляющую строку и список аргументов. Главное различие двух этих функций заключается в особенностях списка аргументов. Функция prihtf() использует в качестве аргументов переменные, константы, выражения, в то время как scanf() - только указатели на переменные, как показано на рисунке 1.1. При применении функции scanf() надо соблюдать два правила:

1) при вводе значения некоторой переменной в качестве аргумента используется не имя переменной, а адрес этой переменной в памяти, то есть указатель на переменную, поэтому перед именем переменной ставится символ операции получения адреса &;

2) если вводится значение строковой переменной, использование символа & необязательно, ибо строка сама указывает на себя в памяти.

```
void main (void)
 {
   int age;
   float assets;
   char pet[30];
   printf("Укажите возраст, состояние и любимое животное:\n");
     scanf("%d  %f", &age , &assets);
     scanf("%s", pet); /* & отсутствует при указании массива символов */
     printf("%d   %f  %s \n", age assets , pet);
   }
```

Рисунок 1.1 – Пример программы с линейной структурой алгоритма

Функция scanf() использует некоторые специальные знаки (пробелы, табуляции, "enter") для разбиения входного потока символов на отдельные поля. Она согласует последовательность спецификаций преобразования с последовательностью полей, опуская упомянутые специальные знаки между ними. Единственным исключением из этого является спецификация %c, которая обеспечивает чтение каждого следующего символа даже в том случае, если это пустой символ.

Функция scanf() использует практически тот же набор спецификаций преобразования, что и функция printf(). Основные отличия в случае функции scanf() :

1) отсутствует спецификация %g;
2) спецификации %f и %e эквиваленты;
3) для чтения целых чисел типа short применяется %h;

8. Оператор присваивания – это основная рабочая сила большинства программ. С помощью этого оператора переменной присваивается некоторое значение.

Общий вид оператора присваивания:

N=T; где N-имя переменной;

T-выражение.

Вход в Visual Studio

При первом запуске Visual Studio предоставляется возможность выполнить вход с использованием учетной записи Майкрософт, например Live или Outlook.Вход позволяет обеспечить синхронизацию пользовательских параметров на всех устройствах.Для получения дополнительной информации см. Вход в Visual Studio.

После открытия Visual Studio можно увидеть три основные части интегрированной среды разработки: окна инструментов, меню с панелями инструментов и область главного окна.Окна инструментов закреплены в левой и правой частях окна приложения, а панель **Быстрый запуск**, строка меню и стандартная панель инструментов закреплены в его верхней части.В центре окна приложения находится **Начальная страница**. При открытии решения или проекта редакторы и конструкторы отображаются в этом пространстве (рисунок 1.2).

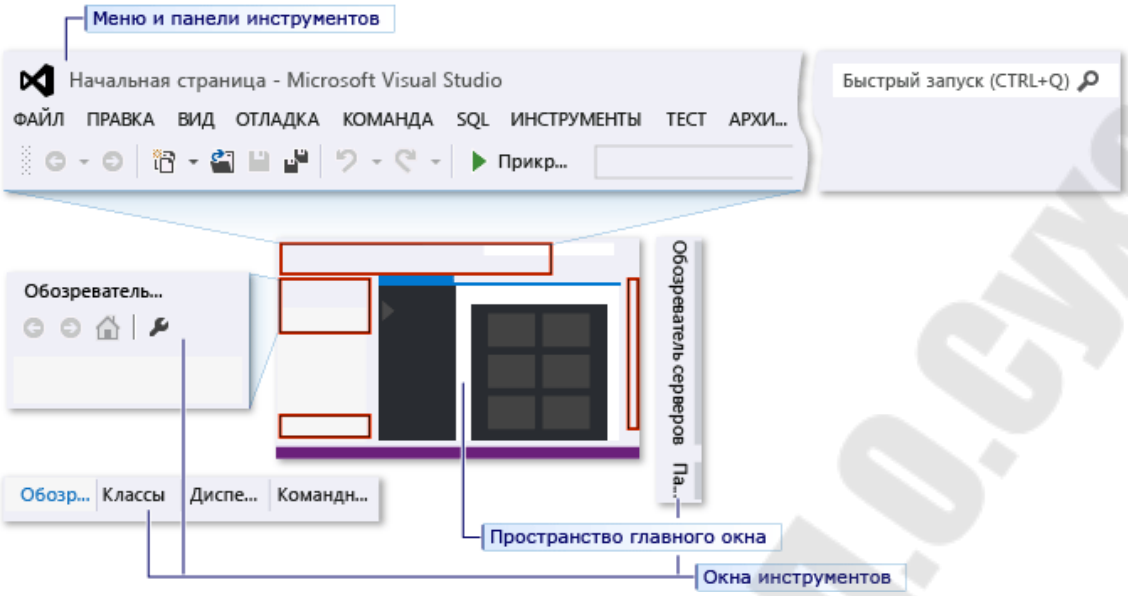

Рисунок 1.2 – Интегрированная среда разработки Visual Studio

Создание простого приложения

При создании приложения в Visual Studio необходимо сначала создать проект и решение .В этом примере создается консольное приложение Windows.

Создание консольного приложения

1. В строке меню выберите **Файл**, **Создать**, **Проект** (рисунок 1.3).

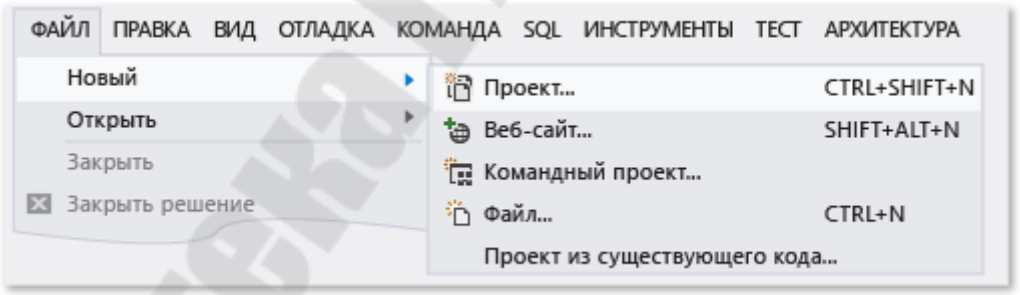

Рисунок 1.3 – Этапы создания проекта

2. В категории **Visual C++** выберите шаблон **Консольное приложение Win32** и назовите проект **GreetingsConsoleApp** (рисунок 1.4).

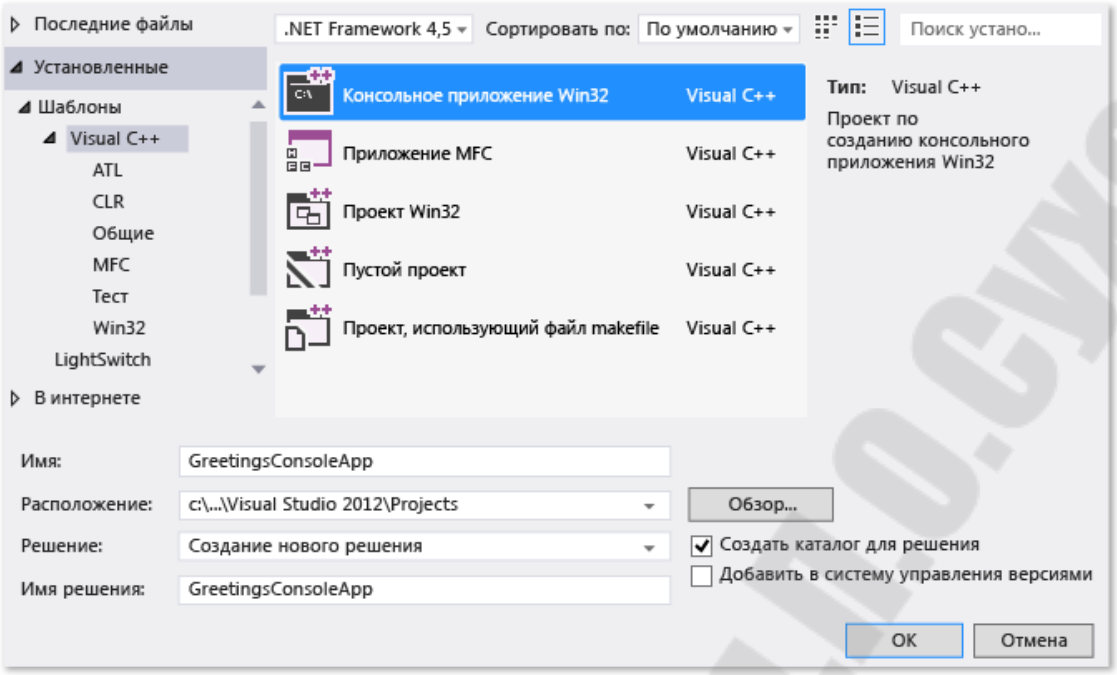

Рисунок 1.4 – Шаблоны приложений Visual Studio

3. Когда появится мастер приложений Win32, нажмите кнопку **Готово** (рисунок 1.5).

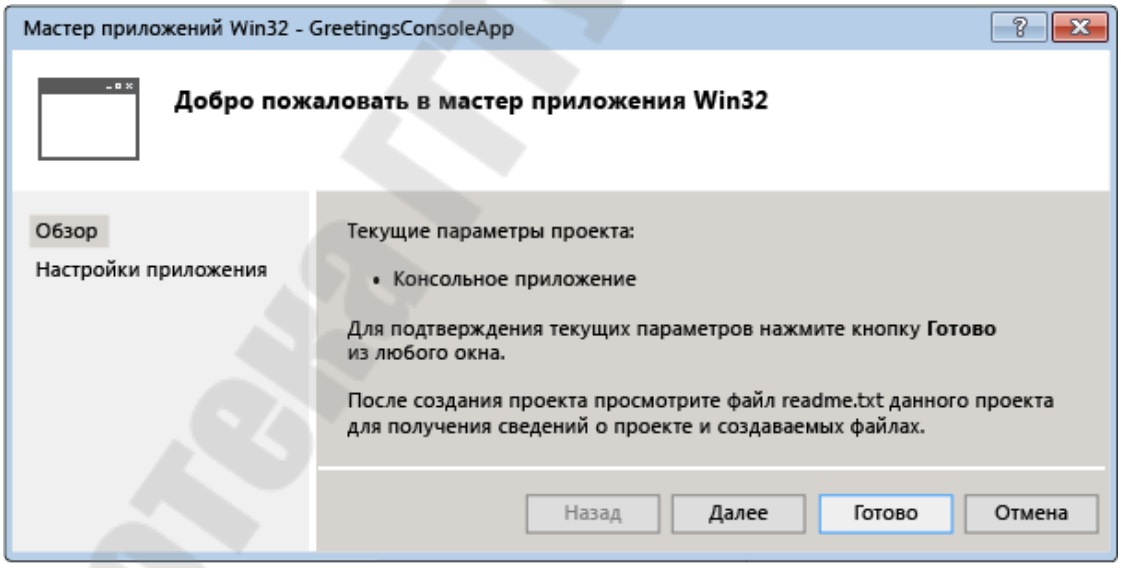

Рисунок 1.5 – Диалоговое окно мастера приложений Visual Studio

Проект GreetingsConsoleApp и решение с основными файлами для консольного приложения Win32 создадутся и автоматически загрузятся в **Обозреватель решений**.Файл GreettingsConsoleApp.cpp откроется в редакторе кода. Следующие элементы отображаются в **Обозревателе решений** (рисунок 1.6).

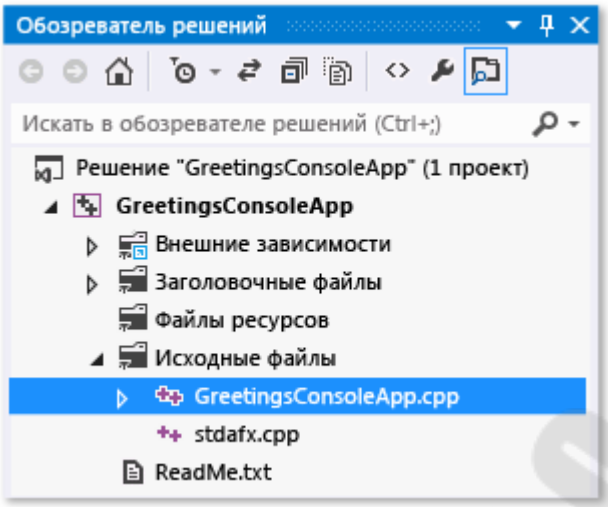

Рисунок 1.6 – Обозреватель решений проекта Visual Studio

Добавление кода в приложение

Далее необходимо добавить код для отображения слова "Hello" в окне консоли.

Отображение "Hello" в окне консоли

1. В файле GreetingsConsoleApp.cpp введите пустую строку перед строкой return 0;, а затем введите в нее следующий код:
2. Printf(“Hello\n”);

Красная волнистая линия появится под Printf. При наведении на нее отобразится сообщение об ошибке (рисунок 1.7).

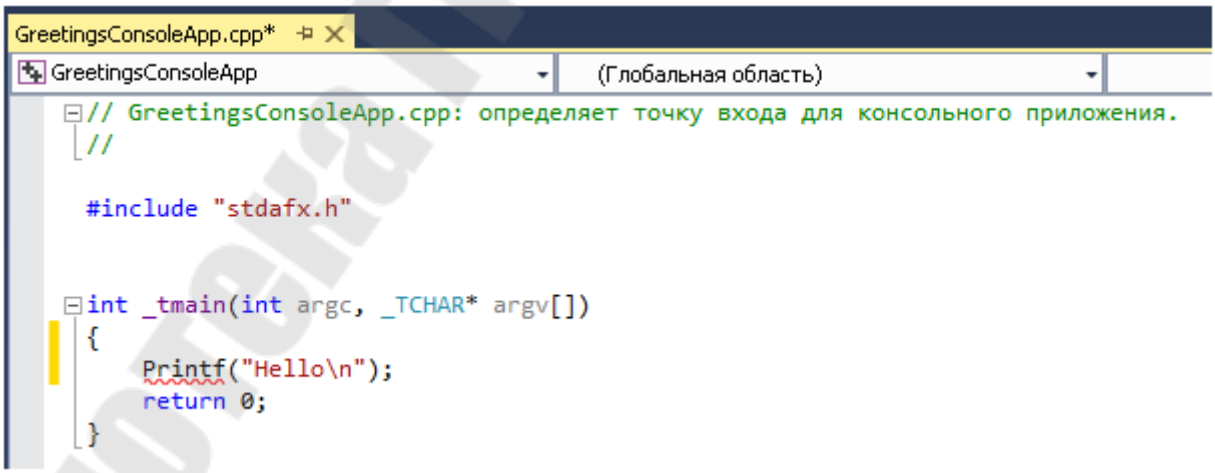

Рисунок 1.7 – Ошибка в тексте программы

Сообщение об ошибке также отобразится в окне **Список ошибок**. Можно отобразить это окно, выбрав в строке меню **Вид**, **Список ошибок**. В тексте программе была допущена ошибка по написанию функции printf.

Красная волнистая линия под Printf исчезнет после исправления ошибки (рисунок 1.8).

3. Сохраните изменения в файле.

