МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

Основы программирования

Учебно-методическое пособие

Санкт-Петербург 2021

Составители: А.В. Туманова

Рецензент:

В учебно-методическое пособие включены требования к содержанию отчета и порядку выполнения работы, а также варианты индивидуальных заданий.

Учебно-методическое пособие предназначено для студентов направлений 02.03.03 – «Математическое обеспечение и администрирование информационных систем» и 09.03.04 – «Программная инженерия», изучающих дисциплину «Основы программирования».

Учебно-методическое пособие подготовлено кафедрой компьютерных технологий и программной инженерии и рекомендованj к изданию редакционно-издательским советом Санкт-Петербургского государственного университета аэрокосмического приборостроения.

Подписано к печати Формат 60x84 1/16. Бумага офсетная. Печать офсетная
Усл. печ. л Усл. кр.-отт. 0,00. Уч.- изд. л Тираж экз. Заказ №

Редакционно-издательский отдел
Отдел электронных публикаций и библиографии библиотеки
Отдел оперативной полиграфии
СПб ГУАП
190000, Санкт-Петербург, ул. Б. Морская, 67

Содержание

Общие указания

Важно внимательно прочитать следующие разделы данного документа:

- Общие указания.
- Общие требования к содержанию отчётов.
- Приложение 1. Рекомендованный список литературы.
- Приложение 3. Пример оформления отчёта.
- Приложение 7. Перечень функций стандартной библиотеки C++.
- Приложение 8. Создание нового проекта в среде Visual C++ 2008.
- Приложение 9. Отладка кода проекта в среде Visual C++.
- Приложение 8. Полезные советы по работе в среде VisualC++ 2008.
- Приложение 9. Печать русских букв в среде Visual C++ 2008

Выбор варианта

Для студентов **очной и очно-заочной (вечерней) формы обучения** номер варианта выбирается как номер студента из списка группы по модулю количества вариантов заданий на лабораторную работу плюс один, то есть

$N_{\text{вар}} = (nn - 1)\ mod\ K + 1$, где

$N_{\text{вар}}$ – номер варианта;

nn – номер студента в списке группы;

K – количество вариантов заданий на лабораторную работу.

Например, номер студента в группе – "23", количество вариантов – 20. Тогда номер варианта задания на лабораторную работу, которое необходимо выполнить студенту будет

$N_{\text{вар}} = (23 - 1)\ mod\ 20 + 1 = 3.$

Для студентов **заочной формы обучения** номер варианта задания на лабораторную работу определяется студентом как две последние цифры номера его студенческого билета, взятые по модулю количества вариантов заданий на лабораторную работу плюс один, то есть

$N_{\text{вар}} = (nn - 1)\ mod\ K + 1$, где

$N_{\text{вар}}$ – номер варианта;

nn – две последние цифры номера студенческого билета;

K – количество вариантов заданий на лабораторную работу.

Например, две последние цифры номера студенческого билета – «76», количество вариантов – 20. Тогда номер варианта задания на лабораторную работу, которое необходимо выполнить студенту будет

$N_{\text{вар}} = (76 - 1)\ mod\ 20 + 1 = 16.$

На первом курсе (второй семестр) в первом семестре дисциплины выполняются 1-5 лабораторные работы. При этом лабораторная работа №5 оформляется как контрольная.

На втором курсе (третий семестр) во втором семестре дисциплины выполняются 6-8 лабораторные работы.

Ссылки на литературу

В данном методическом пособии по ходу изложения материала будут встречаться ссылки на литературу вида [1]. Такая ссылка означает источник в «Приложение 1. Рекомендованный список литературы» на странице 46.

Универсальность функций

В варианте может быть представлено несколько заданий. Каждое задание должно быть выполнено в виде отдельной функции, которая возвращает результат, который выводится или используется в функции «main». Никаких вычислений из текста задания в «main» не должно быть. Только ввод исходных данных и вывод результата.

Для примера рассмотрим задание лабораторной работы «Массивы» вариант №1:

> *В одномерном массиве, состоящем из n вещественных элементов, вычислить:*
> *1. сумму отрицательных элементов массива;*
> *2. произведение элементов массива, расположенных между максимальным и минимальным элементами.*
> *Упорядочить элементы массива по возрастанию.*

В данном варианте можно выделить три самостоятельных операции:

1. Вычисление суммы отрицательных элементов массива.
2. Вычисление произведения элементов массива, расположенных между максимальным и минимальным элементами.
3. Сортировка элементов массива по возрастанию.

Для выполнения лабораторной работы необходимо разработать 3 отдельные функции. Очень важно сперва проанализировать текст задания, выделить решаемые задачи и согласовывать их с преподавателем (см. «Общие требования к содержанию »). Поступая так, студент учится проектировать, а затем кодировать.

В некоторых случаях разумно и необходимо создавать дополнительные функции. Для примера рассмотрим задание лабораторной работы «Файлы» вариант №14:

> *Написать программу, которая считывает текст из файла и выводит его на экран, заменив цифры от 0 до 9 на слова «ноль, «один», …, «девять», начиная каждое предложение с новой строки.*

Здесь в качестве дополнительной функции можно выделить преобразование цифр в слова.

Оформление кода

Программный код, который предоставляется на защиту, должен по возможности быть качественным. К критериям качества относят корректность, надёжность, читабельность, преносимость и эффективность. Поскольку студенты, как правило, только изучают возможности языка C++, то лучше сосредоточится на хороших приёмах и возможностях языка, чтобы в последующем не надо было переучиваться и сразу писать хорошие программы.

Другими словами:

- в программном коде должны быть отступы;
- имена функций и переменных должны отражать своё назначение;
- следует избегать использования глобальных переменных;
- переменные должны иметь подходящий тип с учётом знака и наименьшего размера;
- неиспользуемые переменные должны быть удалены;
- лишние операторные скобки (пары «{}») также должны быть удалены;
- входные параметры функций должны быть объявлены как константы (спецификатор «const»);
- в качестве библиотеки ввода-вывода необходимо использовать потоковую библиотеку ввода-вывода «iostream»;
- по результатам компиляции код не должен содержать ни одной ошибки и предупреждения;
- весь ввод-вывод должен происходить внутри функции «main»;
- программа должна осуществлять проверку значений входных данных как в функции main, так и в функциях, решающие задачу;
- сообщения об ошибках должны выводиться в поток ошибоксerr;
- для явных преобразований необходимо использовать преобразования в стиле C++;
- комментарии программы должны отражать псевдокод.

Рассмотрим эти положения более подробно.

Отступы

Можно использовать два стиля отступов: стиль Олмана

```
void output_matrix(const double M[5][4])
{
	unsigned int i, j;
	for (i=0; i<5; i++)
	{
		for (j=0; j<4; j++)
			cout<<M[i][j] <<" ";
		cout<<endl;
	}
}
```

истиль «K&R»:

```
void output_matrix(const double M[5][4]) {
	unsigned int i, j;
	for (i=0; i<5; i++) {
		for (j=0; j<4; j++)
			cout<<M[i][j] <<" ";
		cout<<endl;
	}
}
```

Как видно из текста, основное отличие заключается в местоположении операторных скобок «{}».

Имена функций и переменных

Имя переменной или функции помечает объект и содержит некоторую информацию о его назначении. Имя должно быть информативным, лаконичным, запоминающимся и, по возможности, произносимым. Многое становится ясным из контекста и области видимости переменной: чем больше область видимости, тем более информативным должно быть имя.

Следует использовать осмысленные имена для глобальных переменных и короткие имена для локальных. Глобальные переменные по определению могут проявиться в любом месте программы, поэтому их имена должны быть достаточно длинными и информативными, чтобы напомнить читателю об их предназначении. Полезно описание каждой глобальной переменной снабжать коротким комментарием:

```
const double MROT = 4330; // минимальный размер оплаты труда
```

Для локальных переменных, наоборот, лучше подходят короткие имена; для использования внутри функции вполне сойдет просто «n», неплохо будет смотреться «size», а вот «numberOfElements» будет явным перебором.

Обычно используемые локальные переменные по соглашению могут иметь очень короткие имена. Так, употребление «i», «j» и «k» для обозначения счетчиков цикла, «p» и «q» для указателей, «s» и «t» для строк стало настолько привычным, что применение более длинных имен не принесет никакой пользы, а наоборот, может даже навредить. Например,

```
for (theElementIndex=0; theElementIndex<numberOfElements; theElementIndex++)
      elementArray[theElementIndex] = theElementIndex;
```

или

```
for (i=0; i<size; i++)
      elem[i] = i;
```

Для имён функций следует использовать активные имена. Обычно имя функции базируется на активном глаголе (в действительном залоге), за которым может следовать существительное:

```
int main()
{
      double M[5][4];
      input_matrix(M);
      transpose_matrix(M);
      output_matrix(M);
      return 0;
}
```

Функции, которые возвращают логическое значение (истина или ложь – «true» или «false»), нужно называть так, чтобы их смысл не вызывал сомнений. Так, из вызова

```
const double MROT = 4330; // минимальный размер оплаты труда
if (check_digit(c)) ...
```

непонятно, когда будет возвращаться «true», а когда «false», а вот вызов

```
if (is_digit(c)) ...
```

не оставляет сомнений в том, что результат будет истиной, если аргумент – число, и ложью – в противном случае.