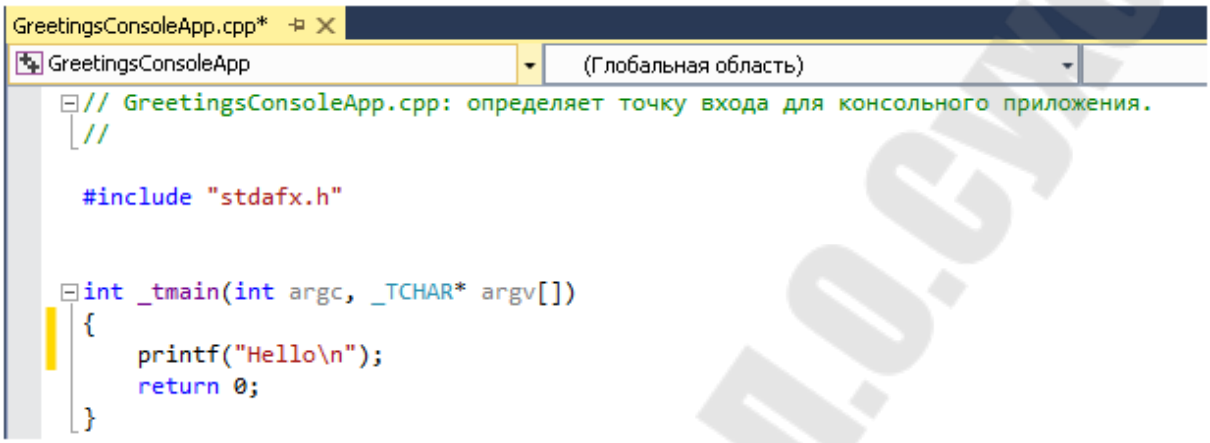

Рисунок 1.8 – Текстовый редактор среды разработки Visual Studio

Отладка и тестирование приложения

С помощью отладки GreetingsConsoleApp можно посмотреть, отображается ли слово Hello в окне консоли.

Отладка приложения

Запустите отладчик (рисунок 1.9).

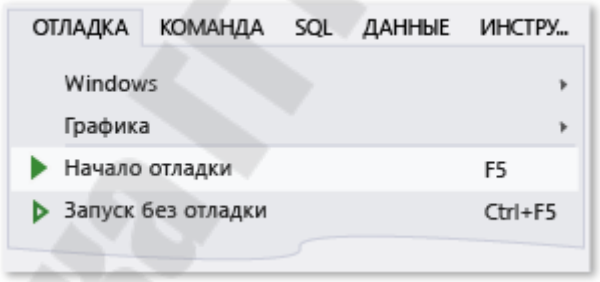

Рисунок 1.9 – Пункт меню Отладка среды разработки Visual Studio

Отладчик запустится и выполнит код.Окно консоли (отдельное окно, подобное командной строке) отображается в течение нескольких секунд, но при остановке отладчика быстро закрывается.Чтобы просмотреть текст, необходимо установить точку останова выполнения программы.

Добавление точки останова

1. Добавьте точку останова из меню в строке return 0;.Для установки точки останова можно также просто щелкнуть область слева (рисунок 1.10).

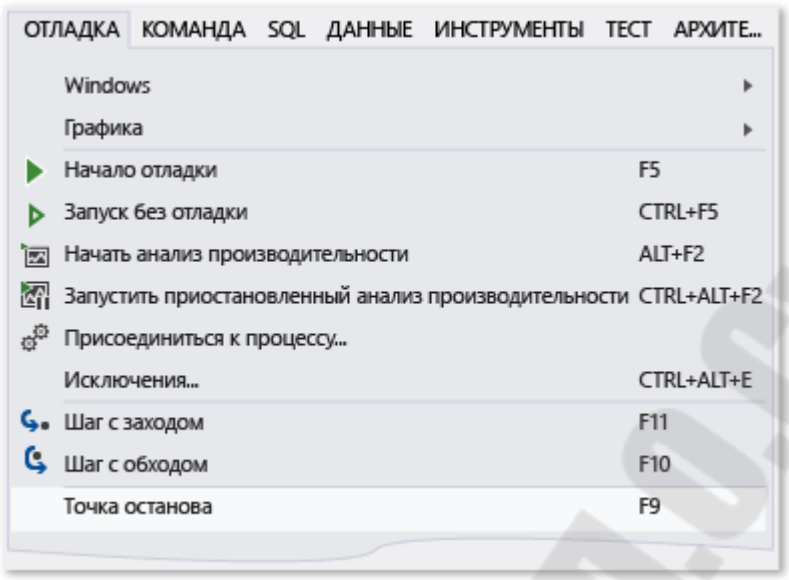

Рисунок 1.10 – Добавление точки останова

Красный кружок появится рядом с строкой кода с краю левого поля окна редактора.

2. Нажмите клавишу F5, чтобы начать отладку.

Запускается отладчик, и появляется окно консоли, отображающее слово Hello (рисунок 1.11).

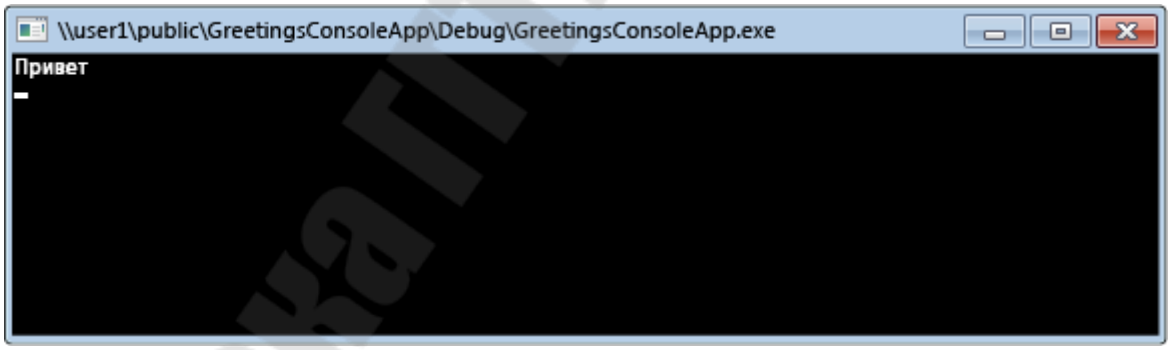

Рисунок 1.11 – Результаты работы программы

3. Нажмите SHIFT+F5 для остановки процесса отладки.

Для получения дополнительной информации см. Подготовка к отладке: консольные проекты.

Сборка окончательной версии приложения

Теперь, когда проверено, что все работает, можно подготовить окончательную сборку приложения.

Очистка файлов решения и сборка окончательной версии

1. При помощи команды главного меню, удалите промежуточные файлы, а также выходные файлы, созданные во время предыдущих построений (рисунок 1.12).

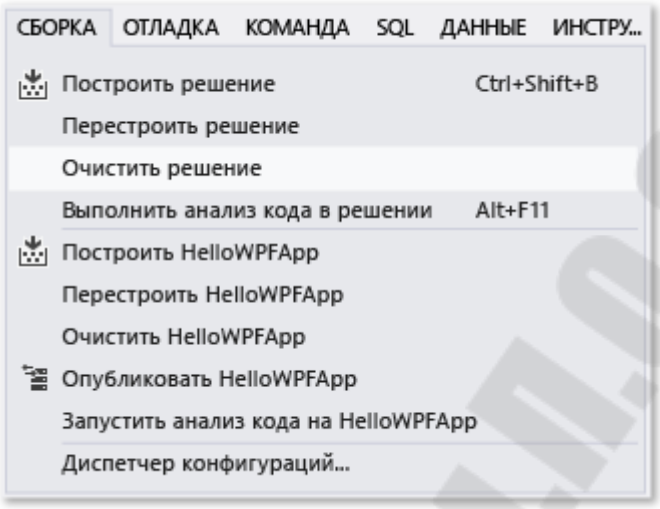

Рисунок 1.12 – Пункт меню Сборка среды разработки Visual Studio

2. Измените конфигурацию сборки для GreetingsConsoleApp с **Отладка** на **Выпуск** (рисунок 1.13).

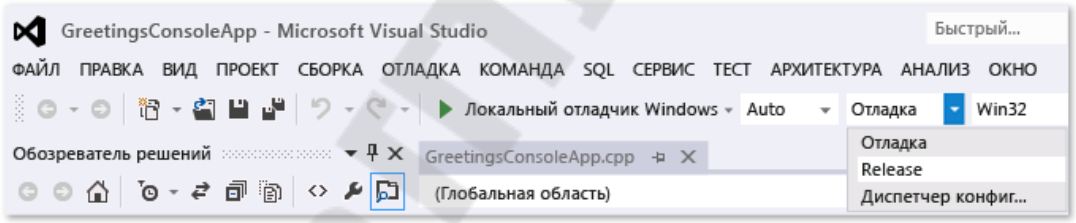

Рисунок 1.13 – Конфигурация сборки среды разработки Visual Studio

3. Постройте решение (рисунок 1.14).

math.h — заголовочный файл стандартной библиотеки языка программирования C, разработанный для выполнения простых математических операций. Большинство функций привлекают использование чисел с плавающей точкой. C++ также реализует данные функции для обеспечения совместимости, все они содержатся в заголовочном файле cmath.

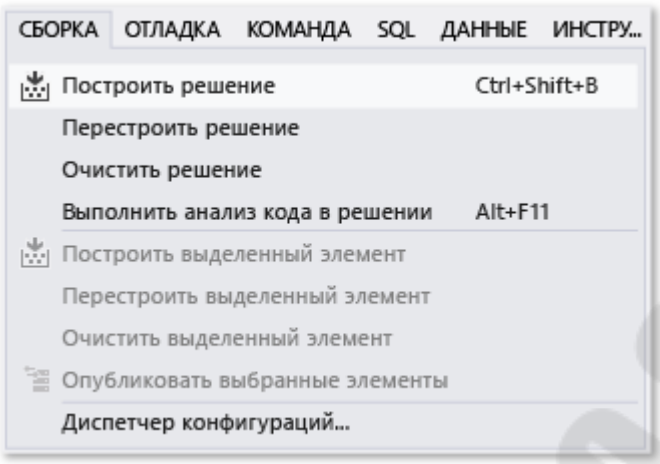

Рисунок 1.14 – Пункт меню Сборка среды разработки Visual Studio

Все эти функции принимают double, если не определено иначе. Для работы с типами float и long double используются функции с постфиксами f и l соответственно. Все функции, принимающие или возвращающие угол, работают с радианами.

Таблица 1.3 – Математические функции

Имя	**Описание**
1	**2**
abs	Возвращает абсолютную величину целого числа
acos	арккосинус
asin	арксинус
atan	арктангенс
atan2	арктангенс с двумя параметрами
ceil	округление до ближайшего большего целого числа
cos	косинус
random	выводит случайное число от 0 до аргумента функции.
exp	вычисление экспоненты
fabs	абсолютная величина (числа с плавающей точкой)
floor	округление до ближайшего меньшего целого числа
fmod	вычисление остатка от деления нацело для чисел с плавающей точкой
frexp	разбивает число с плавающей точкой на мантиссу и показатель степени.
ldexp	умножение числа с плавающей точкой на целую степень двух
log	натуральный логарифм
log10	логарифм по основанию 10

Продолжение таблицы 1.3

1	2
pow(*x*,*y*)	результат возведения *x* в степень *y*, x^y
sin	синус
sinh	гиперболический синус
sqrt	квадратный корень
tan	тангенс
tanh	гиперболический тангенс

Пример выполнения задачи.

Задание: Посчитать сумму двух чисел a и b.

Алгоритм решения задачи, представлен на рисунке 1.15.

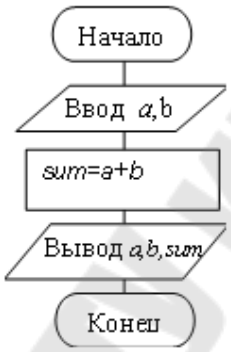

Рисунок 1.15 – Линейный алгоритм

Исходный код решения задачи представлен на рисунке 1.16.

```
#include "stdafx.h"
#include <iostream>
#define _CRT_SECURE_NO_WARNINGS
using namespace std;


int _tmain(int argc, _TCHAR* argv[])
{
    //русский алфавит
    setlocale(0,"");
    //объявление переменных
    int a, b, sum;
    printf("Введите значение переменной a  \n");
    scanf("%d", &a);
    printf("Введите значение переменной b  \n");
    scanf("%d", &b);
    sum = a + b;
    printf("Сумма элементов %d и %d равна %d \n",a,b,sum);
    return 0;
}
```

Рисунок 1.16 – Исходный код решения задачи

Для получения результата нажимает клавиши Ctrl+F5.
Результат работы программы, показан на рисунке 1.17.

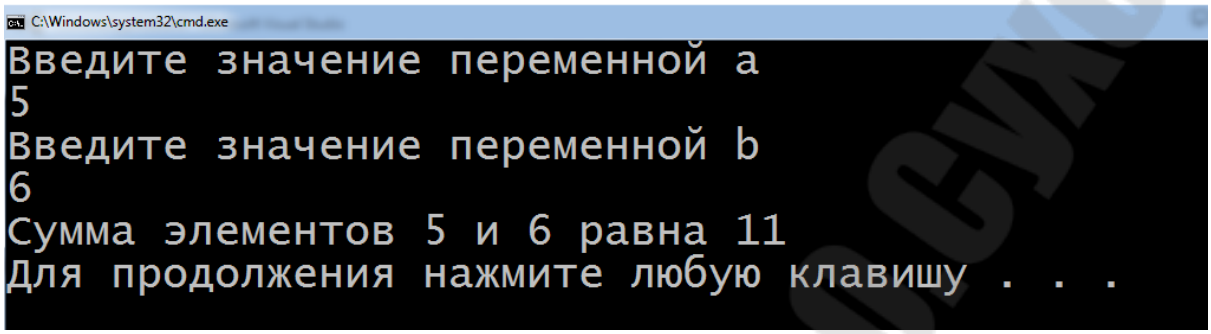

Рисунок 1.17 – Результат решения задачи

Задание

Разработать программу для вычисления арифметического выражения и вывода полученного результата, согласно варианта в таблице 1.4.

Таблица 1.4 – Варианты индивидуальных заданий

№	Выражение	Данные
1	**2**	**3**
1	$a = \ln(y^{-\sqrt{\lvert x\rvert}})(\sin(x) + e^{(x+y)})$	x, y
2	$b = \sqrt{c(\sqrt{y} + x^2)}(\cos(x) - \lvert c - y\rvert)$	c, x, y
3	$c = arctg(x) - \frac{3}{5}e^{xy} + 0.5\frac{\lvert x+y\rvert}{(x+y)^b}$	b, x, y
4	$d = \frac{e^{\lvert x-y\rvert} \cdot tg(z)}{arctg(y) + \sqrt{x}} + \ln(x)$	x, y, z
5	$e = \frac{(\cos(x) - \sin(y))^3}{\sqrt{tg(z)}} + \ln(xyz)$	x, y, z
6	$f = y^x + \sqrt{\lvert x\rvert + e^y} - \frac{z^3 \sin^2(y)}{y + z^2 / (y - x)}$	x, y, z
7	$g = \frac{1 + \cos(x+y)}{\lvert e^x - 2y / (1 + x^2 \cdot y^2)\rvert} \cdot x^3 + \arcsin(y)$	x, y

Продолжение таблицы 1.4

1	2	3
8	$h = 2 + \frac{x^2}{\sqrt{2}} + \frac{\lvert y^3 \rvert}{\sqrt{2}} + \frac{z^4 \cdot (\ln(x)+1)\sqrt{2}}{\sqrt{3}}$	x, y, z
9	$j = ((1+y)\sqrt{\sin^2(z)} - \frac{\lvert y-x \rvert}{5})^3$	x, y, z
10	$k = \ln\left\lvert (y - \sqrt{\lvert x \rvert})(x - \frac{y}{z + x^2/4}\right\rvert$	x, y, z
11	$l = 0.5x^5 + 3\cos(x+y) + e^{-0.1yz} - \sqrt{\lvert xy \rvert}$	x, y, z
12	$m = \sqrt{\lvert -3tg(x)\lg(x^4+y)/e^{-x}+1 \rvert}$	x, y
13	$n = \sqrt{e^x + tg(x) + 1} \cdot (\lg(y) + \cos(xy) + \sqrt[3]{x})$	x, y
14	$p = \frac{\lg(x) - e^{x+y}}{\sqrt{2} + y^2 + \lvert x^3 - \ln(y) \rvert}$	x, y
15	$q = \sqrt{12x^4 - 3x^2 + 4x^2 - 5x + 6} - \lg^2(z)$	x, y
16	$r = \lg\lvert 1 - 2x + 3x^2 - 4x^3 \rvert + \sqrt{\lvert x \rvert} / z$	x, y
17	$s = \frac{2\cos(x - 1/6)}{1/2 + \sin^2(y)} - \frac{1}{\lvert x^2/(y+x^3) \rvert}$	x, y
18	$t = \frac{x \cdot y \cdot z - y \cdot \lvert x + \sqrt{z} \rvert}{10^7 + \sqrt[4]{\lg(4)}}$	x, y, z
19	$u = \frac{(x+y-z)^3 - (x-y+z)^2 + \sqrt{\lvert x+y+z \rvert}}{\log_2(tg(2))}$	x, y, z

Окончание таблицы 1.4

1	2	3
20	$w = \dfrac{(x / y)(z + x)e^{\lvert x-y \rvert} + \ln(1+e)}{\sin^2(y) - (\sin(x) \cdot \sin(y))^2}$	x, y, z
21	$z = \sqrt{\dfrac{\lg^2(x^2+3)}{\sin 4x + 0.6a}}$	x, a
22	$p = arctg\left(\dfrac{x^3}{\sqrt{\ln x^2 + \cos^2 x}}\right)$	x
23	$b = \ln^2(y+a) + \dfrac{y - z / \sin x^2}{e^{-x} + \cos(x+y)^2}$	x, y, z, a
24	$Z = \exp\sqrt[8]{\dfrac{x - a\sin b}{ax^3 + b - c^7}}$	a, b, c
25	$a = \sqrt[3]{x + \sqrt[4]{\lvert y - 5 \rvert}} + tg^2 xy^3$	x
26	$Z = arctg(0.03x) - \dfrac{x^{2.5x} + 3\ln x}{\sqrt[3]{e^{2x} + x}}$	x
27	$y = e^{\sin x + \cos x} + \lg\sqrt{\lvert x^3 / (\cos x + 1) \rvert}$	x
28	$y = \cos\dfrac{x^3 + \sqrt{tgx}}{e^x} - \dfrac{1}{2.1^x}$	x
29	$y = \sin 9x + \operatorname{cosec} x^{1/7} + \dfrac{1}{x^2 + ab}$	x, a, b
30	$y = \sin(\lg(x^{2\cos\lvert\sqrt{e^x+5}-8\rvert}))$	x

Лабораторная работа № 2

Тема работы: Программирование задач, содержащих алгоритмы разветвляющихся структур.

Теоретические сведения по изучению темы:

1. Сущность алгоритма разветвляющейся структуры.
2. Условный оператор if – else.
3. Переключатель switch.

Изложение вопросов:

1. Под алгоритмом разветвляющейся структуры понимают алгоритм, в котором направление процесса выбирается в зависимости от выполнения или невыполнения того или иного условия.

2. Условный оператор дает возможность осуществлять разветвление выполнения программы. В качестве условного оператора в С используется конструкция if – else. Обычно используется три формы записи конструкции if – else.

Форма записи 1.

```
If  (условное выражение)      Оператор  выполняется
        оператор;             если выражение истинно.
```

Форма записи 2.

```
If  (условное выражение)      Если выражение истинно,
        оператор 1;           то выполняется оператор 1,
else                          в противном случае оператор 2
        оператор 2;
```

Форма записи 3.

```
If (условное выражение 1)     Если выражение истинно, то
        оператор 1;           выполняется оператор 1.
 else                         Если выражение 1 ложно, но
 If (условное выражение 2)    выражение 2 истинно,
        оператор 2;           выполняется в случае, когда
 else                         оба выражения ложны,
        оператор 3;           выполняется оператор 3.
```

В каждой из этих трех форм оператором может быть либо простой оператор, либо составной. Рассмотрим следующий пример:

```
If (legs =  = 4)
   printf ("Это возможно лошадь. \n");
else
   If (legs > 4)
      printf ("Это не лошадь.\n");
else
   {  legs ++;
      printf ("Теперь животное имеет еще одну ногу.\n");
   }
```

3.Часто в программе необходимо произвести выбор одного из нескольких вариантов. Это возможно осуществить, используя конструкцию if – else. Но во многих случаях это удобнее сделать с помощью оператора switch (переключателя).

Оператор switch имеет следующий общий вид:

```
switch (выражение)
{
  case 1 : оператор 1;
   break;
  case 2 : оператор 2;
  break;
    . . .
  case n : оператор n;
  break;
  default : оператор;
  break;
}
```

Порядок работы:

1. Определяется значение выражения;
2. Затем управление передается оператору, у которого в качестве метки используется значение вычислительного выражения;
3. Осуществляется выход из выбранного case по break и соответственно со всего switch, либо осуществляется “провал” на следующий case если не предусмотрен break;
4. Если значения не совпало ни с одним из case, то при наличии метки default выполняется оператор, помеченный этой меткой; если default отсутствует, то происходит переход к оператору, расположенному за оператором switch.

Замечание:

1.Выражение должно иметь значение целого типа (включая тип char).

2.Метки должны быть константами или константными выражениями.

3.Присутствие default необязательно, но является хорошим стилем программирования.

4.Наличие break во всех случаях case – хороший стиль программирования.

Пример использования оператора switch, приведен на рисунке 2.1.

```
#include "stdafx.h"
#include <iostream>
#include <math.h>

using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
        int c, i, nwhite, nother, ndigit[10];
        nwhite = nother = 0;
        for (i = 0; i<10; i++)
            ndigit[i] = 0;
        while (((c = getchar())) != EOF)
        switch (c)
        {
        case'0':
        case'1' :
        case'2' :
        case'3' :
        case'4' :
        case'5' :
        case'6' :
        case'7' :
        case'8' :
        case'9' : ndigit[c - '0']++; break;
        case' ' :
        case'\n' :
        case'\t': nwhite++; break;
        default: nother++; break;
        }

printf("Цифры = ");
for (i = 0; i<10; i++)
    printf("%d", ndigit[i]);
printf("пробелы = %d, прочие = %d\n", nwhite, nother);
return 0;
}
```

Рисунок 2.1 – Исходный код задачи

Пример выполнения задачи.

Задание: На числовой оси расположены три точки: A, B, C. Определить, какая из двух последних точек (B или C) расположена ближе к A, и вывести эту точку и ее расстояние от точки A.

Алгоритм решения задачи, представлен на рисунке 2.2.

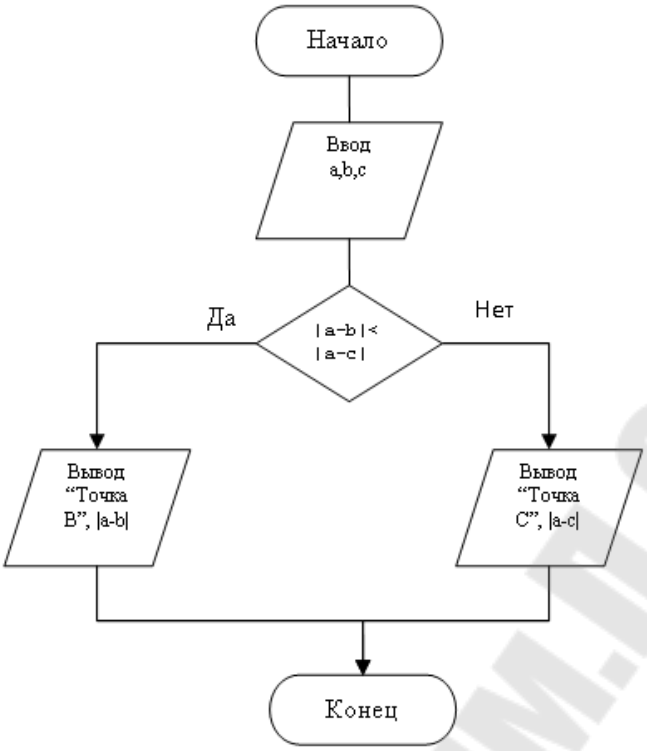

Рисунок 2.2 – Разветвляющийся алгоритм решения задачи

Исходный код решения задачи представлен на рисунке 2.3.

```
#include "stdafx.h"
#include <iostream>
#include <math.h>

using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    //русский алфавит
    setlocale(0,"");
    //объявление переменных
    float a, b, c;

    printf("Введите координату точки A:");
    scanf("%f", &a);
    printf("Введите координату точки B:");
    scanf("%f", &b);
    printf("Введите координату точки C:");
    scanf("%f", &c);
    if (fabs(a - b)<fabs(a - c))
        // Ближе точка B
        printf("Ближе к точке A расположена точка B. Расстояние от A до B =%5.2f", fabs(a - b));
    else
        // Ближе точка C
        printf("Ближе к точке A расположена точка C. Расстояние от A до C =%5.2f", fabs(a - c));

    return 0;
}
```

Рисунок 2.3 – Исходный код решения задачи

Для получения результата нажимает клавиши Ctrl+F5.
Результат работы программы, представлен на рисунке 2.4.

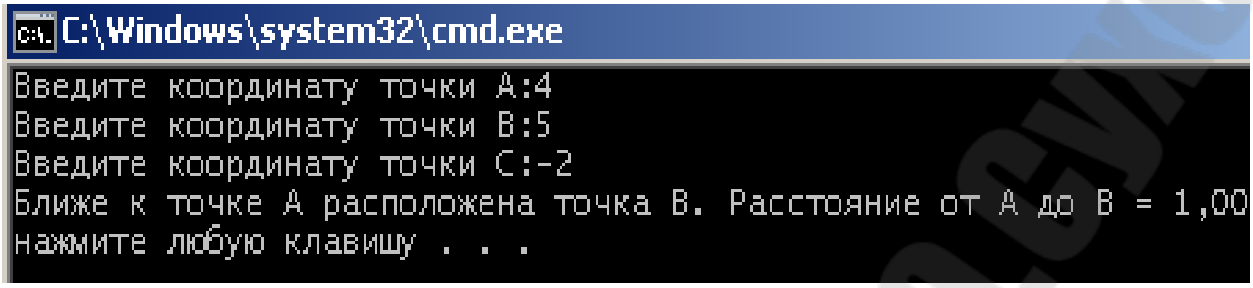

Рисунок 2.4 – Результат решения задачи

Задание

Разработать программу вычисления выражения и вывода полученного результата (на экран). Соответствующие исходные данные ввести с клавиатуры., согласно варианта в таблице 2.1.

Таблица 2.1 – Варианты индивидуальных заданий

№ п/п	Выражение	Исходные данные
1	**2**	**3**
1	$a=\begin{cases}(x+y)^2-\sqrt{xy}, & xy>0\\ (x+y)^2+\sqrt{\lvert x\cdot y\rvert}, & xy<0\\ (x+y)^2+1, & x=0\end{cases}$	x, y
2	$b=\begin{cases}\ln(x/y)+(x^2+y\ )^3, & x/y>0\\ \ln\lvert x/y\rvert+(x^2+y)^3, & x/y<0\\ (x^2+y)^3, & x=0\\ 0 & y=0\end{cases}$	x, y
3	$c=\begin{cases}x^2+y^2+\sin x, & x-y=0\\ (x-y)^2+\cos x, & x-y>0\\ (y-x)^2+tgx, & x-y<0\end{cases}$	x, y
4	$d=\begin{cases}(x-y)^3+arctgx, & x>y\\ (y-x)^3+arctgx, & y>x\\ (y+x)^3+0{,}5, & y=x\end{cases}$	x, y

Продолжение таблицы 2.1

1	2	3
5	$e=\begin{cases} i\sqrt{a}, & i-\text{нечетное}, a>0 \\ i/2\sqrt{\lvert a\rvert}, & i-\text{четное}, a<0 \\ \sqrt{\lvert ia\rvert}, & \text{иначе} \end{cases}$	i, a
6	$e=\begin{cases} e^{\lvert a\rvert-\lvert b\rvert}, & 0{,}5<ab<10 \\ \sqrt{\lvert a+b\rvert}, & 0{,}1<ab<0{,}5 \\ 2x^2, & \text{иначе} \end{cases}$	a,b,x
7	$h=\begin{cases} arctg(x-\lvert y\rvert), & x<y \\ arctg(\lvert x\rvert+y), & x>y \\ (x+y)^2, & x=y \end{cases}$	x, y
8	$j=\begin{cases} \sin(5k+3m\lvert k\rvert), & -1<k<m \\ \cos(5k+3m\lvert k\rvert), & y>x \\ k^3, & k=m \end{cases}$	k,m
9	$l=\begin{cases} 3k^3+3p^2, & k>\lvert p\rvert \\ \lvert k-p\rvert, & 3<k<\lvert p\rvert \\ (k+p)^2, & k=\lvert p\rvert \end{cases}$	k,p
10	$k=\begin{cases} \ln(\lvert f\rvert+\lvert q\rvert), & \lvert fq\rvert>10 \\ e^{f+q}, & \lvert fq\rvert<10 \\ f+q, & \lvert fq\rvert=10 \end{cases}$	f,q
11	$m=\dfrac{\max(x,y,z)}{\min(x,y)}+5$	x,y,z
12	$n=\dfrac{\min(x+y,y-z)}{\max(x,y)}$	x,y,z
13	$p=\dfrac{\lvert\min(x,y)-\max(y,z)\rvert}{2}$	x,y,z
14	$q=\dfrac{\max(x+y+z,xyz)}{\min(x+y+z,xyz)}$	x,y,z
15	$r=\dfrac{\max(\min(x,y),z)}{3}$	x,y,z

Продолжение таблицы 2.1

1	2	3
16	$s = \dfrac{\min(\max(x,y),\max(y,z))}{\max(y,z)}$	x,y,z
17	$t = \dfrac{\max(\min(x,5),\max(y,0))}{5}$	x,y
18	$v = \max(\min(x-y, y-x),0)$	x,y
19	$w = \max^2(\max(xy, x+y),0)$	x,y
20	$z = \dfrac{\min(0,x)-\min(0,y)}{\max^2(y,x)}$	x,y
21	$y = \begin{cases} 1-3x, & x>0, x<>8 \\ x^2-\sin x, & x<-1 \\ \cos x, & \textit{иначе} \end{cases}$	x=-4,82 x=7,12 x—0,36
22	$y = \begin{cases} \cos x^2, & 2{,}16 > x > 0{,}98 \\ \sin\dfrac{x}{2}, & x \le 0{,}98 \\ tg^2 x, & \textit{иначе} \end{cases}$	x=0,98 x=2,16 x=4,07
23	$y = \begin{cases} e^x, & x<0 \textit{ или } 3<x<4 \\ \ln^2 x, & x>4 \\ e^{\sqrt{x}}, & \textit{иначе} \end{cases}$	x=25,51 x=-2,17 x=1,87 x=3,6
24	$y = \begin{cases} x^3+1, & x \le 0, x \ne -10 \\ \sqrt[3]{x+5}, & x>1 \\ tgx, & \textit{иначе} \end{cases}$	x=-5,18 x=0,19 x=21,13
25	$y = \begin{cases} \sqrt{x+1}, & 0 \le x < 3 \\ 3x^2-2, & x>5, x \ne 7 \\ \dfrac{1}{x}+4, & \textit{иначе} \end{cases}$	x=2,57 x=-0,89 x=8,54
26	$y = \begin{cases} arctx^2, & x-2<0 \\ 3x, & x-2>13 \\ x+5, & \textit{иначе} \end{cases}$	x=4,81 x=20,32 x=-1,75

Окончание таблицы 2.1

1	2	3
27	$y=\begin{cases}\ln\left\|\frac{2}{x}\right\|, & x>-5, x\neq 0\\ 2x^2, & x<-5 \ \ или \ \ x=10\\ \|x\|+2, & иначе\end{cases}$	x=0,18 x=-5 x=-7,79
28	$y=\begin{cases}\ln x^2+\frac{x}{2}, & x>25, x=0{,}7\\ \sqrt{x}, & 1<x<25\\ \frac{x^2}{2}, & иначе\end{cases}$	x=10,18 x=-0,17 x=36,35
29	$y=\begin{cases}\sqrt[3]{x}, & x>2\\ \frac{1}{x}, & x<2, x\neq 0\\ \sqrt[6]{x}, & иначе\end{cases}$	x=5,63 x=-7,1 x=2
30	$y=\begin{cases}\frac{1}{x}, & x>-5, x\neq 0\\ x^2, & x<-10\\ \sqrt{\|x\|+1}, & иначе\end{cases}$	x=-18,76 x=-3,57 x=-7,15

Примечание. Здесь min, max – операции нахождения минимального и максимального из перечисленных в скобках значений элементов.