Глобальные переменные

Следует отказаться от использования глобальных переменных при выполнении лабораторных работ. Данные, необходимые для вычисления результата, передаются как входные параметры функции. Исключение составляют константы, которые влияют на всю логику работы программы.

```
const double pi = 3.1415926;
const unsigned int n = 5; // размер матрицы

void output_matrix(const double M[n][n]) {
	unsigned int i, j;
	for (i=0; i<n; i++)	{
		for (j=0; j<n; j++)
			cout<<M[i][j] <<" ";
		cout << endl;
	}
}

int main()
{
	double M[n][n];
	...
	output_matrix(M);
	return 0;
}
```

Тип переменной

Все переменные имеют тип, который определяет множество значений этой переменной. При выборе типа для переменных с числовыми значениями следует учитывать:

- может ли содержать переменная только целые числа или вещественные;
- являются значения знаковыми или беззнаковыми (для целых чисел);
- минимально возможный размер памяти.

Например, индекс элементов массива является целым беззнаковым числом и не может принимать дробные значения. Если в массиве не более 255 элементов, то следует использовать тип «unsigned char» иначе «unsigned int». Среднее значение целых чисел не является целым числом, поэтому следует использовать тип «double».

Неиспользуемые переменные

В программе должны отсутствовать неиспользуемые локальные и глобальные переменные.

Лишние операторные скобки

В программе должны отсутствовать лишние операторные скобки. Если внутри цикла или условного оператора находится один оператор, то его не следует заключать в операторные скобки. Например, фрагмент следующая функция

```
double output_matrix(const double M[5][4])
{
	unsigned int i, j;
	double s = 0;
	for (i=0; i<5; i++)
	{
		for (j=0; j<4; j++)
```

```
            {
                  s += M[i][j];
            }
      }
      return s;
}
```

Может быть записана так

```
double output_matrix(const double M[5][4])
{
      unsigned int i, j;
      double s = 0;
      for (i=0; i<5; i++)
            for (j=0; j<4; j++)
                  s += M[i][j];
      return s;
}
```

Спецификатор const

Если в программе имеются переменные, значения которых не требуется изменять, то такие переменные должны быть объявлены со спецификатором «const». Например:

```
int main()
{
      const double pi = 3.1415926;
      double alpha;
      cout << "Введите угол в градусах: ";
      cin >> alpha;
      cout << "Угол в радианах: " << alpha*pi/180 <<endl;
      return 0;
}
```

Все входные параметры функций (в т.ч. передаваемые по ссылкам), также должны быть объявлены со спецификатором «const».

```
double degree2radian(const double angle)
{
      return angle*3.1415926/180;
}
```

Такой код будет более надёжным. Даже если программист опечатается или случайно ошибётся программа не откомпилируется, что позволит быстро найти ошибку.

Библиотека ввода-вывода

В качестве библиотеки ввода-вывода следует использовать потоковую библиотеку ввода-вывода «iostream». Лабораторные работы, выполненные с помощью библиотеки «stdio» (за исключением работы №8, где нужно выполнить работу с использованием двух библиотек) к защите не принимаются.

Ошибки и предупреждения на этапе компиляции

Предоставляемая к сдаче программа должна компилироваться и не содержать ни одной ошибки и ни одного предупреждения, как это показано на рисунке 1.

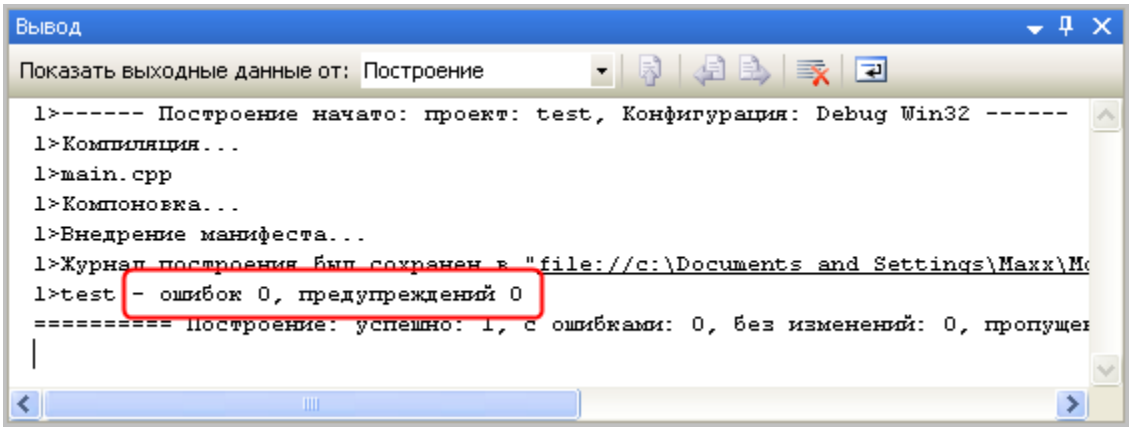


Рис. 1. Окно вывода результатов компиляции проекта

Ввод-вывод

Весь ввод-вывод должен происходить внутри функции «main». Не допускается использование операторов ввода-вывода в отдельных функциях за исключением функций, которые специально предназначены для ввода и вывода информации (например, ввод и вывод значений элементов матрицы).

Это позволяет использовать алгоритм решения задачи вне зависимости от того как получены входные данные (ручной ввод, чтение из файла, генератор случайных чисел, передача по сети и т.д.) и отображены выходные (на экране, в файле, на принтере и т.д.).

Сообщения об ошибках выводятся в поток cerr

Все предупреждения и ошибки выводятся в поток cerr. Более подробно об этом потоке можно прочитать в книгах в разделе «потоковый ввод-вывод» или аналогичных по названию.

Контроль вводимых значений

Программа должна осуществлять проверку значений входных данных. Если вводимое значение не соответствует области допустимых значений (ОДЗ) или формату, то:

1. в функции main осуществляется повторный ввод этого значения;
2. подпрограмма, решающая текущую задачу, сигнализирует об ошибке. Делается это, например, с помощью возврата кода ошибки[1] и последующим выводом сообщения в функции main.

Рассмотрим следующую программу:

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

/*
	Подпрограмма вычисления логарифма по основанию a числа c.
	a ^ b = c
	b = log(a, c)
	Входные данные:
		a - основание логарифма (a > 0)
		c - число (c >= 0)
```

[1] Допускается, если это уместно, использовать исключительные ситуации (для тех, кто знаком с этим механизмом обработки ошибок).

```
      Выходные данные:
            b - логарифм по основанию a числа c
*/
double logAny(double a, double c)
{
      // проверка ОДЗ
      if (!(a> 0 &&c>= 0))
            return numeric_limits<double>::quiet_NaN();
      return log(c) / log(a);
}

int main()
{
      setlocale(LC_ALL, "rus");
      // Ввод a и c
      double a;
      cout <<"Введите основание логарифма (a > 0): ";
      cin >> a;
      while (!cin.good() || !(a > 0))
      {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(),'\n');
            cout<<"Введено некорректное значение. Введите основание логарифма
(a> 0): ";
            cin>>a;
      }

      double c;
      cout<<"Введите число, логарифм которого требуется вычислить (c>= 0): ";
      cin >> c;
      while (!cin.good() || !(c >= 0))
      {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(),'\n');
            cout <<"Введено некорректное значение. Введите число, логарифм
которого требуется вычислить (c >= 0): ";
            cin >> c;
      }

      // вычислить логарифм по заданному основанию
      double b = logAny(a, c);

      // вывести результат
      if (_isnan(b))
            cerr <<"Произошла ошибка при вычислении!"<< endl;
      else
            cout <<"log(a,c)=log("<< a <<","<< c <<")="<< b << endl;
      return 0;
}
```

В данной программе производится две проверки. Первая проверка в main позволяет повторить ввод значений. Вторая проверка сигнализирует о неправильных входных данных. Может показаться, что такой подход избыточен и достаточно только одной проверки. Но не следует забывать, что пользователь может просто опечататься при вводе информации. Также не

следует забывать, что данная функция может быть использована в других проектах, где проверки в функции main может и не быть.

В приведённом примере функция возвращала специальное значение, чтобы сигнализировать об ошибке. Такой подход используется, как правило, для математических функций. Чаще поступают следующим образом. Подпрограмма возвращает несколько значений: первое значение – флаг успешного выполнения, а второе и последующие – сам результат. Например:

```
#include <clocale>
#include <iostream>
#include <fstream>
using namespace std;
/*    Получить размер файла
      Входные данные:   filename - имя файла
      Выходные данные:  tf - флаг успешного завершения
                        size – размер файла           */
bool getFileSize(const char* filename, size_t &size)
{
      // открытьфайл
      ifstream f;
      f.open(filename, ios_base::in | ios_base::binary);
      if (!f.is_open())
            return false;