Лабораторная работа № 3

Тема работы: Программирование циклических структур.

Теоретические сведения по изучению темы:

Алгоритм циклических структур называется такой алгоритм, который повторяет какое-то действие до тех пор, пока верно некоторое условие.

В языке Си существует три оператора цикла- while, for и do-while.

Оператор цикла while имеет следующий синтаксис:

While(выражение)

<оператор;>

Оператор цикла while позволяет выполнять оператор до тех пор, пока значение выражения неравно нулю.

Оператор while является оператором цикла с предусловием, т.е. в начале вычисляется выражение, а затем выполняется оператор. Если выражение ложно, т.е. равно нулю – оператор ни разу не выполняется. Перед каждым следующим выполнением оператора выражение выполняется заново.

Если в теле цикла присутствует оператор break, то при выполнении этого оператора происходит выход из цикла. Если в теле цикла имеется оператор-контейнер (continue), то при выполнении этого оператора начинается выполняться следующая итерация.

Цикл for имеет следующий синтаксис:

For ([инициирующее выр-е]; [условие]; [выр-е приращение])

<оператор;>

Оператор for выполняется следующим образом:

1) Вначале вычисляется инициирующее выражение. Если инициирующее выражение отсутствие, то никаких действий не выполняется.

2) Вычисляется выражение условие. Если оно истинно, то переходит к шагу 3, если значение выражения равно нулю, то управление передается на следующий за for оператор. Если условие отсутствует, то считается, что условное выражение истинно.

3) Выполняется тело цикла оператора for. Если в теле цикла присутствует оператор разрыва break, то цикл завершает вне зависимости от условного выражения. Если в теле цикла встречается оператор continue, то управление сразу передается на шаг 4, причем если оператор составной, то все операторы от continue, до конца тела цикла не выполняются.

4) Вычисляется выражение приращения, и затем переходит к шагу 2.

Цикл вида for (;;);- является бесконечным циклом.

В инициализирующем выражении и выражении приращения можно указывать несколько выражений, разделяя их запятой.

Цикл do while имеет следующий синтаксис:

do

<оператор>

while (<выражение>);

Тело оператора цикла do while выполняется один или несколько раз до тех пор, пока значение <выражения> не станет ложным (равным нулю). Вначале выполняется тело цикла - <оператор>, затем вычисляется условие – <выражение>. Если выражение ложно, то оператор цикла do while завершается и управление передается следующему за оператором while оператору программы. Если значение выражения истинно (не равно нулю), то тело цикла выполняется снова, и снова вычисляется выражение. Выполнение тела оператора цикла do while повторяется до тех пор, пока выражение не станет ложным. Оператор do while может также завершиться при выполнении в своем теле операторов break, goto, return.

Пример выполнения задачи.

Задание: Написать программу для вычисления значения функции Y

Интервал изменения аргумента [-1;15] шаг изменения аргумента 1

$$y = \begin{cases} 1 + x, & x > 14.5 \\ e^{-x}, & 3 \le x \le 14.5 \\ \cos(x), & x < 3 \end{cases}$$

Алгоритм решения задачи, представлен на рисунке 3.1.

Исходный код решения задачи представлен на рисунке 3.2.

Для получения результата нажимает клавиши Ctrl+F5.

Результат работы программы представлен на рисунке 3.3.

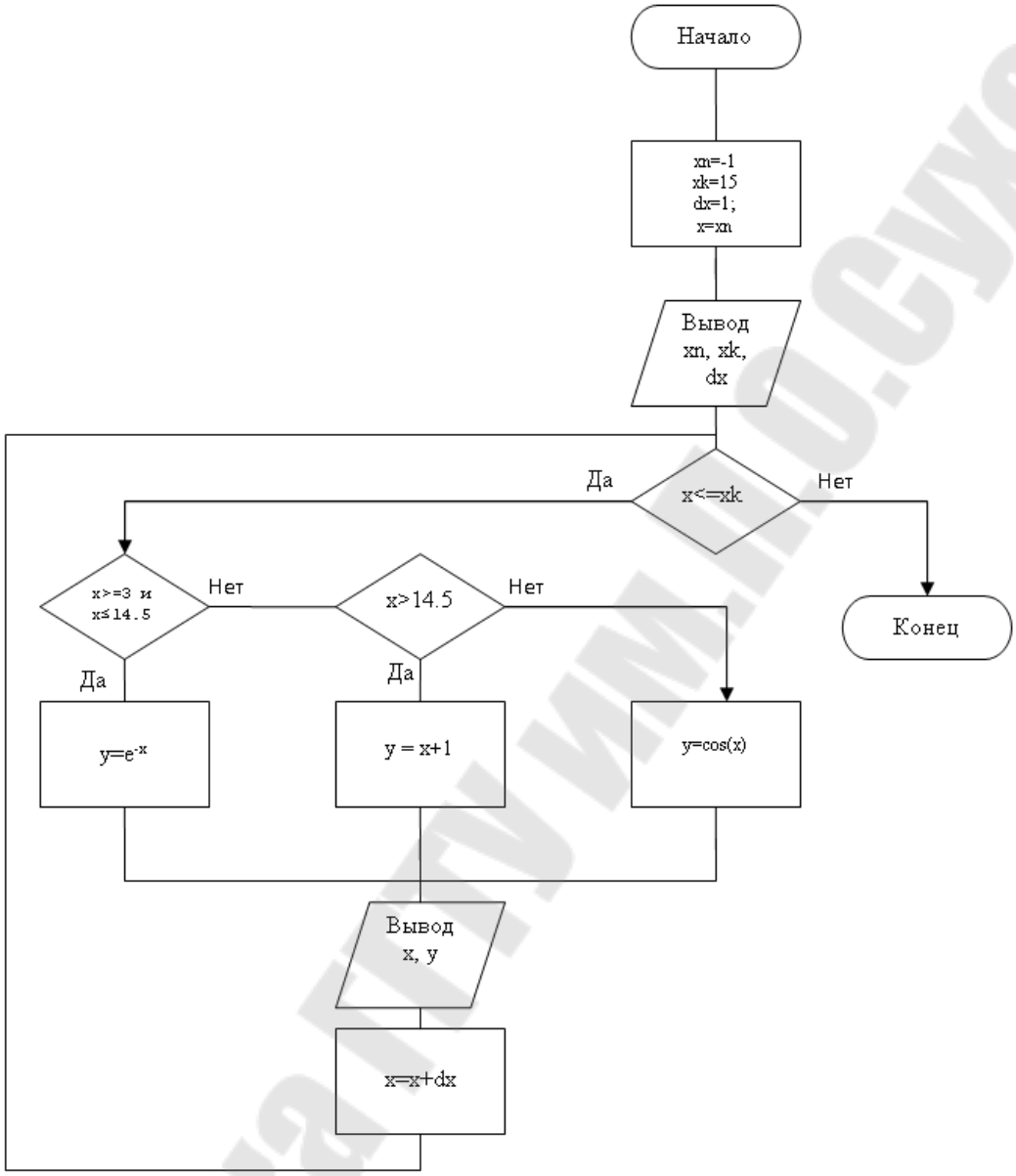

Рисунок 3.1 – Циклический алгоритм решения задачи

```
#include "stdafx.h"
#include <iostream>
#include <math.h>

using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    //русский алфавит
    setlocale(0,"");
    //объявление переменных
    float x, xn = -1, xk = 15, dx = 1, y;

    printf("Начальное значение x=%5.1f\n", xn);
    printf("Конечное значение x=%5.1f\n", xk);
    printf("Шаг x=%f\n", dx);
    printf("---------------------------\n");
    printf("|    X    |     Y=f(x)    |\n");
    printf("---------------------------\n");
    for (x = xn; x <= xk; x += dx)
    {
        if (x >= 3 && x <= 14.5)
            y = exp(-x);
        else
            if (x>14.5)
                y = 1 + x;
            else
                y = cos(x);
        printf("|%8.2f|%13.3f   |\n", x, y);
    }
    printf("---------------------------\n");


    return 0;
}
```

Рисунок 3.2 – Исходный код решения задачи

```
C:\Windows\system32\cmd.exe
Начальное значение x= -1,0
Конечное значение x= 15,0
Шаг x=1,000000
---------------------------
|    X    |     Y=f(x)    |
---------------------------
|   -1,00|        0,540   |
|    0,00|        1,000   |
|    1,00|        0,540   |
|    2,00|       -0,416   |
|    3,00|        0,050   |
|    4,00|        0,018   |
|    5,00|        0,007   |
|    6,00|        0,002   |
|    7,00|        0,001   |
|    8,00|        0,000   |
|    9,00|        0,000   |
|   10,00|        0,000   |
|   11,00|        0,000   |
|   12,00|        0,000   |
|   13,00|        0,000   |
|   14,00|        0,000   |
|   15,00|       16,000   |
---------------------------
Для продолжения нажмите любую клавишу . . .
```

Рисунок 3.3 – Результат решения задачи

Задание

I. Разработать программу табулирования функции, согласно задания в лабораторной работе 2.

II. Разработать программу для вычисления арифметического выражения и вывода полученного результата, согласно варианта. Разработать программу вычисления выражения и вывода полученного результата (на экран). Соответствующие исходные данные ввести с клавиатуры.

1.Вычислить число сочетаний из n и m по формуле $C_m^n = \frac{n!}{m!(n-m)!}$, где n!=1*2*3*...*(n-1)*n , целые числа n , m(n>=m>0) ввести с клавиатуры.

2. Вычислить значение выражения $b=(1+\frac{1}{2}+\frac{1}{2^2}+\frac{1}{2^3}+...+\frac{1}{2^n})^n$, где n ввести с клавиатуры .

3. Вычислить число размещений n из m по формуле $A_n^m = n*(n+1)...(n-m+1)$, где целые числа m и n ввести с клавиатуры (n>=m>0).

4. Вычислить приближенно значение бесконечной суммы с точностью до E=0.001: $S=1-\frac{n}{2*3}+\left(\frac{n}{3*4}\right)^2+\left(\frac{n}{4*5}\right)^3+...+\left(\frac{n}{(k+1)*(k+2)}\right)^k+...$

Значение n и точность расчетов ввести с клавиатуры.

Примечание : Считать , что требуемая точность достигнута , если очередное слагаемое оказалось по модулю меньше , чем E .

5. Вычислить приближенное значение бесконечной суммы с точностью до 0,0005: $S=1-\left(\frac{n}{2*3*4}\right)^2+\left(\frac{n}{3*4*5}\right)^4+\left(\frac{n}{4*5*6}\right)^6+...+\left(\frac{n}{(k+1)*(k+2)*(л+3)}\right)^{2k}+...$ значение n и точность ввести как константы (см. Примечание к варианту 4).

6.Дано натуральное n, определить количество цифр в числе n и сумму всех его цифр. Значение n ввести с клавиатуры.

7.Вывести на печать таблицу n значений функции $y=\sqrt{\left[a*x^2+b*x+c\right]}$ при изменении x от x_1 до x_2 с шагом $h=(x_2-x_1)/(n-1)$. Коэффициенты a, b,c, границы x_1, x_2 число n ввести с клавиатуры.

8. Вычислить значение функции $y=\frac{k}{x^k+y^{k-1}}$ для k=1,2,... . Вычисления производить до тех пор, пока y>=z .Исходные данные x , y и z (x , y>1) ввести с клавиатуры.

9.Найти приближенно , с точностью до 0,001 минимум функции $f(x)=ax^2+bx+c$ на отрезке $x_1 \le x \le x_2$.Значения x_1 ,x_2 и a ,b, c ввести как константы.

10. Пусть $a_1=u$; $b_1=v$; $a_k=2b_{k-1}+a_{k-1}$; $b_k=2a_{k-1}+b_{k-1}$; k=2,3,4... . Даны действительные u ,v , натуральное n. Найти $\sum_{k=1}^{n}\frac{a_k b_k}{(k+1)!}$.

11. Вычислить бесконечную сумму $\sum_{n=1}^{\infty}(-1)^n\frac{x^{2n}}{(2n)!}$ с точностью E=0,001. См. Примечание к варианту 4.

12. Вычислить бесконечную сумму $\sum_{i=1}^{\infty}\frac{(-1)^i}{(i+1)!}$ с точностью E=0,0001. См.примечание к варианту 4.

13. Вычислить и вывести на экран таблицу значений функции y=k/x!, больших e, если k=1,2,3... . Исходные данные :x=1,55; e=0,0183

14. Вычислить и вывести на экран таблицу всех 20 значений функции $z(i,j)=x_i^2+y_j^3$ для $x_i=x_1,x_2,x_3,x_4$; $y_j=y_1,y_2,...,y_5$. Исходные данные : $x_1=0,1;y_1=1,1;x_2=0,2;y_2=1,2;x_3=0,3;y_3=2,1;x_4=0,4;y_4=2,2;y_5=2,5$.

15. Вычислить значение выражения $(9+(9+...+(9+9^{1/2})^{1/3}...)^{1/m-1})^{1/m}$. Число m ввести с клавиатуры.

16.Составить программу для вычисления и вывода на экран таблицы значений суммы $S=\sum_{k=1}^{n}=(-1)^k\frac{(1+x)^{2k}}{k!}$ для a≤x≤b, изменяются с шагом h=(b-a)/10. Исходные данные : a=-1,05; b=362;n=10.

17.Вычислить и вывести на экран таблицу значений функции $y=x^2-ab-cx$ для x= -5,-6,-7,...,-35. Исходные данные (в экспоненциальной форме записи): a=1.14E+1; b= -4.21E+2; c=3.25E+3.

18. Составить программу для вычисления и вывода на экран таблицы сумм $S=\sum_{k=1}^{n}\frac{\cos(kx)}{k}$, где x изменяется в пределах a≤x≤b с шагом h=(b-a)/10. Исходные данные : a=0,1 ; b=13; n=12.

19. Вычислить и вывести на экран таблицу значений функции $y=x^2-ab+cx$, для $x_{нач}<=x<=x_{кон}$ с шагом h. Исходные данные : a= -1,14; b=-4,21; c=3,25; $x_{нач}$=4,5; $x_{кон}$= 35,5 ; h=0,5.

20. Найти наибольшее значение функции $y=ax^3+bx-c$, при изменении x от $x_{нач}$ до $x_{кон}$ с шагом h . Исходные данные : a=2,14; b= -4,21 ; c=3,25; $x_{нач}$= -4,5; $x_{кон}$= -35,5 ; h=-0,5.

21. Дано натуральное число N. Вычислить: $\sqrt{3+\sqrt{6+..+\sqrt{3(N-1)+\sqrt{3N}}}}$.

22. Даны натуральные числа N и K (N >=K >=0). Вычислить: $\frac{N(N-1)...(N-K-1)}{K!}$.

23. Дано натуральное число N, вещественное число X. Вычислить : $\sum_{i=1}^{N}(\frac{1}{i!}+\sqrt{[x]})$.

24. Вычислить : $\prod_{i=1}^{52}\frac{i^2}{i^2+2i+3}$;

25. Вычислить бесконечную сумму с заданной точностью E(E>0): $\sum_{i=0}^{\infty}\frac{1}{4^i+5^{i+2}}$

26. Население города ежегодно увеличивается на 1/40 наличного состава жителей. Через сколько лет население города утроится ?

27. Даны натуральное число N , вещественное число X. Вычислить : $\prod_{k=1}^{N}\frac{(1-X)^{k+1}+1}{((K-1)!+1)^2}$.

28. Дано натуральное число N. Вычислить : $\frac{1}{\sin 1}+\frac{1}{\sin 1+\sin 2}+...+\frac{1}{\sin 1+...+\sin N}$.

29. Дано натуральное число M. Получить наибольшее целое K , при котором 3^k<M.

30. Вычислить : $\frac{\cos 1}{\sin 1}*\frac{\cos 1+\cos 2}{\sin 1+\sin 2}*...*\frac{\cos 1+...+\cos n}{\sin 1+...+\sin n}$.

Лабораторная работа № 4

Тема работы: Программирование задач по работе с одномерными массивами при адресации элементов массива обычным способом и через указатели.

Теоретические сведения по изучению темы:

1. Указатели и адреса
2. Указатели и одномерные массивы.
3. Адресная арифметика или операции с указателями.

1.Рассмотрим упрощенную схему организации оперативной памяти машины. Память типичной машины представляет собой массив последовательно пронумерованных и проадресованных ячеек, с которыми можно работать по отдельности или связными кусками. Применительно к любой 16-разрядной или 32-разрядной машине верны следующие утверждения:

- один байт может хранить значение типа char;
- двухбайтовая ячейка может рассматриваться как целое типа short
- четырехбайтовая - как целое типа long

Указатель - это группа ячеек, в которых может храниться адрес (рисунок 3.1). Например, ch имеет тип char, p - указатель, ссылающийся на ch, то графическая ситуация выглядит, как показано на рисунке 4.1.

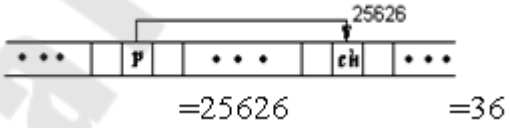

Рисунок 4.1 – Работа с указателями

В С имеется унарная операция & - операция получения адреса. Запись вида pch=&ch; присваивает адрес ячейки, где находится ch переменной pch, которая является переменной указателя. В этом случае принято говорить: pch указывает на ch, или, что одно и то же, pch ссылается на ch.

Указатель на переменную содержит адрес памяти расположения этой переменной.

Объявление указателя имеет следующее формальное описание:

тип_переменной *имя_переменной_адреса;

Инициализация указателя выполняется следующим образом:

тип_переменной имя_переменной_содержания;

имя_переменной_адреса =
&имя_переменной_содержания;

Объявление указателя может быть выполнено с одновременной инициализацией:

тип_переменной *имя_переменной_адреса =
&имя_переменной_содержания;

Доступ к значению переменной по указателю имеет следующее формальное описание: имя_переменной_содержания1= *имя_переменной_адреса;

При работе с указателями действуют следующие правила:

- при объявлении переменной-указателя перед именем переменной указывается операция *;
- если одним оператором объявляется несколько переменных-указателей, то перед каждой такой переменной следует указывать операцию *;
- после объявления указателя его следует инициализировать адресом значения того же типа, что и тип указателя;
- для получения адреса переменной перед ее именем указывается операция взятия адреса &;
- для получения значения переменной по указателю на нее перед указателем ставится операция разыменования * (называемая иногда операцией взятия значения);
- указатель строки содержит адрес первого символа строки;
- при увеличении указателя на единицу значение, содержащееся в переменной-указателе, увеличивается на число байт, которое отведено под переменную данного типа.

Правило: операция получения адреса & применяется только к объектам, расположенным в памяти: к переменным и элементам массивов. Её операндом не может быть ни выражение, ни константа, ни регистровая переменная.

Унарная операция * есть операция раскрытия ссылки. (другие названия: операция разадресации, операция косвенной разадресации, наиболее часто используется термин операция разадресации). Эта операция применяется к указателю и выдает обьект, на который данный указатель ссылается.

Приведем следующие записи:

```
int x=1, y=2, z[10];          /* описание переменных и массива*/
int pint;                     /* описание указателя на тип int  */
pint=&x;                      /* теперь pint указывает на x     */
y=*pint;                      /* y теперь равен x, то есть 1    */
*pint=0;                      /* x теперь обнуляется            */
pint=&z[0];                   /* теперь pint указывает на z[0]  */
```

Таким образом, описание переменной как указателя, аналогично описанию обычной переменной с добавлением символа (*) перед именем переменной. Приведем примеры правильного описания указателей:

```
int *pint;
char *pchar;
float *pfloat, *ptf;
double *pdb;
```

После описания указателю разрешено ссылаться только на обьекты заданного типа. Исключение составляет указатель на тип void, который может ссылаться на обьекты любого типа. Но к такому указателю нельзя применить оператор разадресации.

Замечание: При выводе на экран или печать значения указателя с помощью функции printf() необходимо пользоваться спецификацией преобразования %u, поскольку значением указателя является адрес - целое число без знака.

2. В языке С между массивами и указателями существует очень тесная связь, поэтому при изучении обычно их рассматривают вместе.

Обозначение массива представляет собой скрытую форму использования указателей. Имя массива определяет адрес его первого элемента, то есть если mas[] - есть массив, то mas ↔&mas[0]. Рассмотрим пример:

```
void main (void)
{
 int dates[4], *pti, index;
 float bills[4], *ptf;
 pti=dates;   /* присвоить адрес массива указателю */
 ptf=bills;
 for (index=0;index<4;index++)
  printf("Указатели +%d : %10u \n", index, pti+index, ptf+index);
}
```

Результат:

```
Указатели +0:     56014     86026
Указатели +1:     56016     86030
Указатели +2:     56018     86034
Указатели +3:     56020     86038
```

Первая строка содержит начальные адреса двух массивов. Следующая строка результат прибавления единицы к адресу и т. д. Вопрос: почему получается 56014+1=56016; 86026+1=86030.

3. Обычно в вычислительной системе единицей адресации является байт; однако тип int использует два байта, тип float - четыре. При прибавлении единицы к указателю компилятор генерирует код: добавить единицу памяти. Для массивов это означает, что осуществляется переход к адресу следующего элемента, а не следующего байта. Отсюда вывод: при

описании указателя обязательно указывается тип объекта, на который ссылается указатель, ибо машина должна знать, сколько байтов потребуется для запоминания объекта. Запишем следующие соотношения:

dates+2 ↔ &dates[2]; - один и тот же адрес

*(dates+2) ↔ dates[2]; - одно и то же значение

Эти соотношения суммируют теснейшую связь между массивами и указателями. Они показывают, что можно использовать указатель для определения отдельного элемента массива, и также для получения его значения.

Замечание: компилятор с языка С построен таким образом, что превращает обозначение массива в указатели, поэтому метод указателей более предпочтителен.

Например:

```
float fArray[3];  // Массив
float* pArray;
pArray=fArray;   // Эквивалентно оператору
                 // pArray=&fArray[0];
pArray++;        // Указывает на второй
                 // элемент массива
float* pArray2;
pArray2=&fArray[1]; // Указывает на второй
                    //элемент массива
```

Задание.

Задачи должны иметь два варианта решения. Первый вариант решения – обычная адресация элементов массива; второй вариант – адресация через указатели и использование функций.

1. Заданы натуральное число n и последовательность вещественных чисел $b_1, b_2, ..., b_n$. Сформировать одномерный массив А такой, что $a_j=1/(b_i-b_{i-1}), i$=2, 3, ...,n; $a_1=1/(b_1-b_n)$. Значение n и последовательность чисел $b_1, b_2, ..., b_n$ ввести с клавиатуры.

2. Заданы натуральное число n и последовательность вещественных чисел (результаты экспериментов) $a_1, a_2, ..., a_n$. Рассчитать их среднее арифметическое значение по формуле $S = (a_1, a_2, ..., a_n)/n$ и выборочную дисперсию по формуле $D = ((a_1 - S)^2 + (a\ 2 - S)^2 + . . . + (a_n — S)^2)/ (n- 1)$. Исходные данные ввести с клавиатуры.

3. Заданы целые положительные числа n, m и p. Определить и вывести на печать вектор $x=(x_1, x_2, ..., x_m)$, элементы которого

$$x_j = \begin{cases} n & при \quad g_j = 0 \\ g_j / n^{m-j} & при \quad g_j \neq 0 \end{cases}$$

$j=1,2,...,m$, где $g_i=p-g_{i-1}/n, g_0=p$

Исходные данные: n-3, m=4, p=46.

4. В массиве из 20 целых чисел наибольший элемент и поменять его местами с первым элементом.

5. В массиве из 10 целых чисел найти наименьший элемент и поменять его местами с последним элементом.

6. В массиве из 15 вещественных чисел найти наибольший элемент и поменять его местами с последним элементом.

7. В массиве из 25 вещественных чисел найти наименьший элемент и поменять его местами с первым элементом.

8. Упорядочить по неубыванию массив, содержащий 20 чисел.

9. Упорядочить по невозрастанию массив, содержащий 10 целых чисел.

10. Упорядочить по неубыванию массив, содержащий 15 вещественных чисел.

11. Упорядочить по невозрастанию массив, содержащий 25 вещественных чисел.

12. Дан массив целых чисел, содержащий 20 элементов, записать в этот же массив сначала все отрицательные числа и нули, затем все положительные, сохраняя порядок их следования.

13. Дан массив целых чисел, содержащий 10 элементов, записать в этот же массив сначала все положительные числа, а затем все отрицательные и нули, сохраняя порядок их следования.

14. Дан массив вещественных чисел, содержащий 15 элементов, записать в этот же массив сначала все отрицательные числа, а затем все положительные и нули, сохраняя порядок их следования.

15. Дан массив вещественных чисел, содержащий 25 элементов, записать в этот же массив сначала все положительные числа и нули, а затем все отрицательные, сохраняя порядок их следования.

16. Дан массив из N чисел. Подсчитать количество чисел, принадлежащих промежутку [a,b] и сумму чисел, стоящих на местах кратных 3.

17. Дан массив из N чисел. Подсчитать сумму чисел, меньших заданного D, и количество чисел, стоящих на четных местах и больших заданного C.

18. Дан массив из N чисел. Найти максимальное число и поменять его местами с последним элементом массива.

19. Дан массив из N чисел. Найти минимальное число и поменять его местами с предыдущим элементом массива.

20. Дан массив из N чисел. Найти максимальное число и поменять его местами с 6 элементом массива.

21. Дан массив из N чисел. Найти минимальное число и записать вместо него число N^2+N.

22. Дан массив из N чисел. Найти максимальное число и вместо него записать значение элемента N+2.

23. Даны 2 массив чисел произвольной длины. Сформировать третий массив из элементов обоих массивов, больших второго элемент первого массива, и положительных элементов второго массива.

24. Даны 2 массив чисел произвольной длины. Сформировать третий массив из отрицательных элементов первого массива и элементов обоих массивов, больших первого элемента второго массива.

25. Дан массив из N чисел. Поменять местами первый элемент массива с самым левым минимальным элементом, а затем поделить все элементы массива на минимальное значение.

26. Дан массив из N чисел. Поменять местами последний элемент массива с самым правым максимальным элементом, а затем умножить все элементы массива на заданное число.

27. Дан массив из N чисел. Инвертировать часть массива с первого элемента до самого правого минимального элемента.

28. Задан массив размера N. Поменять местами элемент с заданным номером и самый левый минимальный элемент, после чего все элементы, имеющий номера с первого по заданный, поделить на найденное минимальное значение.

29. Задан массив размера N. К каждому элементу с первого по самый левый максимальный добавить заданное число, после чего поменять местами второй элемент с найденным максимальным.

30 Даны целые числа A1…….A50. Получить сумму тех чисел последовательности, которые нечетны и отрицательны.

Лабораторная работа № 5

Тема работы: Программирование задач, содержащих в себе пользовательские функции

Теоретические сведения по изучению темы:

1. Общие сведения.
2. Определение, объявление и вызов функции.
3. Формальные параметры. Аргументы. Возвращение значения. Оператор return.

1. В отличии от других языков программирования высокого уровня в С нет деления на процедуры, подпрограммы и функции (например, Pascal). С- программа состоит только из функций.

Функция – самостоятельная единица программы, спроектированная для реализации конкретной задачи. Каждая функция должна иметь имя, которое используется для вызова. Имя функции main(), которая обязательно присутствует в любой С- программе, зарезервировано. В программе на С могут присутствовать несколько функций, причем функция main() необязательно должна быть первой. Но с функции main() всегда начинается выполнение любой С- программы.

При вызове функции ей могут быть переданы данные посредством аргументов. Функция может возвращать значение, которое и есть основной результат выполнения функции; этот результат подставляется на место вызова функции, где бы этот вызов не встретился в программе. Могут быть определены функции, которые не имеют никаких параметров и не возвращают никакого значения. Тем не менее действие таких функций может состоять в изменении внешних переменных или статических переменных или выполнении каких-либо других действий, не связанных с данными.

2. С использованием функции на языке С связаны три понятия:

-определение функции;

-объявление функции;

-вызов функции;

Определение функции задаёт имя функции, типы и число её формальных параметров, а так же объявление и операторы, которые определяют действие функции. Последовательность объявлений и операторов называется телом функции. В определении функции также может быть задан тип значения, возвращаемого функцией (он же и является типом функции), а также класс памяти.

Синтаксис определения функции следующий:

[<спецификация КП>][<спецификация типа>] <описатель>([<список параметров>])<тело функции>

Приведем пример определения функции, проверяющей, является ли заданный символ русской буквой:

```
int rus(int c)
{
if(c>=(unsigned char)'A'&&c<=(unsigned char)' ё ')
  return 1;
else
  return 0;
}
```

В С нет требования, чтобы определение функции обязательно предшествовало вызову функции. Определения используемых функций могут следовать за определением функции main() или могут находится вообще в другом файле.

Однако, для того, чтобы компилятор мог выполнить проверку соответствия типов передаваемых аргументов типам формальных параметров в определении функции, до вызова функции нужно поместить объявление (прототип) функции.

Прототип функции имеет такой же формат, что и определение функции, с той лишь разницей, что не имеет тела функции и что заголовок функции заканчивается точкой с запятой. Прототип функции задаёт имя функции, типы и число формальных параметров, тип возвращаемого значения (тип функции) и класс памяти. Формальные параметры в прототипе могут иметь имена, но эти имена компилятору не нужны. Прототип приведенной выше функции имеет вид: int rus(int);

Данный прототип объявляет, что функция rus() возвращает целое значение (int) и имеет один формальный параметр типа int.

Вызов функции представляет собой либо отдельный оператор вызова, либо выражение, которое является частью другого оператора, как правило, присваивания. При вызове функции указывается имя и фактические аргументы, которые представляются вместо формальных параметров в определении функции.

Форма заголовка функции в определении, когда объявление типов формальных параметров находится в круглых скобках вслед за именем функции, соответствует стандарту ANSI языка С.

3. Формальные параметры – это переменные, которые принимают значения, передаваемые функции во время вызова. Предполагается, что функция имеет столько аргументов, сколько формальных параметров задано в списке.

Если функции не передаются аргументы, то вместо списка формальных параметров необходимо указать ключевое слово void.

Порядок и типы формальных параметров должны быть одни и те же в определении функции и в объявлении (прототипе). Типы фактических аргументов в вызовах функции должны быть совместимы с типами соответствующих формальных параметров. Формальный параметр может быть любого установленного типа, структурой, объявлением, перечислением, указателем или массивом.

Формальные параметры в теле функции используются в качестве ссылок на передаваемые аргументы.

Тело функции- составной оператор, содержащий операторы, определяющие действия функции. Этот оператор может содержать внутри себе объявления переменных, используемых внутри функции. Все переменные, объявленные в теле функции, имеют класс памяти auto, если они не объявлены иначе, и являются локальными. Когда вызывается функция, то для локальных переменных отводится память в стэке и производится инициализация (если она задана). Далее управление передаётся первому оператору функции, и начинается процесс выполнения, который продолжается до тех пор, пока не встретится оператор return или конец тела функции. Управление при этом передается в точку вызова функции.