      // переместить указатель чтения на конец файла
      f.seekg(0, ios_base::end);
      if (f.fail())
      {
            f.close();
            return false;
      }
      // получить текущую позицию = количеству байт в файле
      size = static_cast<size_t>( f.tellg() );
      f.close();
      return true;
}

int main()
{
      setlocale(LC_ALL, "rus");
      size_t size;
      const char filename[] = "c:\\Windows\\notepad.exe";
      if (!getFileSize(filename, size))
            cerr <<"Не удалось вычислить размер файла '"<< filename <<"'"<<
endl;
      else
       cout <<"Размер файла '"<< filename <<"' равен "<< size <<" байт"<<
endl;
      return 0;
}
```

Преобразование значений в стиле C++

В выполняемых работах нежелательно использование преобразование значений в стиле C:

```
int i = 1000;
```

```
char c = (char) i; // преобразование в стиле C
```

Аналогичный фрагмент, но с преобразованием в стиле C++:

```
int i = 1000;
char c = static_cast<char>(i); // преобразование в стилеC++
```

Фактически имеется 4 оператора преобразования, но для выполнения лабораторных работ достаточно оператора static_cast. Другие операторы не запрещается использовать.

Одна из причин введения такого ограничения состоит в том, что при использовании объектно-ориентированного программирования преобразование в стиле C может привести к трудноуловимым ошибкам. Преобразования в стиле C++ сразу видно, их легко найти во всём проекте (проектах) и выявить потенциальные места ошибок.

Комментарии и псевдокод

Описание в виде псевдокода должно быть трансформировано в комментарии программы. Ниже приведён псевдокод

```
// Печать табулированных значений функции sqrt(|X| - 2) с учётом ОДЗ
      // Ввести входные данные
      // Для каждого значения аргумента
            // Если он входит в область допустимых значений
            // Вычислить значение функции
            // И вывести его
```

и соответствующий ему текст программы

```
// Печать табулированных значений функции sqrt(|X| - 2) с учётом ОДЗ
#include <iostream>
using namespace std;
#include <cmath>