Если функция возвращает значение, то должен быть выполнен оператор return, содержащий некоторое выражение. Значение возврата не определено, если оператор return не выполнен, или если в операторе return не было указано выражение.

```
#include "stdafx.h"
#define _CRT_SECURE_NO_WARNINGS // для возможности использования scanf
#include <stdio.h>
// Функция вычисления суммы двух чисел
int sum(int x, int y) // в функцию передаются два целых числа
{
    int k = x + y;  // вычисляем сумму чисел и сохраняем в k
    return k;       // возвращаем значение k
}
int main()
{
    int a, r;      // описание двух целых переменных
    printf("a= ");
    scanf("%d", &a); // вводим a
    r = sum(a, 5);    // вызов функции: x=a, y=5
    printf("%d + 5 = %d", a, r); // вывод: a + 5 = r
    getchar(); getchar(); // мы использовали scanf(),
    return 0;  // поэтому getchar() вызываем дважды
}
```

Рисунок 5.1 – Код программы с применением пользовательских функций

Задание.

Общее условие для всех вариантов:

Ввести в память машины одномерный целочисленный массив, размерностью не более 50 элементов.

Произвести манипуляции с массивом данных согласно условию, предложенному в варианте задания.

Вывести на экран исходный массив и данные, полученные согласно заданию в варианте.

Ввод массива, манипуляции с массивом, вывод данных произвести через отдельные пользовательские функции.

Вариант 1.

1. Поменять местами максимальный и минимальный элементы массива. Все элементы массива разные.

2. Вычислить сумму ненулевых чисел, стоящих на четных местах, и подсчитать количество чисел, принадлежащих промежутку (M, K).

Вариант 2.

1. Подсчитать количество ненулевых элементов.
2. Поменять местами первый элемент и самый левый максимальный.

Вариант 3.

1. Поменять местами последний элемент с первым нулевым элементом.
2. Поменять местами предпоследний элемент с самым правым минимальным.

Вариант 4.

1. Определить, есть ли в массиве простые числа. Если да то вывести их на экран.
2. Определить, есть ли в массиве отрицательные числа. Если да, то вывести их на

экран.

Вариант 5.

1. Найти среднее арифметическое чисел, неравных заданному числу C, и подсчитать количество неположительных чисел, стоящих на четных местах.

2. Посчитать количество трехзначных отрицательных чисел.

Вариант 6.

1. Посчитать количество чисел, которые начинаются одинаково слева на право и

справа налево.

2. Посчитать сумму всех чисел, сумма цифр которых меньше 8.

Вариант 7.

1. Посчитать количество всех чисел, сумма цифр которых меньше 10.
2. Вывести все числа, сумма цифр которых меньше 15.

Вариант 8.

1. Перевернуть массив так, чтобы первый стал последним, а последний – первым и так далее.
2. Упорядочить массив таким образом, чтобы в начале шли отрицательные элементы, затем нулевые, затем положительные.

Вариант 9.

1. Найти максимальное число и поменять его местами с последним элементом массива.
2. Найти минимальное число и поменять его местами с предыдущим элементом массива.

Вариант 10.

1. Найти минимальное число и поменять его местами с последующим элементом массива.
2. Найти максимальное число и поменять его местами с 6 элементом массива.

Вариант 11.

1.Найти максимальное число, присвоить его значение последнему элементу, вместо максимального числа записать –1.

2. Найти минимальное число, присвоить его значение первому элементу массива, а вместо минимального записать число 99999.

Вариант 12.

1. Найти минимальное число и поменять его местами с третьим элементом массива.
2. Найти максимальное число и поменять его местами с предпоследним элементом массива.

Вариант 13.

1. Найти минимальное число и присвоить его значение элементу с номером (N –3),а вместо минимального числа записать число 101.
2. Найти максимальное число и поменять его местами с элементом под номером (N –4).

Вариант 14.

1. Найти минимальное число и записать вместо него число N^2+N.
2. Найти максимальное число и поменять его местами со вторым элементом массива.

Вариант 15.

1. Найти минимальное число и поменять его местами с последним элементом массива.
2. Найти максимальное число и вместо него записать значение N+2.

Вариант 16.

1. Найти минимальное число и присвоить его значение третьему элементу массива.
2. Найти минимальное число и вместо него записать значение N^2.

Вариант 17.

1. Найти максимальное число и поменять его местами со вторым элементом массива.
2. Найти минимальное число, присвоить его значение последнему элементу массива, а вместо минимального числа записать значение 3N.

Вариант 18.

1. Найти максимальное число и поменять его местами с четвертым элементом массива.
2. Найти минимальное число и вместо него записать значение SIN (N).

Вариант 19.

1. Найти максимальное число и присвоить его значение элементу с номером (N-3).
2. Найти минимальное число и присвоить это значение второму элементу массива.

Вариант 20.

1. Найти максимальное число и поменять его местами со вторым элементом массива.
2. Найти минимальное число и поменять его местами с элементом массива, номер которого задан.

Вариант 21.

1. Найти максимальное число и поменять его местами с последующим элементом массива.
2. Найти минимальное число, присвоить его значение первому элементу массива, а вместо минимального числа записать 10.

Вариант 22.

1. Найти минимальное число, присвоить его значение второму и четвертому элементу массива, а вместо минимального числа записать сумму второго и четвертого элементов массива.
2. Найти максимальное число и поменять его местами с элементом, номер которого задан.

Вариант 23.

1. Найти минимальное число и заменить его на полусумму первого и последнего элементов.
2. Найти максимальное число и поменять его местами с третьим от конца элементом массива.

Вариант 24.

1.Найти максимальное из чисел a_n , a_1 a_2, … , a_1 a_2 … a_n и поменять его местами с первым элементом массива.

2. Найти минимальное из чисел a_1+a_2, a_2+a_3, … , $a_{n-1}+a_n$ и поменять его местами с предпоследним элементом массива.

Вариант 25.

1. Подсчитать количество максимальных элементов.

2. Вычислить количество чисел, которые кратны 3 и не кратны 5; подсчитать произведение положительных чисел, стоящих на нечетных местах.

Вариант 26.

1.Подсчитать количество нечетных чисел, имеющих четные порядковые номера; вычислить сумму чисел, кратных заданному P.

2.Вычислить сумму квадратов отрицательных чисел, стоящих на местах, кратных 3; подсчитать количество чисел, непринадлежещих промежутку {A, B).

Вариант 27.

1. Вычислить сумму чисел, принадлежащих промежутку {A, B}; Подсчитать количество нулей, стоящих на местах, кратных 4.

2. Вычислить сумму квадратов чисел, принадлежащих промежутку (A, B} и стоящих на четных местах; подсчитать количество чисел, больших заданного C.

Вариант 28.

1. Вычислить произведение чисел, больших заданного D и стоящих на местах, кратных 3; подсчитать также количество чисел, являющихся квадратами четных чисел.

2. Подсчитать количество ненулевых чисел, стоящих на нечетных местах, и вычислить среднее арифметическое чисел, меньших заданного T.

Вариант 29.

1. Вычислить сумму квадратов чисел, неменьших заданного V и стоящих на местах, кратных 4; подсчитать также количество чисел, равных W.

2. Подсчитать количество чисел, больших заданного L и стоящих на местах, кратных 3; вычислить произведение чисел, принадлежащих промежутку {C, D).

Вариант 30.

1. Найти среднее арифметическое чисел, неравных заданному числу C, и подсчитать количество неположительных чисел, стоящих на четных местах.

2. Посчитать количество двузначных положительных чисел.

Лабораторная работа № 6

Тема работы: Программирование задач по работе с многомерными массивами при адресации элементов массива обычным способом и через указатели.

Теоретические сведения по изучению темы:

Язык Си, позволяет использовать массивы, размерность которых отлична от единицы. Такой массив можно описать следующим образом int a[2][3]. В данном случае объявлен двумерный массив в виде матрицы

a[0][0] a[0][1] a[0][2]
a[1][0] a[1][1] a[1][2]

Количество байт памяти, которое необходимо для хранения двумерного массива, вычисляется по формуле:количество байт = <размер типа данных>*<количество строк>*<количество столбцов>.

В памяти компьютера двумерный массив располагается непрерывно по строкам, т.е. a[0][0], a[0][1], a[0][2], a[1][0], a[1][1], a[1][2]. Память для всех массивов, которые определены как глобальные, отводится в процессе компиляции и сохраняется все время, пока работает программа.

При работе с многомерными массивами возникает вопрос, как создать указатель на многомерный массив и как с ним работать. Предположим что у нас описан массив mas[3][4] типа int и указатель на тип int: int * ptmas;

Ptmas указывает на первый столбец первой строки:

Ptmas ⇔ mas ⇔ &mas[0] [0];

На что будет указывать ptmas+1?

В связи с тем, Что элементы двумерного массива располагаются в памяти построчно, то можно записать:

ptmas ⇔ &mas [0] [0];
ptmas+1 ⇔ &mas [0] [1];
ptmas+2 ⇔ &mas [1] [0];
ptmas+3 ⇔ &mas [1] [1];
…
ptmas+5 ⇔ &mas [2] [1];
…
ptmas+7 ⇔ &mas [3] [1].

Рассмотрим массив mas как массив массивов, то есть это массив из четырех строк, каждая из которых является массивом из двух элементов. Имя первой строки mas [0], четвертой – mas [3].

Мы знаем ,что имя массива – есть указатель на его первый элемент, тогда можем записать следующие соотношения:

Ptmas[0] ⇔ &mas [0] [0];
Ptmas[1] ⇔ &mas [1] [0];
Ptmas[2] ⇔ &mas [2] [0];
Ptmas[3] ⇔ &mas [3] [0]

Данные выводы позволяют использовать функцию, предназначенного для одномерного массива, для работы с двумерным массивом. В качестве примера рассмотрим нашу программу нахождения одного значения элементов массива, но в качестве массива будем использовать двумерный массив (рисунок 6.1).

```
#include "stdafx.h"
#include <iostream>
#define _CRT_SECURE_NO_WARNINGS // для возможности использования scanf
using namespace std;
int mean(int [], int);
void main(void)
{
    setlocale(0, "");
    static int mas[3][4] = {
        { 2, 4, 6, 8 },
        {100, 200, 300, 400},
        {10, 40, 60, 90}
    };

    for (int row = 0; row<3; row++)
     printf("Среднее строки %d равно  %d  \n", row, mean(mas[row], 4));
}

int mean(int array[], int n)
{
    int index;
    long sum;
    if (n>0)
    {
        for (index = 0, sum = 0; index<n; index++)
            sum += (long)array[index];
        return((int)(sum / n));
    }
    else
    {
        printf("Нет массива.\n");
        return  (0);
    }
}
```

Рисунок 6.1 – Исходный код программы

Результаты работы программы, представлен на рисунке 6.2.

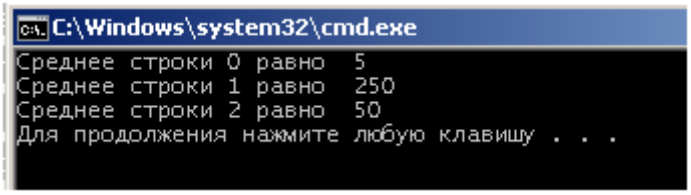

Рисунок 6.2 – Результат работы программы

Предположим, программист хочет иметь функцию, работающую с двумерным массивом не по частям, как в предшествующем примере, а со всем целиком. Пусть функция main() выглядит, как показано на рисунке 6.3.

```
#include "stdafx.h"

void funk(int mas[][4]);

int _tmain(int argc, _TCHAR* argv[])
{
    {
        static int  mas[3][4] = {
            { 2, 4, 6, 8, },
            { 100, 200, 300, 400 },
            { 10, 40, 60, 90 }
        };
        funk(mas);
    }

    return 0;
}


void funk(int mas[][4])
{}
```

Рисунок 6.3 – Исходный код программы

Здесь функция funk() использует в качестве аргумента mas, являющимся указателем на весь массив. Как записать правильно заголовок в объявлении функции funk(). Возможны следующие варианты:

1)funk (int mas[])
2)funk (int mas[][])
3)funk (int mas[][4])

В первом случае рассматривается mas как одномерный массив, состоящий из 12 элементов. Информация о расчленении на строки отсутствует. Второй случай ошибочный. Указывается, что mas двумерный массив, но нигде не говорится из чего он состоит. Третий

случай приемлем. Он сообщает компилятору, что массив следует разбить на строки по четыре столбца.

Задание.

Задачи должны иметь два варианта решения. Первый вариант решения – обычная адресация элементов массива; второй вариант – адресация через указатели с использованием функций.

1. Задан двухмерный массив (матрица) вещественных чисел. Найти наибольший и наименьший элементы массива. Размерность массива (число столбцов, число строк) и значения его элементов ввести с клавиатуры.

2. Задан двухмерный массив вещественных чисел размерности 3х3. Найти номер строки и номер столбца, в которых находится наименьший элемент. Массив описать как типизированную константу.

3.Задан двухмерный массив вещественных чисел. Необходимо каждый элемент соответствующей строки разделить на сумму элементов этой строки. Размерность массива (число столбцов, число строк) и значения его элементов ввести с клавиатуры.

4. Задан двухмерный массив вещественных чисел. Заменить все отрицательные элементы нулями и подсчитать сумму положительных элементов в каждом столбце. Размерность массива (число столбцов, число строк) и значения его элементов ввести с клавиатуры.

5. Задан двухмерный массив *B* размерности Зх.5. Получить новый массив с именем *V* путем удаления из *B* строки и столбца, в которых содержится минимальный элемент. Описать массив *B* как типизированную константу.

6. Задан двухмерный массив *A* размерности /мхи. Дополнить его (n+1)-й строкой и (п-И)-м столбцом, в которых записать суммы элементов соответствующих строк или столбцов исходного массива *A*. В элементе $a_{m+n,n+1}$ должна храниться сумма элементов первоначального массива. Исходные элементы массива *A, m и n* необходимо ввести с клавиатуры.

7. Задана матрица (двухмерный массив) размерности *m*х*n.* Необходимо, не используя другого вспомогательного массива, транспонировать данную матрицу. Исходные данные ввести с клавиатуры.

8. Задана целочисленная квадратная матрица порядка 4. Из этой матрицы выбрать столбец, который обладает наибольшей суммой модулей элементов. Если таких столбцов несколько, то взять первый из них. Далее в данном столбце найти наименьшее из значений его элементов.

9. Задана вещественная матрица размерности *n*х*m,* все элементы которой различны. В каждой строке выбрать элемент с наименьшим значением, затем среди этих чисел выбрать наибольшее. Указать индексы (номер строки и номер столбца) элемента с найденным значением.

10. Задан массив *A* целых чисел следующего вида:

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a1n \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots. & \ldots & \ldots & \ldots \\ a_{n1}a_{n2}\ldots a_{nn} & & & \end{bmatrix}$$

Вывести на экран значения его элементов по столбцам в следующем виде:

a_{11}

a_{21} a_{31} … a_{n1}

a_{12} a_{22}

a_{32} a_{42} … a_{n2}

… … … …

a_{1n} a_{2n} … a_{nn}

11. Заданы матрицы *A* размерности *m*x*n* и *B* размерности *n*x*l*. Определить матрицу *C* размерности *m*x*l*, равную произведению матриц *A* и *B,* т.е.

$$c_{ij} = \sum_{k=l}^{n} a_{ik} b_{kj}$$ для *j*=l,..., /, Исходные данные: m=2, n=2, l=3,

12. Задана вещественная матрица размерности nxm, все элементы которой различны. В каждой строке выбрать элемент с наименьшим значением, затем среди этих чисел выбрать наибольшее. Указать индексы (номер строки и номер столбца) элемента с найденным значением.

13. Дан двухмерный массив, содержащий 4 строки и 5 столбцов. Элементами массива являются целые числа. Упорядочить массив по невозрастанию элементов пятого столбца.

14. Определить, является ли заданная целая квадратная матрица 10-го порядка симметричной (относительно главной диагонали).

15. 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 цели матрицы размером 10x15 напечатать индексы всех ее седловых точек.

16. Дана вещественная матрица размером 7x7, все элементы которой различны. Найти скалярное произведение строки, в которой находится наибольший элемент матрицы, на столбец с наименьшим элементом.