int main()
{
      double X, Xn, dx, Xk, Y;
      // Ввести входные данные
      cout<<"Введите Хнач, шаг, Хкон: ";
      cin>>Xn>>dx>>Xk;
      // Для каждого значения аргумента
      for (X=Xn; X<=Xk; X+=dx)
      {
            // Если он входит в область допустимых значений
            if (-2 <X&&X< 2) continue;
            // Вычислить значение функции
            Y = sqrt(abs(X) - 2);
            // И вывести его
            cout<<"sqrt(|"<<X<<"| - 2) = "<<Y<<endl;
      }
      return 0;
}
```

Общие требования к содержанию отчётов

В методические указания включены задания на 8 лабораторных работ, содержанием которых является разработка программы на языке C/C++. Процесс разработки программы студент должен отразить в письменном отчете о выполнении лабораторной работы. Отчет должен состоять из следующих разделов:

1. Цель работы.
2. Задание.
3. Описание созданных функций.
4. Текст программы.
5. Пример выполнения программы.
6. Анализ результатов и выводы.

Титульный лист должен быть оформлен по правилам оформления нормативной документации. Пример оформления титульного листа представлен в приложении 2. Бланк оформления можно скачать по ссылке http://guap.ru/guap/standart/otch_lab_43.rtf. Пример оформления отчёта (шаблон) представлен в приложении 3.

Как минимум, первые три пункта отчета следует подготовить и обсудить с преподавателем до написания текста программы (см. «Универсальность функций»). Даже если студент выполнил лабораторную работу полностью, преподаватель смотрит первые три пункта и проверяет их правильность. Если имеются замечания к ним, то студент должен исправить эти замечания с последующей переделкой программного кода. Такой подход учит студента сперва проектировать, а затем кодировать.

Во втором разделе не следует приводить непосредственное задание варианта (как правило, студенты используют функцию «скопировать и вставить»). Надо доработать задание так, чтобы оно содержало постановку задачи безотносительно варианта. В дополнение к тексту из настоящих методических указаний следует внести описание реакции будущей программы на некоторые неоговоренные в задании значения исходных данных, в том числе на некорректные с точки зрения постановки задачи значения.

В третьем разделе для каждого действия программы приводится описание:

- имени функции;
- назначение функции;
- перечень входных[2] и выходных[3] данных;
- перечень побочных эффектов[4];
- несколько вариантов тестовых данных;
- прототип функции;
- описание алгоритма с использованием псевдокода и блок-схем;

[2] Входными данными называется минимальный набор параметров-переменных, которые необходимы для вычисления результата. Глобальные константы и переменные также являются входными данными, и при описании таких параметров делается соответствующая пометка, что они глобальные.

[3] Выходными данными называются набор параметров-переменных, в которых функция возвращает результат. Глобальные переменные, значения которых изменяет функция, также являются выходными данными, и при описании таких параметров делается соответствующая пометка, что они глобальные.

[4] Как правило, функция возвращает результат своей работы с помощью оператора return. Но помимо вычисления результата функция может осуществлять дополнительный действия, например, присваивать значения глобальным переменным, удалять файл на диске, очищать экран, проигрывать песни и т.д. Такие действия функции, помимо вычисления основного результата, и называются побочным эффектом. Говоря более строго, побочный эффект – любое изменение переменных вне функции (например, глобальные переменные), а также внешнего окружения (удаление файла, очистка экрана, проигрывание песен и т.д.) функцией.

Тестовые данные содержат несколько вариантов значений входных параметров с соответствующими им значениями выходных, вычисленные вручную. Часто студенты копируют выходные значения из результатов выполнения программы. Если в программе имеется ошибка, то тестовые данные тоже содержат ошибки, что недопустимо. Поэтому их следует вычислять вручную до написания программного кода.

Основное требование к описанию алгоритма заключается в том, чтобы оно было более подробным описанием процесса решения задачи, чем постановка задачи из п.1, но менее подробным, чем текст программы. В описании должна найти отражение основная идея решения поставленной задачи. Описание приводится в виде блок-схемы и псевдокода для каждого действия программы.

Листинг программы четвёртого раздела должен соответствовать требованиям из раздела «Оформление кода».

При подготовке рисунков пятого раздела рекомендуется использовать нажатие клавиш Alt + PrtSc, одновременное нажатие которых позволяет скопировать в буфер обмена Windows текущее окно (после его рисунок вставляетс в отчёт).

В заключительном разделе приводится критический анализ проделанной работы с указанием достоинств и недостатков разработанного алгоритма решения задачи и его программной реализации.

Лабораторная работа 1. Вычисление математических выражений

Цель работы

Целью работы является вычисление сложных математических выражений, а также отладка программы для поиска ошибок.

Задание на лабораторную работу

Напишите программу для расчёта двух выражений. Предварительно подготовьте тестовые примеры по обеим формулам (в excel или с помощью калькулятора; результат вычисления по первой формуле должен совпадать со второй). Значение параметров тригонометрических функций должны задаваться пользователем в градусах.

Примечание 1: список математических функций библиотеки C++ приведён в «Приложение 7. Перечень функций стандартной библиотеки C++».

Примечание 2: сведения об отладке приводятся в «Приложение 9. Отладка кода проекта в среде Visual C++».

Варианты заданий

Вариант 1

$$z_1 = 2\sin^2(3\pi - 2\alpha)\cos^2(2\pi + 2\alpha)$$
$$z_2 = \frac{1}{4} - \frac{1}{4}\sin\left(\frac{5}{2}\pi - 8\alpha\right)$$

Вариант 2

$$z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha$$
$$z_2 = 2\sqrt{2}\cos\alpha \cdot \sin\left(\frac{\pi}{4} + 2\alpha\right)$$

Вариант 3

$$z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha}$$
$$z_2 = 2\sin\alpha$$

Вариант 4

$$z_1 = \frac{\sin\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha - \cos 3\alpha + \cos 5\alpha}$$
$$z_2 = \mathrm{tg}3\alpha$$

Вариант 5

$$z_1 = 1 - \frac{1}{4}\sin^2 2\alpha + \cos 2\alpha$$
$$z_2 = \cos^2\alpha + \cos^4\alpha$$

Вариант 6

$$z_1 = \cos\alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha$$
$$z_2 = 4\cos\frac{\alpha}{2} \cdot \cos\frac{5}{2}\alpha \cdot \cos 4\alpha$$

Вариант 7

$$z_1 = \cos^2\left(\frac{3}{8}\pi - \frac{\alpha}{4}\right) - \cos^2\left(\frac{11}{8}\pi + \frac{\alpha}{4}\right)$$
$$z_2 = \frac{\sqrt{2}}{2}\sin\frac{\alpha}{2}$$

Вариант 8

$$z_1 = \cos^4 x + \sin^2 y + \frac{1}{4}\sin^2 2x - 1$$
$$z_2 = \sin(y + x) \cdot \sin(y - x)$$

Вариант 9

$$z_1 = (\cos\alpha - \cos\beta)^2 - (\sin\alpha - \sin\beta)^2$$
$$z_2 = -4\sin^2\frac{\alpha - \beta}{2} \cdot \cos(\alpha + \beta)$$

Вариант 10

$$z_1 = \frac{\sin\left(\frac{\pi}{2} + 3\alpha\right)}{1 - \sin(3\alpha - \pi)}$$
$$z_2 = \operatorname{ctg}\left(\frac{5}{4}\pi + \frac{3}{2}\alpha\right)$$

Вариант 11

$$z_1 = \frac{1 - 2\sin^2\alpha}{1 + \sin 2\alpha}$$
$$z_2 = \frac{1 - \operatorname{tg}\alpha}{1 + \operatorname{tg}\alpha}$$

Вариант 12

$$z_1 = \frac{\sin 4\alpha}{1 + \cos 4\alpha} \cdot \frac{\cos 2\alpha}{1 + \cos 2\alpha}$$
$$z_2 = \operatorname{ctg}\left(\frac{3}{2}\pi - \alpha\right)$$

Вариант 13

$$z_1 = \frac{\sin\alpha + \cos(2\beta - \alpha)}{\cos\alpha - \sin(2\beta - \alpha)}$$
$$z_2 = \frac{1 + \sin 2\beta}{\cos 2\beta}$$

Вариант 14

$$z_1 = \frac{\cos\alpha + \sin\alpha}{\cos\alpha - \sin\alpha}$$
$$z_2 = \operatorname{tg}2\alpha + \sec 2\alpha$$

Вариант 15

$$z_1 = \frac{\sqrt{2b + 2\sqrt{b^2 - 4}}}{\sqrt{b^2 - 4} + b + 2}$$
$$z_2 = \frac{1}{\sqrt{b + 2}}$$

Вариант 16

$$z_1 = \frac{x^2 + 2x - 3 + (x+1)\sqrt{x^2 - 9}}{x^2 - 2x - 3 + (x-1)\sqrt{x^2 - 9}}$$
$$z_2 = \sqrt{\frac{x+3}{x-3}}$$

Вариант 17

$$z_1 = \frac{\sqrt{(3m+2)^2 - 24m}}{3\sqrt{m} - \frac{2}{\sqrt{m}}}$$
$$z_2 = \sqrt{m}$$

Вариант 18

$$z_1 = \left(\frac{a+2}{\sqrt{2a}} - \frac{a}{\sqrt{2a}+2} + \frac{2}{a-\sqrt{2a}}\right) \cdot \frac{\sqrt{a} - \sqrt{2}}{a+2}$$
$$z_2 = \frac{1}{\sqrt{a} + \sqrt{2}}$$

Вариант 19

$$z_1 = \left(\frac{1 + a + a^2}{2a + a^2} + 2 - \frac{1 - a + a^2}{2a - a^2}\right)^{-1} (5 - 2a^2)$$
$$z_2 = \frac{4 - a^2}{2}$$

Вариант 20

$$z_1 = \frac{(m-1)\sqrt{m} - (n-1)\sqrt{n}}{\sqrt{m^3 n} + nm + m^2 - m}$$
$$z_2 = \frac{\sqrt{m} - \sqrt{n}}{m}$$

Вариант 21

$$z_1 = \frac{a-2}{\mathrm{a}^2} + \frac{\sqrt{\mathrm{a}^2 - 4\mathrm{a}\sqrt{\mathrm{a}} + 4\mathrm{a}}}{\sqrt{\mathrm{a} - 2\sqrt{\mathrm{a}}}}$$
$$z_2 = \frac{1}{\mathrm{a}} - \frac{2}{\mathrm{a}^2} + \sqrt{\mathrm{a} - 2\sqrt{\mathrm{a}}}$$

Вариант 22