17. Определить, является ли. заданная матрица 10-го порядка орто-нормированной, т.е. такой, в которой скалярное произведение каждой пары раз-личных строк равно 0, а скалярное произведение каждой строки на себя равно 1.

18. Определить, является ли заданная целая квадратная матрица 9-го порядка магическим квадратом, т.е. такой, в которой суммы элементов во всех строках и столбцах одинаковы.

19. Дана квадратная матрица n-ого порядка (n=6). Найти матрицу, обратную ей, или установить, что такой не существует. (Замечание: если линейными преобразованиями строк привести заданную матрицу к единичной, то этими же преобразованиями единичная матрица будет приведена к обратной матрице.)

20. Даны координаты n-векторов n-мерного линейного пространства (n=7). Определить, является ли она линейно независимой.

Дополнительное задание.

Вариант 1:

Задана квадратная матрица порядка N. Подсчитать количество положительных чисел, лежащих выше главной диагонали. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 2:

Задана квадратная матрица порядка N. Вычислить сумму отрицательных элементов, лежащих ниже главной диагонали. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 3:

Задана квадратная матрица порядка N. Вычислить произведение чисел, принадлежащих промежутку A, B и лежащих на главной диагонали и выше ее. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 4:

Задана квадратная матрица порядка N. Вычислить сумму квадратов элементов, больших заданного C и находящихся на главной диагонали и выше ее. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 5:

Задана квадратная матрица порядка N. Подсчитать количество чисел, меньших заданного D и находящихся выше побочной диагонали. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 6:

Задана квадратная матрица порядка N. Вычислить сумму положительных элементов, лежащих ниже побочной диагонали.

Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 7:

Задана квадратная матрица порядка N. Вычислить произведение отрицательных элементов, находящихся на побочной диагонали и выше ее. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 8:

Задана квадратная матрица порядка N. Вычислить сумму квадратов элементов, принадлежащих заданному промежутку (E, F] и лежащих на побочной диагонали и ниже ее. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 9:

Задана квадратная матрица порядка N. Вычислить среднее арифметическое элементов, больших заданного G и лежащих выше главной диагонали. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 10:

Задана квадратная матрица порядка N. Подсчитать количество чисел, меньших заданного H и лежащих ниже главной диагонали. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 11:

Задана квадратная матрица порядка N. Вычислить сумму чисел, принадлежащих промежутку [K,L) и лежащих на главной диагонали и выше ее. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 12:

Задана квадратная матрица порядка N. Вычислить произведение положительных чисел, лежащих на главной диагонали и ниже ее. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 13:

Задана квадратная матрица порядка N. Вычислить сумму квадратов отрицательных чисел, лежащих выше побочной диагонали. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 14:

Задана квадратная матрица порядка N. Вычислить среднее арифметическое чисел, не больших заданного M и лежащих ниже побочной диагонали. Программа должна выполнять ввод и вывод матрицы

и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 15:

Задана квадратная матрица порядка N. Подсчитать количество чисел, не меньших заданного P и лежащих на побочной диагонали и выше ее. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 16:

Задана квадратная матрица порядка N. Вычислить сумму чисел, больших заданного R и лежащих на побочной диагонали и ниже ее. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 17:

В матрице M x N переставить строки таким образом, чтобы получилась последовательность $x_1 \geq x_2 \geq \ldots \geq x_m$, где x_i – произведение всех элементов i – ой строки. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 18:

В матрице M x N переставить столбцы таким образом, чтобы получилась последовательность $z_1 \leq z_2 \leq \ldots \leq z_n$, где z_j – сумма всех элементов j – ого столбца. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 19:

В матрице M x N переставить строки таким образом, чтобы получилась последовательность $q_1 \geq q_2 \geq \ldots \geq q_m$, где q_i – минимальное значение среди всех элементов i – ой строки. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 20:

В матрице M x N переставить столбцы таким образом, чтобы получилась последовательность $t_1 \leq t_2 \leq \ldots \leq t_n$, где t_j – максимальное значение среди всех элементов j – ого столбца. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 21:

В матрице M x N переставить строки таким образом, чтобы получилась последовательность $s_1 \leq s_2 \leq \ldots \leq s_m$, где s_i - сумма абсолютных значений всех элементов i – строки. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 22:

В матрице M x N переставить столбцы таким образом, чтобы получилась последовательность $r_1 \geq r_2 \geq \ldots \geq r_n$, где r_j – количество нулевых элементов в j – ом столбце. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 23:

В матрице M x N переставить строки таким образом, чтобы получилась последовательность $l_1 \leq l_2 \leq \ldots \leq l_m$, где L_i – количество отрицательных элементов i – ой строки. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 24:

В матрице M x N переставить столбцы таким образом , чтобы получилась последовательность $k_1 \geq k_2 \geq \ldots k_n$, где k_j – количество положительных элементов в j – ом столбце. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 25:

В матрице M x N переставить столбцы таким образом, чтобы получилась последовательность $p_1 \leq p_2 \leq \ldots \leq p_n$, где p_j – произведение всех элементов j – ого столбца. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 26:

В матрице M x N переставить строки таким образом, чтобы получилась последовательность $f_1 \geq f_2 \geq \ldots \geq f_m$, где f_i – сумма всех элементов i – ой строки. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 27:

В матрице M x N переставить столбцы таким образом, чтобы получилась последовательность $c_1 \leq c_2 \leq \ldots \leq c_n$, где c_j – минимальное значение среди всех элементов j –ого столбца. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 28:

В матрице M x N переставить строки таким образом, чтобы получилась последовательность $d_1 \geq d_2 \geq \ldots \geq d_m$, где d_i- максимальное значение среди всех элементов i – ой строки. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 29:

Задана квадратная матрица порядка N. Вычислить сумму положительных чисел, лежащих в диапазоне от 2 до 8 на главной диагонали и выше ее. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Вариант 30:

Задана квадратная матрица порядка N. Вычислить количество чисел, лежащих в диапазоне [-5;5] на побочной диагонали и выше ее. Программа должна выполнять ввод и вывод матрицы и дополнительных данных, выполнять необходимые действия и выводить результаты.

Лабораторная работа № 7

Тема работы: Обработка символьной информации

Теоретические сведения по изучению темы:

Символьные строки представляют собой один из наиболее полезных и важных типов данных в языке Си. Символьные строки являются массивом типа char, который заканчивается нуль-символом (символ, ASCII-код у которого равен нулю). Существует много способов определения символьных строк в программе.

Основными среди них являются:

1. использование строковых констант;
2. использование массивов типа char;
3. использование указателей типа char;
4. использование массивов символьных строк.

Строковая константа – это список литер, заключенных в двойные кавычки: “Это строковая константа”.

В конец строковой константы не надо ставить символ ‘\0’ (нуль-символ).

Символьная строка – это одномерный массив типа char, заканчивающийся нуль-символом. Для нуль-символа определена специальная символьная константа ‘\0’. Это следует учитывать при описании соответствующего массива символов. Так, если строка должна содержать N символов, то в описании массива следует указать N+1 элемент. Описание: char str[11];, предполагает, что строка str содержит 10 символов, а последний байт зарезервирован под нулевой байт.

Символьные массивы (массивы типа char) занимают в языке Си особое место. Во многих языках программирования есть специальный тип данных – строка символов (string). В языке Си отдельного типа строки символов нет, а реализована работа со строками путем использования одномерных массивов типа char и специальных библиотечных функций.

При инициализации строки символов ее можно рассматривать как одномерный массив символов и соответственно инициализировать как одномерный массив:char str[]= “Строка”; /*массив*/

Также можно инициализировать строку через указатель на тип char. Достаточно присвоить указателю значение строковой константы. При этом не нужно заранее выделять память для строки. Компилятор по внешнему виду строки определит необходимое количество байтов, в том числе один байт на символ конца строки: char *pas; pas= “Строка”; или char *pas = “ Строка”; /*указатель*/, str – это массив, имеющий такой объем, что в нем как раз помещается указанная последовательность символов и ‘\0’. Отдельные символы внутри массива могут изменяться, но

str всегда указывает на одно и то же место памяти. В противоположность ему pas есть указатель, инициализированный так, чтобы указывать на строковую константу. А значение указателя можно изменить, и тогда последний будет указывать на что-либо другое. Кроме того, результат будет неопределен, если вы попытаетесь изменить содержимое константы.

Рассмотрим реальную задачу сортировки строк в алфавитном порядке (рисунок 7.1). Данная задача возникает весьма часто при написании профессиональных программ. Если полагать, что играющую роль будет играть функция strcmp(), используемая для определения старшинства двух строк.

```
#include "stdafx.h"
#include <iostream>
#define _CRT_SECURE_NO_WARNINGS // для возможности использования scanf
using namespace std;
void sort(char *[], int);

void main()
{   setlocale(0, "");
    static char input[20][81];
    static char halt[] = "";
    char *pts[20];
    int count = 0;
    int k, kol = 20;

    printf("Введите до %d строк и они будут отсортированы\n", kol);
    printf("Для прекращения ввода нажмите на Enter в начале строки\n");
    while ((gets(input[count]) != NULL) && (strcmp(input[count], halt) != 0) && (count++<20))
        pts[count - 1] = input[count - 1];
    sort(pts, count);
    puts("Отсортированный список");
    for (k = 0; k<count; k++)
        puts(pts[k]);
}

void sort(char *string[], int name)
{

    char *temp;
    int toop, seek;

    for (toop = 0; toop<name - 1; toop++)
        for (seek = toop + 1; seek<name; seek++)
            if (strcmp(string[toop], string[seek])>0)
            {
                temp = string[toop];
                string[toop] = string[seek];
                string[seek] = temp;
            }
}
```

Рисунок 7.1 – Исходный код программы

В данном примере вместо перегруппировки строк перегруппировываются их указатели, а именно вначале **pts[0]** ссылается на **input[0], pts[1]** ссылается на **input[1]**. Каждый **input** является массивом из

81 элемента, а каждый элемент **pts** является отдельной переменной функции сортировки **sort().** Перегруппировывает **pts** не трогая **input[]**. Если например, **input[1]** стоит перед **input[0]** по алфавиту, то программа изменяет указатели **pts,** так что **pts[0]**ссылается на **input[1]**, а **pts[1]** на **input[0]**. Это гораздо легче, чем использовать strcmp(), меняя местами две введенные строки.

Символьная строка состоит из последовательности символов, заключенных в двойные кавычки. Эта последовательность представляется в памяти как массив элементов типа char. Символьная строка представляет в выражении адрес этого массива, т.е. адрес первого элемента строки.

Поскольку символьная строка представляет адрес массива, она может быть использована в контексте, допускающем значение типа указатель, подчиняясь при этом тем же ограничениям. Однако, поскольку адрес символьной строки является постоянной величиной, символьная строка не может составлять левую часть операции присваивания.

Для работы со строками существует специальная библиотека, описание которой находится в файле string.h. Наиболее часто используемыми функциями из этой библиотеки являются функции: strcpy(), strcat(), strlen(), strcmp().

Функция strcpy()

Вызов функции имеет вид: strcpy(s1,s2);

Функция strcpy() используется для копирования содержимого строки s2 в строку s1. Массив s1 должен быть достаточно большим, чтобы в него поместилась строка s2. Если мало места, компилятор не выдает указания на ошибку или предупреждения; это не прервет выполнения программы, но может привести к порче других данных или самой программы и неправильной работе программы в дальнейшем.

Функция strcat()

Вызов функции имеет вид: strcat(s1,s2);

Функция strcat() присоединяет строку s2 к строке s1 и помещает ее в массив, где находилась строка s1, при этом строка s2 не изменяется. Нуль-символ, который завершал строку s1, будет заменен первым символом строки s2. Ив функции strcpy(), и в функции strcat() получающаяся строка автоматически завершается нуль-символом.

Функция strcmp()

Вызов функции имеет вид: strcmp(s1,s2);

Функция strcmp() сравнивает строки s1 и s2 и возвращает значение 0, если строки равны, т.е. содержат одно и то же число одинаковых символов. Код первого символа одной строки сравнивается с кодом символа второй строки. Если они раны, рассматриваются вторые символы, и т.д. Если s1 больше s2 (сравнение происходит по кодам символов), то функция strcmp() возвращает положительное значение, если меньше – отрицательное значение.

Функция strlen()

Вызов функции имеет вид: strlen(s);

Функция strlen() возвращает длину строки s, при этом завершающий нуль-символ не учитывается. Вызов: strlen(“Привет”);Вернет значение 6.

Задание.

В задаче задается строка текста, состоящая из нескольких слов. Слова отделяются последовательностью пробелов. Варианты заданы в таблице 7.1.

Таблица 7.1 – Варианты индивидуальных заданий

Вариант	**Задание**
1	**2**
1	Подсчитать количество слов и после каждого поставить запятую
2	Подсчитать количество букв в третьем слове.
3	Последнее слово строки поставить после первого.
4	Определить количество слов, начинающихся с буквы «А».
5	Перед первой буквой каждого слова вставить символ «*».
6	Во втором слове после каждой буквы вставить пробел.
7	Для каждого слова указать количество букв, из которых оно состоит.
8	Выделить те слова, длина которых превышает 5.
9	Определить количество слов, в которых буква «П» встречается хотя бы один раз.
10	Удалить последнюю букву в каждом слове.
11	Подсчитать количество букв в предпоследнем слове.
12	Перед каждой буквой третьего слова поставить символ «\».
13	Удалить все пробелы из строки, кроме тех, которые стоят между первым и вторым словом.

Продолжение таблицы 7.1

1	2
14	После последней буквы каждого слова вставить символ «-».
15	Определить количество слов, которые оканчиваются на «Е».
16	Для каждого слова указать, сколько букв «И» в нем содержится.
17	Выделить те слова, которые по длине меньше 5,
18	Определить количество слов, в которых нет ни одной буквы «Е».
19	Переставать первое слово в конец строки.
20	Каждое слово заключить в квадратные скобки.
21	Подсчитать количество букв во втором слове.
22	После каждой буквы предпоследнего слова вставить символ «*».
23	Удалить пробелы, стоящие между первым и вторым словами, а также между двумя последними словами.
24	После первой буквы каждого с лова 'вставить символ «-».
25	Определить количество слов, вторая буква которых «Р».
26	Для каждого слова, кроме последнего, указать, сколько пробелов стоит после него.
27	Выделить те слова, длина которых равна заданному числу.
28	Определить количество слов, в которых первая и последняя буквы совпадают.
29	Поменять местами первое и последнее слова.
30	Установить пробелы вместо символов, номера позиций которых при делении на 4 дают в остатке 3.

Лабораторная работа № 8

Тема работы: Написание программ с использованием файлов

Цель работы: Получить практические навыки в написании программ на языке С, с использованием файлов

Теоретические сведения по изучению темы:

Файл – это набор данных, размещенный на внешнем носителе и рассматриваемый в процессе обработки как единое целое. В файлах размещаются данные, предназначенные для длительного хранения.

Различают два вида файлов: *текстовые* и *бинарные*.

Текстовые файлы представляют собой последовательность ASCII символов и могут быть просмотрены и отредактированы с помощью любого текстового редактора. Эта последовательность символов разбивается на строки символов, при этом каждая строка заканчивается двумя кодами «перевод строки», «возврат каретки»: 13 и 10 (0xD и 0xA).

Бинарные (двоичные) файлы представляют собой последовательность данных, структура которых определяется программно.

В языке Си не предусмотрены никакие заранее определенные структуры файлов. Все файлы рассматриваются компилятором как последовательность (поток байт) информации.

Для файлов определен маркер или указатель чтения-записи данных, который определяет текущую позицию доступа к файлу. Напомним, что с началом работы любой программы автоматически открываются стандартные потоки *stdin* и *stdout*.

Открытие файла

Каждому файлу в программе присваивается внутреннее логическое имя, используемое в дальнейшем при обращении к нему. Логическое имя (идентификатор файла) – это указатель на файл, т.е. на область памяти, где содержится вся необходимая информация о файле.