$$z_1 = \mathrm{a}^{\log_{\mathrm{a}}(\mathrm{b}+2)} + \log_{\mathrm{b}}(\mathrm{ab})$$
$$z_2 = \mathrm{b} + 3 + \frac{1}{\log_{\mathrm{a}} \mathrm{b}}$$

Лабораторная работа 2. Определение попадания точки в область

Цель работы

Целью работы является изучение логических операций типа НЕ, И, ИЛИ.

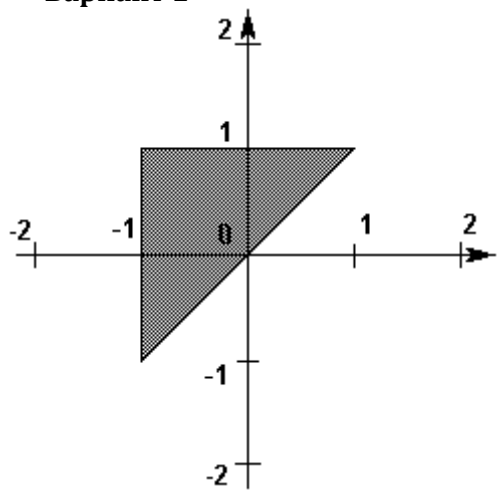
Задание на лабораторную работу

Написать программу, которая для вводимых координат точек(x, y)определяет, попадает ли точка в заштрихованную область на рисунке, который соответствует индивидуальному варианту. Попадание на границу области считать попаданием в область.

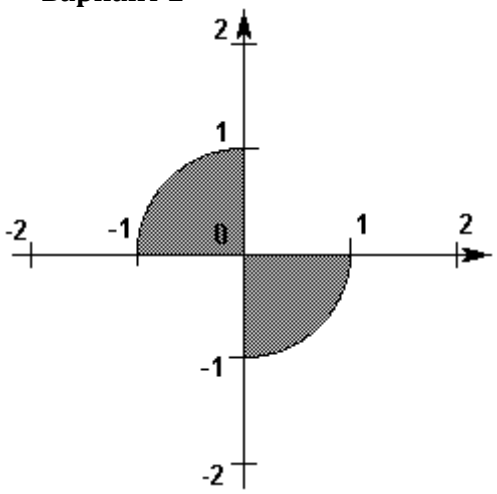
Примечание: для проверки попадания точки в область следует использовать один условный оператор с несколькими условиями.

Варианты заданий

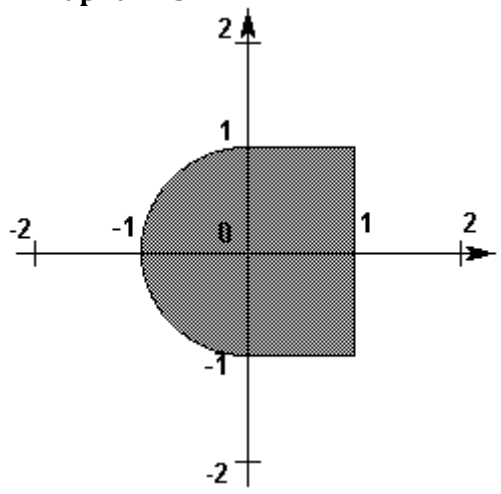
Вариант 1



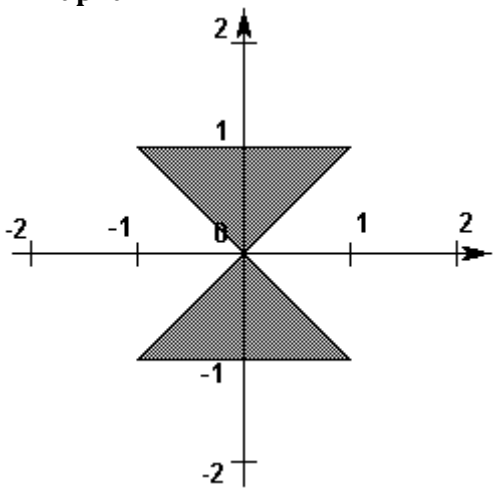
Вариант 2



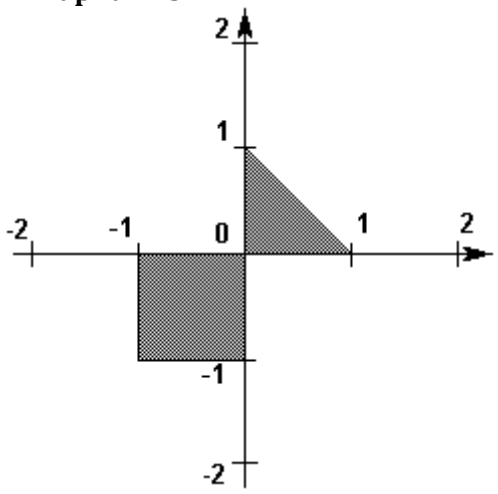
Вариант 3



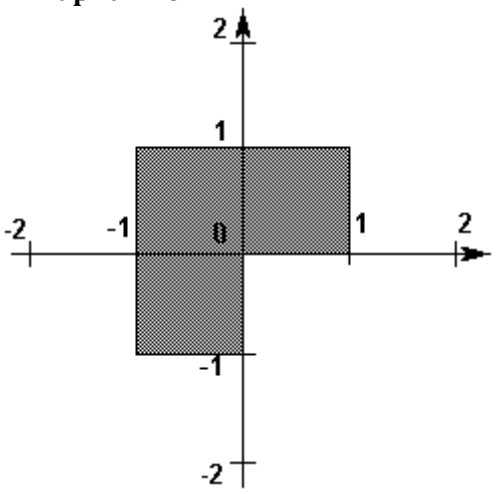
Вариант 4



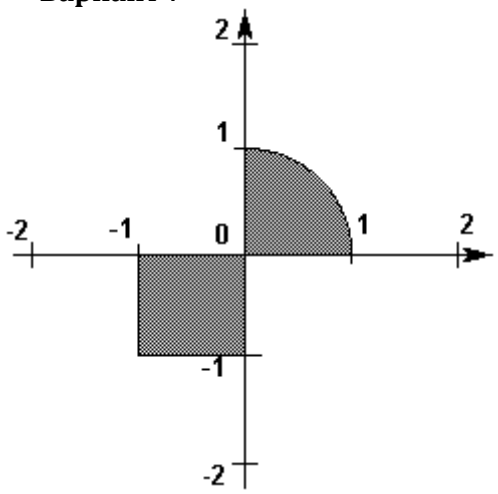
Вариант 5



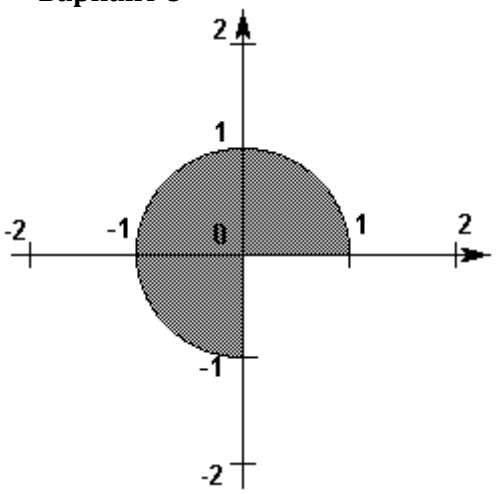
Вариант 6



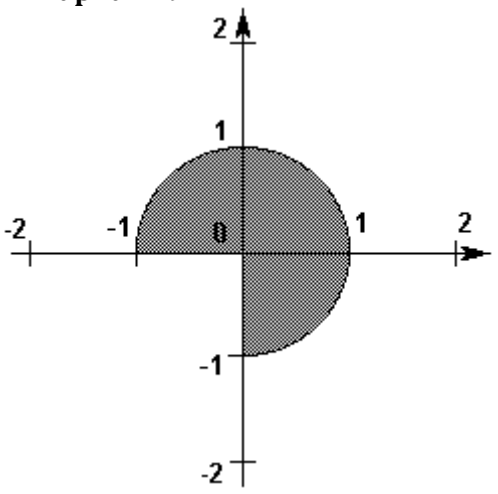
Вариант 7



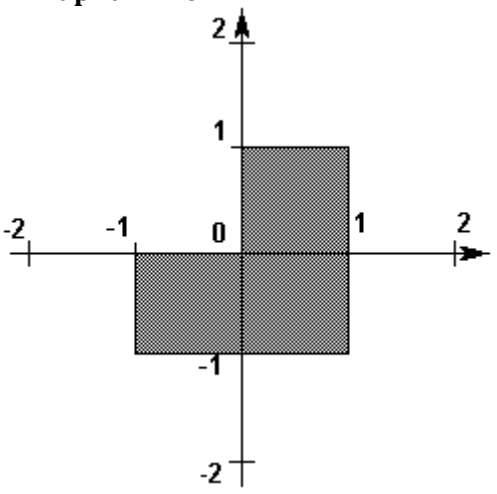
Вариант 8



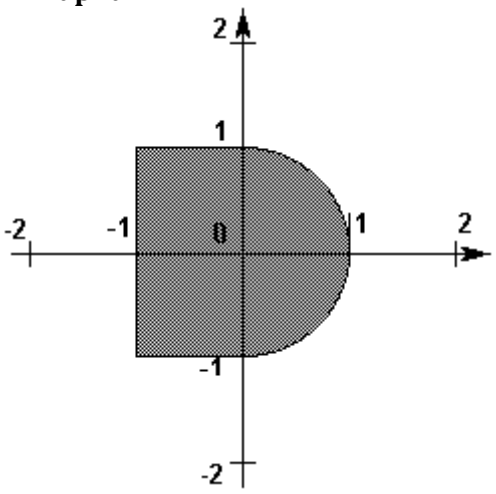
Вариант 9



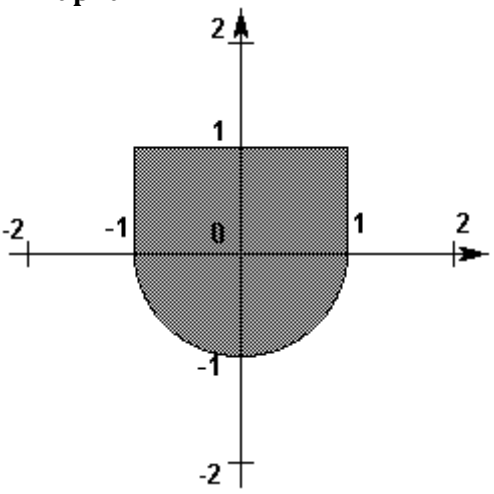
Вариант 10



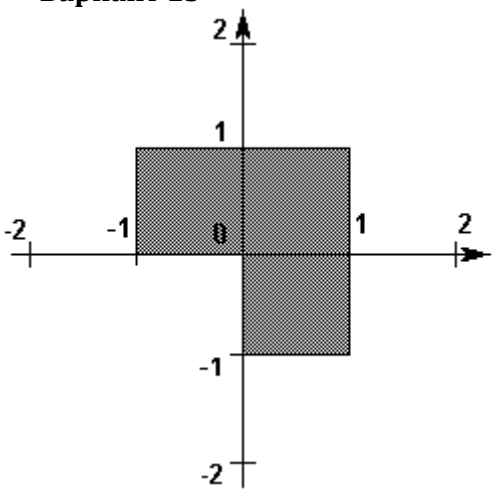
Вариант 11



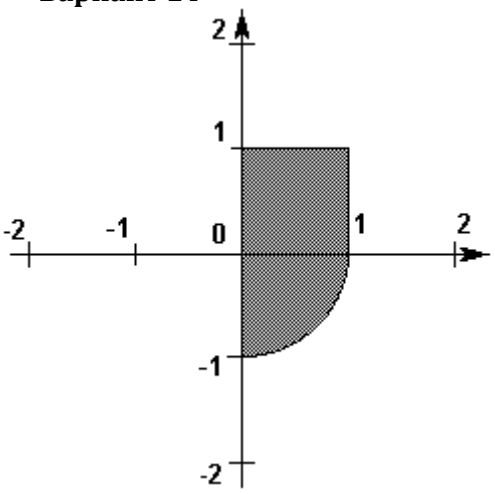
Вариант 12



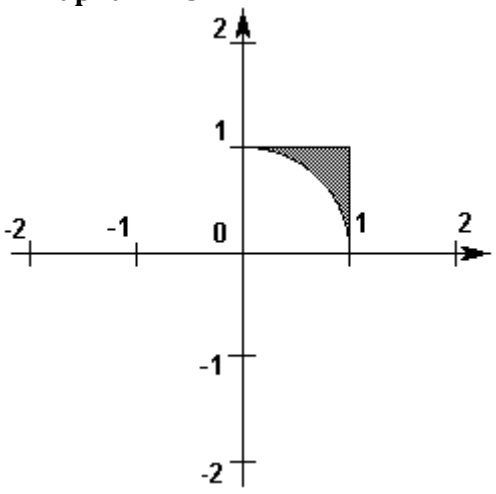
Вариант 13



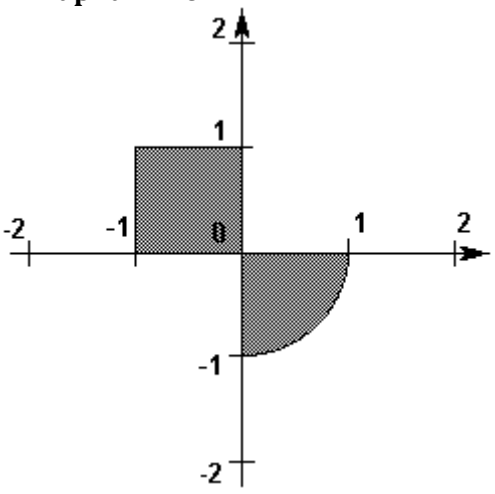
Вариант 14



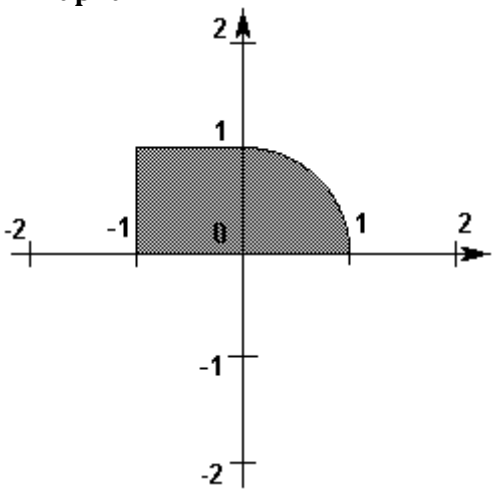
Вариант 15



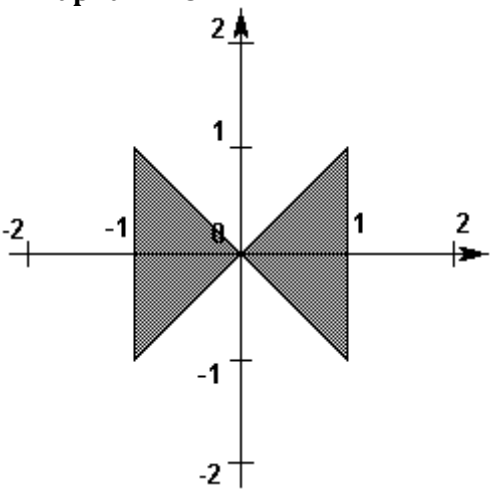
Вариант 16



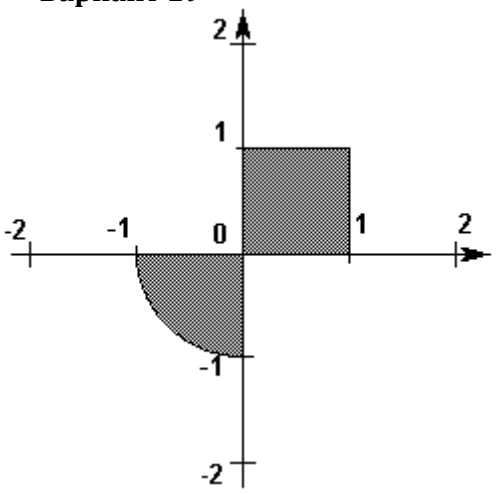
Вариант 17



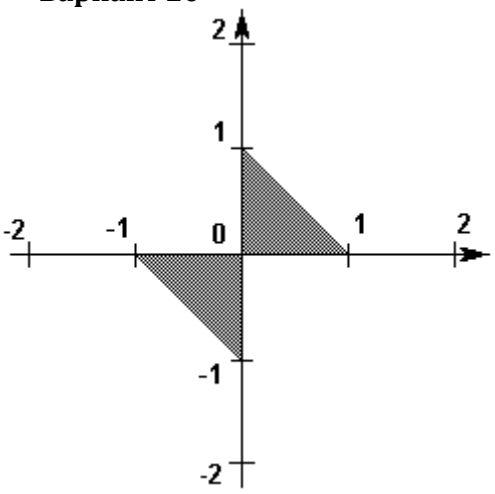
Вариант 18



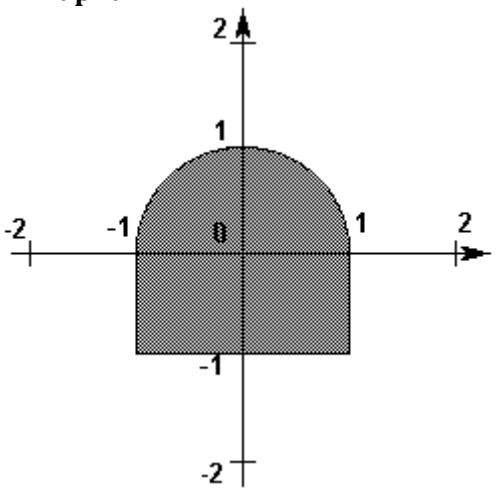
Вариант 19



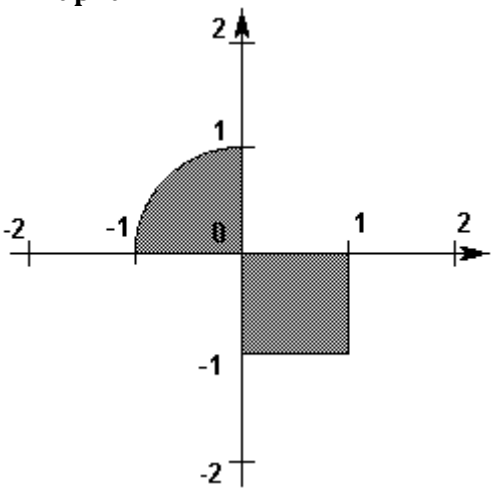
Вариант 20



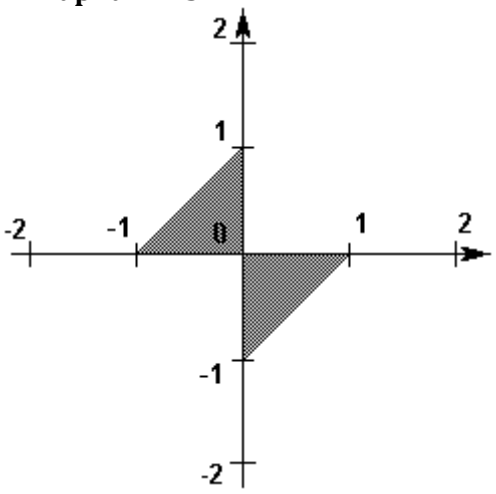
Вариант 21



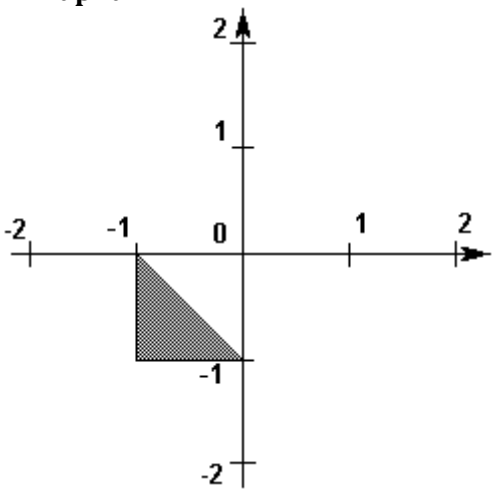
Вариант 22



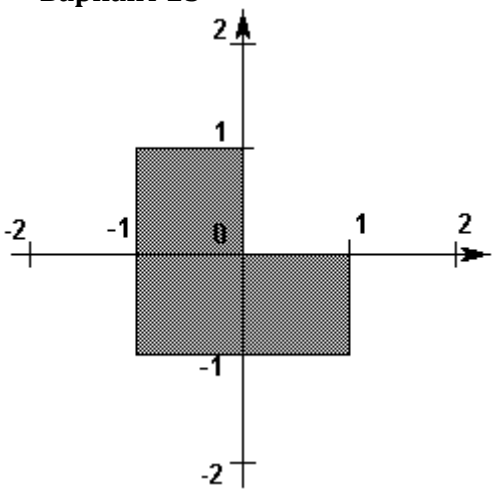
Вариант 23



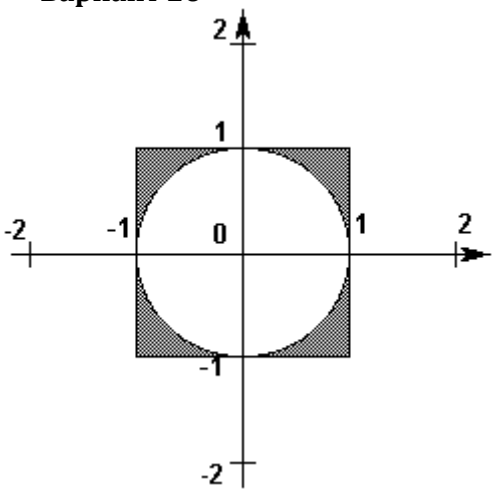
Вариант 24



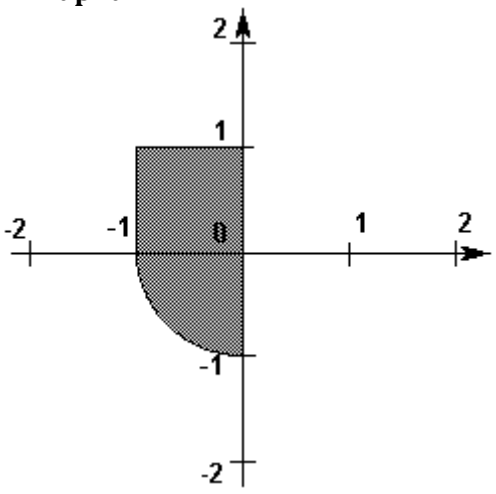
Вариант 25



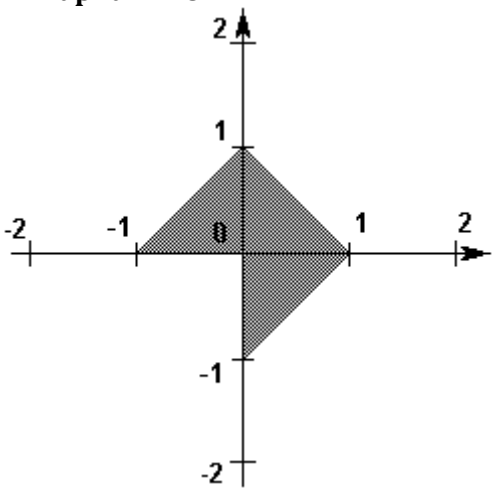
Вариант 26



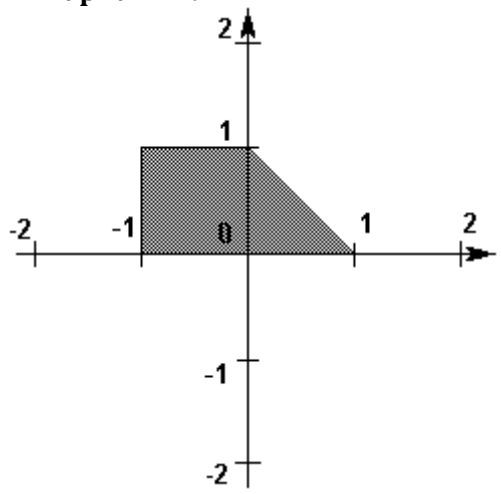
Вариант 27



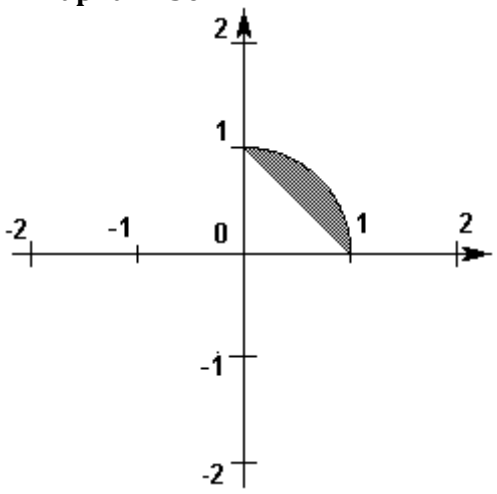
Вариант 28



Вариант 29



Вариант 30



Лабораторная работа 3. Программирование поразрядных операций

Цель работы

Целью работы является изучение поразрядных операций типа НЕ, И, ИЛИ, исключающее ИЛИ и операций сдвига.

Задание на лабораторную работу

Составить две программы, первая из которых вводит составные части структуры данных, приведённой в индивидуальном варианте, как десятичные числа и формирует из них заданную упакованную структуру как 16-ричное число. Вторая программа вводит упакованную структуру как 16-ричное число и выводит значения отдельных её составных частей как десятичные числа.

Программы должны быть оформлены как решение с двумя проектами (см. «Приложение 8. Полезные советы по работе в среде VisualC++ 2008» на стр. 82).

Для чтения и записи в потоки ввода/вывода следует использовать манипуляторы dec и hex.

Пример выполнения лабораторной работы

а) ввод десятичных чисел и формирование из них заданной упакованной структуры как 16-ричного числа

Рассмотрим в качестве примера следующую упакованную структуру с полями A, B, C:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	A	A	A	A	B	B	B	B	B	C	C	C	C	C	C	C

где A, B, C – некоторые значения.

В первой программе пользователь вводит три числа A, B, C (как десятичные числа). Программа должна проверять диапазон вводимых значений. Число A занимает 4 двоичных разряда; следовательно, его значение находится в диапазоне$[0; 2^4 - 1] = [0; 15]$. Число B занимает 5 двоичных разрядов; следовательно, его значение находится в диапазоне$[0; 2^5 - 1] = [0; 31]$. Число C занимает 7 двоичных разрядов; следовательно, его значение находится в диапазоне $[0; 2^7 - 1] = [0; 127]$.

Далее необходимо (с помощью отдельной функции) записать значения этих чисел в соответствующие разряды числа-структуры X. Число-структура – это «контейнер», который хранит значения чисел A, B, C в своих разрядах. В качестве типа X может использоваться, например, тип $short$ (занимает, как правило, 16 бит) или int (занимает, как правило, 32 бита).

Запись в разряды осуществляется с помощью поразрядных операций $\ll$ и $|$. Рассмотрим следующий пример:

```
// вариант 1
unsigned short x=0;
x = c;
x = b <<  7 | x;
x = a << 12 | x;

// вариант 2 (более короткий)
unsigned short x = c | b <<  7 | a << 12;
cout << hex << x; // вывод числа в шестнадцатеричном виде
```

Если $c = 100_{10} = 1100100_2$, то

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
c	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
x=c	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

Если введено слишком большое число, которое не помещается в разрядную сетку из 5 разрядов ($b > 31 = 11111_2$), то число необходимо ввести повторно.

Если $b = 29_{10} = 11101_2$, то

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
b	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1
$b \ll 7$	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0
x	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
$(b \ll 7) \mid x$	0	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0

За счёт операции $b|x$ происходит запись соответствующих разрядов.

Если $a = 9_{10} = 1001_2$, то

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
a	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
$a \ll 12$	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
x	0	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0
$(a \ll 12) \mid x$	1	0	0	1	1	1	1	0	1	1	1	0	0	1	0	0

Таким образом, можно установить заданные разряды числа.

б) из заданной упакованной структуры, представленной как 16-ричное число выделить значения отдельных её составных частей (в виде десятичных чисел).

Чтение данных из разрядов осуществляется аналогичным образом с помощью $\gg$ и &. Суть второй программы состоит в том, чтобы ввести значение числа x (в шестнадцатеричном виде) и разбить его на составляющие (в данном случае A, B, C).

Упакованная структура z

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	A	A	A	A	B	B	B	B	B	C	C	C	C	C	C	C

$z = 51F8_{16} = 0101\ 0001\ 1111\ 1000_2$

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	0	1	0	1	0	0	0	1	1	1	1	1	1	0	0	0

Найдем, например, чему будет равно значение составляющей B в десятичном виде.