Формат объявления указателя на файл следующий:

***FILE** *ID_указателя_на_файл;*

Прежде чем начать работать с файлом, т.е. получить возможность чтения или записи информации в файл, его нужно открыть для доступа.

Для этого обычно используется функция

FILE* ***fopen***(char * ID_файла, char *режим);

Данная функция берет внешнее представление – физическое имя файла на носителе (дискета, винчестер) и ставит ему в соответствие логическое имя (программное имя – указатель файла).

Физическое имя, т.е. *ID* файла и путь к нему задается первым параметром – строкой, например, “*a:Mas_dat.dat*” – файл с именем *Mas_dat* и расширением *dat*, находящийся на дискете, “*d:\\work\\Sved.txt*” – файл с

именем *Sved* и расширением *txt*, находящийся на винчестере в каталоге *work*.

Внимание. Обратный слеш «\», как специальный символ в строке записывается дважды.

При успешном открытии функция *fopen* возвращает указатель на файл (в дальнейшем – указатель файла). При ошибке возвращается *NULL*. Данная ситуация обычно возникает, когда неверно указывается путь к открываемому файлу, например, если указать путь, запрещенный для записи.

Второй параметр – строка, в которой задается режим доступа к файлу.

Возможные значения данного параметра следующие:

w – файл открывается для записи (*write*); если файла с заданным именем нет, то он будет создан; если же такой файл уже существует, то перед открытием прежняя информация уничтожается;

r – файл открывается для чтения (*read*); если такого файла нет, то возникает ошибка;

a – файл открывается для добавления (*append*) новой информации в конец;

r+ (*w*+) – файл открывается для редактирования данных, т.е. возможны и запись, и чтение информации;

a+ – то же, что и для *a*, только запись можно выполнять в любое место файла (доступно и чтение файла);

t – файл открывается в текстовом режиме;

b – файл открывается в двоичном режиме;

Последние два режима используются совместно с рассмотренными выше. Возможны следующие комбинации режимов доступа: “*w*+*b*”, “*wb*+”, “*rw*+”, “*w*+*t*”, “*rt*+”, а также некоторые другие комбинации.

По умолчанию файл открывается в текстовом режиме.

Текстовый режим отличается от двоичного тем, что при открытии файла как текстового пара символов «перевод строки» и «возврат каретки» заменяется на один символ «перевод строки» для всех функций записи данных в файл, а для всех функций вывода – наоборот – символ «перевод строки» заменяется на два символа – «перевод строки» и «возврат каретки».

Пример открытия файла:

FILE *f; – объявляется указатель на файл *f*;

f = fopen (" d:\\work\\Dat_sp.dat ", "w"); – открывается для записи файл с логическим именем *f*, имеющий физическое имя *Dat_sp.dat* и находящийся на диске *d* в каталоге *work*, или более кратко:

FILE *f = fopen ("d:\\work\\Dat_sp.dat", "w");

Закрытие файла

После работы с файлом доступ к нему необходимо закрыть с помощью функции

int ***fclose*** (*указатель файла*);

Например, для предыдущего примера файл закрывается так: *fclose* (*f*);

Для закрытия нескольких файлов введена функция:

void ***fcloseall*** (*void*);

Если требуется изменить режим доступа к открытому в настоящий момент файлу, то его необходимо сначала закрыть, а затем вновь открыть с другими правами доступа. Для этого используется функция *FILE** ***freopen*** (char *ID_файла, char *режим, FILE *указатель_файла); которая сначала закрывает файл, заданный в третьем параметре (указатель файла), как это выполняет функция *fclose*, а затем выполняет действия, аналогичные функции *fopen*, используя указанные первый и второй параметры (открывает файл с *ID_файла* и правами доступа *режим*).

В языке Си имеется возможность работы с временными файлами, которые нужны только в процессе работы программы и должны быть удалены после выполнения некоторых вычислений. В этом случае используется функция *FILE** ***tmpfile*** (*void*); которая создает на диске временный файл с правами доступа *w+b*. После завершения работы программы или закрытия этого (временного) файла он автоматически удаляется.

Запись-чтение информации

Все действия по чтению-записи данных в файл можно разделить на три группы:

– операции посимвольного ввода-вывода;

– операции построчного ввода-вывода;

– операции ввода-вывода по блокам.

Рассмотрим основные функции для записи-чтения данных из файлов.

Для работы с текстовыми файлами в библиотеке языка Си содержится достаточно много функций, самыми распространенными из которых являются функции

fprintf, ***fscanf***, ***fgets***, ***fputs***.

Формат параметров этих функций практически такой же, как и формат рассмотренных ранее (см. разд. 5.3, 5.4) функций *printf*, *scanf*, *gets* и *puts*. Так же практически совпадают и действия этих функций. Отличие состоит в том, что *printf* и другие функции работают по умолчанию с экраном монитора и клавиатурой, а функции *fprintf* и другие – с файлом, указатель которого является одним из параметров этих функций.

Рассмотрим общий пример создания текстового файла (рисунок 8.1).

```c
#include "stdafx.h"
#include <iostream>
#define _CRT_SECURE_NO_WARNINGS // для возможности использования scanf
using namespace std;
#include<stdio.h>
void main(void)
{
    FILE *f1;
    int a = 2, b = 3;
    if (!(f1 = fopen("c:\\work\\f_rez.txt", "w + t"))) {        // f1 = NULL
        puts("Open File Error!");
        return;                          // exit(1);
    }
    fprintf(f1,"\t Файл результатов \n");
    fprintf(f1, " %d плюс %d = %d\n", a, b, a + b);
    fclose(f1);
}
```

Рисунок 8.1 – Исходный код программы

Просмотрев содержимое файла любым текстовым редактором, можно убедиться, что данные в нем располагаются точно так, как на экране, если воспользоваться функцией *printf* с такими же списками параметров.

Создание текстовых результирующих файлов обычно необходимо для оформления отчетов, различных документов, а также других текстовых материалов.

Бинарные (двоичные) файлы обычно используются для организации баз данных, состоящих, как правило, из объектов структурного типа. При чтении-записи бинарных файлов удобнее всего пользоваться функциями *fread* и *fwrite*, которые выполняют ввод-вывод данных блоками.

Такой способ обмена данными требует меньше времени.

Функция

unsigned ***fread*** (*void* ***p**, *unsigned* ***size***, *unsigned* ***n***, *FILE* ***f***);

выполняет считывание из файла *f n* блоков размером *size* байт каждый в область памяти, адрес которой *p*. В случае успеха функция возвращает количество считанных блоков. При возникновении ошибки или по достижении признака окончания файла – значение ***EOF*** (*End Of File* – признак окончания файла).

Обратное действие выполняет функция:

unsigned ***fwrite*** (*void* ***p**, *unsigned* ***size***, *unsigned* ***n***, *FILE* ***f***);

при вызове которой в файл *f* будет записано *n* блоков размером *size* байт каждый из области памяти, начиная с адреса *p*.

Позиционирование в файле

Каждый открытый файл, как уже отмечалось, имеет скрытый указатель на текущую позицию в нем. При открытии файла этот указатель

устанавливается на начало данных, и все операции в файле будут производиться с данными, начинающимися в этой позиции.

При каждом выполнении функции чтения или записи указатель смещается на количество прочитанных или записанных байт, т.е. устанавливается после прочитанного или записанного блока данных в файле – это ***последовательный доступ к данным***.

В языке Си/С++ можно установить указатель на некоторую заданную позицию в файле. Для этого используют стандартную функцию ***fseek***, которая позволяет выполнить чтение или запись данных в произвольном порядке.

Декларация функции позиционирования следующая:

int ***fseek***(*FILE* ***f***, *long* ***size***, *int* ***code***);

Значение параметра *size* задает количество байт, на которое необходимо сместить указатель в файле *f*, в направлении параметра *code*, который может принимать следующие значения:

– смещение от начала файла – 0 (*SEEK_SET*);
– смещение от текущей позиции 1 (*SEEK_CUR*)
– смещение от конца файла – 2 (*SEEK_END*)

Таким образом, смещение может быть как положительным, так и отрицательным, но нельзя выходить за пределы файла.

В случае успеха функция возвращает нулевое значение, а в случае ошибки (например, попытка выхода за пределы файла) – единицу.

Доступ к файлу с использованием функции позиционирования (*fseek*) называют ***произвольным доступом***.

Иногда нужно определить текущее положение в файле. Для этого используют функцию со следующей декларацией:

long ***ftell***(*FILE* ***f***);

которая возвращает значение указателя на текущую позицию в файле или –1 в случае ошибки.

Пример программы с использованием текстовых файлов

Создать текстовый файл "new1.txt" записав в него строку из 50 символов. Все символы, отличные от пробела, переписать в новый файл "new2.txt", как показано на рисунке 8.2.

```c
#include "stdafx.h"
#include <iostream>
#include <string.h>
#define _CRT_SECURE_NO_WARNINGS // для возможности использования scanf
using namespace std;

void main()
{
    char ch, sl[50];
    char text[] = "one to tree four";
    FILE *pf, *pr;                          // Указатели на файлы
    pf = fopen("new1.txt", "w");  // Создание нового файла new1.txt
    fprintf(pf, "%s\n", text);         // Запись в файл строки text
    fclose(pf);                              // Закрытие файла pf
    pf = fopen("new1.txt", "r");    // Открытие файла pf для чтения
    pr = fopen("new2.txt", "w");  // Создание нового файла new2.txt
    while (!feof(pf))                    // Пока не конец файла
    {
        ch = getc(pf);                       // Чтение символа ch из файла pf
        if (ch != ' ')
            putc(ch, pr);                     // Запись в файл pr символа ch
    }
    fclose(pr);                             // Закрытие файла pr
    rewind(pf);                            // Возврат указателя на начало файла pf
    fgets(sl, 50, pf);                     // Чтение из файла pf строки в переменную sl
    printf("%s\n", sl);                  // Вывод строки sl на дисплей
    pr = fopen("new2.txt", "r");    // Открытие файла pr для чтения
    while (!feof(pr))                   // Пока не конец файла pr
    {
        ch = getc(pr);                    // Чтение символа из файла pr
        putchar(ch);                    // Вывод символа ch на дисплей
    }
    fclose(pf);                             // Закрытие файлов
    fclose(pr);
    getchar();

}
```

Рисунок 8.2 – Исходный код программы

Задание.

Создать текстовый файл из 5 строк, прочитать созданный файл и получить новый файл согласно своему варианту.

1. Из строк все цифры переписать в новый файл.
2. Переписать все строки в новый файл, заменив пробелы на запятые.
3. Переписать в новый файл все символы из строк, отличные от цифр.
4. Подсчитать количество слов в каждой строке и записать их в новый файл.
5. Переписать все строки в новый файл, заменив все буквы 'м' на 'М'.

6. В новом файле заменить все цифры на восклицательные знаки.

7. Переписать все латинские буквы верхнего регистра из каждой строки в новый файл.

8. Переписать в новый файл строки, удалив из них все русские буквы нижнего регистра.

9. В новом файле заменить все латинские буквы верхнего регистра на буквы нижнего регистра.

10. Переписать в новый файл все строки, заменив все русские буквы нижнего регистра на буквы верхнего регистра.

11. Все слова, начинающиеся с гласных переписать в один файл, а с согласных – в другой новый файл.

12. В новый файл переписать каждую строку наоборот.

13. В новый файл переписать каждую строку, где каждая буква от «а» до «ю» заменяется на следующую по алфавиту, а «я» заменяется на «а».

14. В новый файл переписать каждую строку, где первая буква «а» заменяется на 11-ю, вторая «б» – на 12-ю, третья – на 13-ю, ... , последняя «я» – на 10-ю.

15. В новый файл переписать каждую строку, где после каждой согласной буквы вставляется буква «а».

16. В новый файл переписать каждую строку, где после каждой согласной буквы вставляется слог «ла».

17. В новый файл переписать каждую строку, где каждая пара букв «ле» заменяется на «ю», «са» – на «щ», «ик» – на «ж».

18. В новый файл переписать каждую строку, где каждая из пары букв «си», «ли» и «ти» заменяются соответственно на «иис», «иил» и «иит».

19. В новый файл переписать каждую строку, где после каждой гласной буквы вставляется буква «с».

20. В новый файл переписать каждую строку, где после каждой гласной буквы вставляется слог «ла».

21. В новый файл переписать каждую строку, где каждая из букв «а», «о», «и» заменяется соответственно на «ц», «ш», «щ».

22. В новый файл переписать каждую строку, где каждая буква заменяется на следующую в алфавите по часовой стрелке.

23. В новый файл переписать каждую строку, где каждая буква заменяется на следующую в алфавите против часовой стрелки.

24. В новый файл переписать каждую строку, где каждая буква «а» заменяется на слог «си», а «и» – на «са».

25. В новый файл переписать каждую строку, где четные и нечетные символы меняются местами.

26. В новый файл переписать каждую строку, где символы, кратные двум по порядку следования, заменяются на единицы.

27. В новый файл переписать каждую строку, где символы, кратные двум по порядку следования, заменяются на свой порядковый номер.

28. Из строк все символы арифметических операций переписать в новый файл.

29. В новый файл переписать каждую строку, отредактировав строку таким образом, чтобы между словами находился только один пробельный символ.

30. В новый файл переписать каждое слово с новой строки.

Список использованных источников

1. Информатика. Базовый курс. 2-е издание / Под ред. С.В.Симоновича. - СПб.: Питер, 2007. – 640 с.

2. Касаткин, А. И. Профессиональное программирование на языке Си : управление ресурсами / А. И. Касаткин. - Минск : Вышэйшая школа, 1992. - 432 с.

3. Котлинская Г. П. Программирование на языке СИ : справ. пособие. - Минск : Вышэйш. шк., 1991. - 155 с. УДК 004.43(035.5) ББК 32

4. Макогон В.С. Язык программирования Си для начинающих : Учеб.пособие. - Одесса : Астропринт, 1993. - 96с. УДК 004.43(078) ББК 32

5. Структуры данных в языке СИ [Электронный ресурс] : пособие по курсам "Модели и структуры данных" и "Основы алгоритмизации и программирования" для студентов специальностей 1-40 01 02 "Информационные системы и технологии (по направлениям)" и 1-36 04 02 "Промышленная электроника" дневной и заочной форм обучения / О. А. Кравченко ; Министерство образования Республики Беларусь, Учреждение образования "Гомельский государственный технический университет имени П. О. Сухого", Кафедра "Информационные технологии". - Гомель : ГГТУ, 2010. - 149 с. УДК 004.43(075.8) ББК 32.973.26-018я73

6. Основы алгоритмизации и программирования : курс лекций по одноименной дисциплине для студентов специальности 1-40 01 02 "Информационные системы и технологии (по направлениям)" дневной формы обучения / О. А. Кравченко, С. М. Мовшович, Е. В. Коробейникова. - Гомель : ГГТУ, 2010. - 111 с.

7. Программирование на языке С. Массивы : пособие по выполнению контрольных и лабораторных работ по дисциплине "Вычислительная техника и программирование" для студентов технических специальностей дневной и заочной форм обучения / О. А. Кравченко, Д. А. Литвинов ; кафедра "Информационные технологии". - Гомель : ГГТУ, 2007. - 38 с.

Гридина Елена Ивановна

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ НА ЯЗЫКАХ ВЫСОКОГО УРОВНЯ

Практикум
по одноименной дисциплине
для слушателей специальности переподготовки 1-40 01 73 «Программное обеспечение информационных систем» заочной формы обучения

Подписано в печать 13.04.18.
Формат 60x84/16. Бумага офсетная. Гарнитура «Таймс».
Ризография. Усл. печ. л. 4,42. Уч.-изд. л. 4,56.
Изд. № 32.
http://www.gstu.by

Отпечатано на цифровом дуплекаторе
с макета оригинала авторского для внутреннего использования.
Учреждение образования «Гомельский государственный
технический университет имени П.О. Сухого».
246746, г. Гомель, пр. Октября, 48.