```
unsigned short z=0x51F8;
short b;
b = (z>>7)&0x1F;
cout << dec << "b=" << b << endl;
cout << hex << x; // вывод числа в шестнадцатеричном виде
```

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
z	0	1	0	1	0	0	0	1	1	1	1	1	1	0	0	0
z>>7	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1
$1F_{16} = 31_{10} = 11111_2$	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
z>>7&31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Варианты заданий

Вариант 1

Физический адрес на диске представляется в виде:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	H	H	T	T	T	T	T	T	T	T	T	S	S	S	S	S

где $H \ldots H$ – номер головки, $T \ldots T$– номер дорожки, $S \ldots S$–номер сектора.

Вариант 2

Слово состояния программы в вычислительной системе представляется в виде:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	C	C	C	C	E	I	P	0	0	0	K	K	K	0	0	0

где $C \ldots C$ – маски каналов (0-3),E – маска внешнего прерывания,I–маска внутреннего прерывания, P– маска программного прерывания,$K \ldots K$– ключ защиты памяти.

Вариант 3

Слово состояния канала в вычислительной системе представляется в виде:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	N	N	N	N	E	C	C	C	C	C	C	C	C	C	0	P

где $N \ldots N$ – номер канала,E – признак ошибки, $C \ldots C$– код причини прерывания, P– признак завершения программы в канале.

Вариант 4

Формат команды загрузки/сохранения в вычислительной системе имеет вид:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	1	1	1	0	0	0	1	D	R	R	R	R	A	A	A	A

гдеD – направление передачи (0 - в регистр, 1- в память), $R \ldots R$ – первый операнд (регистр),$A \ldots A$– второй операнд (адрес).

Вариант 5

Слово состояния оборудования для вычислительной системы представляется в виде:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	P	P	P	0	F	F	0	H	H	H	H	0	0	V	V	V

где $P \ldots P$ – количество принтеров, $F \ldots F$– количество гибких дисков, $H \ldots H$–тип жёсткого диска, $V \ldots V$ – тип видеоадаптера.

Вариант 6

Формат команды сдвига в вычислительной системе имеет вид:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	1	1	0	1	T	T	D	N	N	N	N	N	N	N	N	N

где $T \ldots T$ – тип сдвига, D– направление сдвига, $N \ldots N$–количество разрядов сдвига.

Вариант 7

Блок управления буфером кеша в вычислительной системе представляется в виде:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	S	S	S	0	D	0	F	0	B	B	B	B	B	B	B	B

где $S \ldots S$ – код системной области или 000, D– признак «грязного» буфера, F – признак свободного буфера, $B \ldots B$–номер блока, который отображён в буфере.

Вариант 8

Элемент списка безопасности объекта в вычислительной системе представляется в виде:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	U	U	U	U	U	U	U	U	R	W	E	0	0	0	0	X

где $U \ldots U$ – идентификатор пользователя, R– право читать, W – право писать, E – право выполнять программный код, X–явный запрет доступа.

Вариант 9

Формат команды канала в система ввода-вывода имеет вид:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	O	O	O	O	O	C	D	N	N	N	N	N	N	N	N	N

где $O \ldots O$ – код операции, C– признак цепочки команд, D – признак цепочки данных, $N \ldots N$– количество байтов для передачи.

Вариант 10

Элемент профиля пользователя в вычислительной системе представляется в виде:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	G	G	G	G	G	G	G	G	S	S	S	S	P	0	0	0

где $G \ldots G$ – идентификатор группы пользователя, $S \ldots S$–код системы, которая загружается для пользователя, P– признак привилегированного пользователя.

Вариант 11

Информация о состоянии устройства в системе ввода-вывода представляется в виде:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	C	C	C	U	U	U	U	U	E	E	E	E	E	E	0	B

где $C \ldots C$ – номер канала, $U \ldots U$– номер устройства в канале, $E \ldots E$–код состояния, B – признак занятости устройства.

Вариант 12

Формат команды сложения в вычислительной системе имеет вид:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	1	1	0	0	0	1	1	0	D	A	A	A	B	B	B	0

где D – тип операндов (0 – байты, 1 - слова), $A \ldots A$– регистр - 1-й операнд, $B \ldots B$–регистр - 2-й операнд.

Вариант 13

Формат представления текущей даты в некоторых системных структурах имеет вид:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	D	D	D	D	D	M	M	M	M	M	Y	Y	Y	Y	Y	Y

где $D \ldots D$ – номер дня в месяце, $M \ldots M$– номер месяца в году, $Y \ldots Y$–номер год начиная 1980.

Вариант 14

Формат представления текущего времени в некоторых системных структурах имеет вид:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	S	S	S	S	S	M	M	M	M	M	M	H	H	H	H	H

где $S \ldots S$ – количество секунд, делённых на 2, $M \ldots M$– количество минут, $H \ldots H$–номер часа.

Вариант 15

Дескриптор сегмента для системы виртуальной памяти представляется в виде:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	F	F	F	F	F	F	F	F	R	W	L	L	L	L	L	L

где $F \ldots F$ – номер блока, с которого начинается сегмент, R – доступ для чтения, W – доступ для записи, $L \ldots L$–размер сегмента в блоках.

Вариант 16

Атрибут файла в файловой системе представляется в виде:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	T	T	T	T	A	A	A	0	I	I	I	I	I	I	I	I

где $T \ldots T$ – код типа файла, $A \ldots A$ – код доступа, $I \ldots I$–номер файлового индекса.

Вариант 17

Блок управления памятью в операционной системе имеет формат:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	O	O	O	O	O	O	O	O	0	P	L	L	L	L	L	L

где $O \ldots O$ – идентификатор владельца блока, P – признак программного блока, $L \ldots L$–размер блока.

Вариант 18

Точка изображения на 16-цветном дисплее с размером экрана 64x64 описывается в формате:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	X	X	X	X	X	X	Y	Y	Y	Y	Y	Y	C	C	C	C

где $X \ldots X$ – горизонтальная координата, $Y \ldots Y$ – вертикальная координата, $C \ldots C$–цвет.

Вариант 19

Управляющее слово программируемого таймера имеет формат:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	C	C	F	F	D	D	D	D	D	D	D	D	D	D	D	D

где $C \ldots C$ – номер канала таймера, $F \ldots F$ – форма сигнала, $D \ldots D$–коэффициент деления опорной частоты.

Вариант 20

Информация о критической ошибке на диске представляется в виде:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	O	A	A	R	R	R	0	1	D	D	D	D	D	D	D	D

гдеO – тип операции (0 - чтение, 1 - запись),$A \ldots A$ – код области диска, $R \ldots R$ – возможные реакции на ошибку, $D \ldots D$–номер диска.

Вариант 21

Дескриптор сообщения в системе телекоммуникаций имеет формат:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	N	N	N	N	0	P	P	P	U	U	U	U	U	U	U	U

где $N \ldots N$ – номер узла, из которого пришло сообщение, $P \ldots P$ – приоритет сообщения, $U \ldots U$– идентификатор пользователя - автора сообщения.

Вариант 22

Дескриптор семафора, который защищает пул ресурсов, имеет формат:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	N	N	N	M	M	M	0	L	W	W	W	W	W	W	W	W

где$N \ldots N$ – текущее количество свободных единиц ресурса, $M \ldots M$ – общее количество единиц ресурса,L – признак блокирования семафора (0–разблокирован,1 - блокирован),$W \ldots W$–количество процессов, которые ожидают доступа к ресурсу.

Вариант 23

Заголовок кадра в системе передачи данных имеет формат:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	S	S	S	S	S	S	D	D	D	D	D	D	0	C	C	C

где $S \ldots S$ – адрес источника, $D \ldots D$ – адрес приёмника, $C \ldots C$–управляющий код.

Вариант 24

Заголовок пакета в системе передачи данных имеет формат:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	T	T	T	0	S	S	S	S	L	L	L	L	L	L	L	L

где $T \ldots T$ – тип пакета, $S \ldots S$ – идентификатор источника, $L \ldots L$–длина пакета.

Вариант 25

Поле управления диспетчером кадра в протоколе управления логическим каналом имеет формат:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	1	0	C	C	0	0	0	0	P	N	N	N	N	N	N	N

где$C \ldots C$ – управляющий код, P – признак запроса или завершения(0 - запрос, 1 - завершение),$N \ldots N$–номер последовательности.

Вариант 26

Формат заголовка пакета в системе ретрансляции кадров такой:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	C	C	C	C	C	C	C	C	C	C	R	E	E	E	E	E

где$C \ldots C$ – идентификатор канала передачи, R – признак команды или ответа (0 - команда, 1 - ответ),$E \ldots E$–расширение адреса.

Вариант 27

Формат элемента доступа к объекту в системе безопасности такой:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	0	A	A	A	T	T	M	0	G	G	G	G	G	G	G	G

где $A \ldots A$ – код доступа к объекту из группы PUBLIC,$T \ldots T$ – доступ для чтения и/или доступ для записи, M – модификатор (0 - чтение, 1 - запись), $G \ldots G$–код группы доступа, к которой принадлежит объект.

Вариант 28

Дескриптор массива, который формируется компилятором языка программирования, имеет формат:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	L	L	L	L	0	0	B	B	B	B	B	E	E	E	E	E

где $L \ldots L$ – размер элемента массива, $B \ldots B$ – размер элемента массива, $E \ldots E$–конечный индекс.

Вариант 29

Блок управления сегментом памяти в системе с реальной памятью имеет формат:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	T	T	T	T	T	0	A	0	L	L	L	L	L	L	L	L

где$T \ldots T$ – идентификатор задачи, которой принадлежит сегмент или 0 - для свободного сегмента, A – признак активности задачи(0–неактивна, 1 - активна),$L \ldots L$–длина сегмента.

Вариант 30

Слово состояния устройства в системе ввода-вывода представляется в виде:

№ разряда	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Значение	C	C	C	C	C	0	F	B	N	N	N	N	N	N	N	N

где$C \ldots C$ – код состояния, F – признак ошибки(0–нет ошибки, 1–есть ошибка) , B – признак занятости (0 – свободно, 1 – занято),$N \ldots N$–количество байт, переданных в последней операции.

Лабораторная работа 4. Вычисление кусочной функции

Цель работы

Целью работы является изучение основных управляющих структур программирования и функций.

Задание на лабораторную работу

Вычислить и вывести на экран в виде таблицы значения функции F на интервале $[X_{\text{нач}}, X_{\text{кон}}]$ с шагом dx. Вид функции F определяется индивидуальным вариантом. Коэффициенты a, b, c являются действительными числами. Значения a, b, c, $X_{\text{нач}}$, $X_{\text{кон}}$, dx вводятся с клавиатуры.

Примечание: тестовые данные должны охватывать все ветки функции F.

Варианты заданий

Вариант 1

$$F = \begin{cases} ax^2 + b & \text{при } x < 0 \text{ и } b \neq 0 \\ \dfrac{x-a}{x-c} & \text{при } x > 0 \text{ и } b = 0 \\ \dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 2

$$F = \begin{cases} \dfrac{1}{ax} - b & \text{при } x + 5 < 0 \text{ и } c = 0 \\ \dfrac{x-a}{x} & \text{при } x + 5 > 0 \text{ и } c \neq 0 \\ \dfrac{10x}{c-4} & \text{в остальных случаях} \end{cases}$$

Вариант 3

$$F = \begin{cases} ax^2 + bx + c & \text{при } a < 0 \text{ и } c \neq 0 \\ \dfrac{-a}{x-c} & \text{при } a > 0 \text{ и } c = 0 \\ a(x+c) & \text{в остальных случаях} \end{cases}$$

Вариант 4

$$F = \begin{cases} -ax - c & \text{при } c < 0 \text{ и } x \neq 0 \\ \dfrac{x-a}{-c} & \text{при } c > 0 \text{ и } x = 0 \\ \dfrac{bx}{c-a} & \text{в остальных случаях} \end{cases}$$

Вариант 5

$$F = \begin{cases} a - \dfrac{x}{10+b} & \text{при } x < 0 \text{ и } b \neq 0 \\ \dfrac{x-a}{x-c} & \text{при } x > 0 \text{ и } b = 0 \\ 3x + \dfrac{2}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 6

$$F = \begin{cases} ax^2 + b^2x & \text{при } c < 0 \text{ и } b \neq 0 \\ \dfrac{x+a}{x+c} & \text{при } c > 0 \text{ и } b = 0 \\ \dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 7

$$F = \begin{cases} -ax^2 - b & \text{при } x < 5 \text{ и } c \neq 0 \\ \dfrac{x-a}{x} & \text{при } x > 5 \text{ и } c = 0 \\ -\dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 8

$$F = \begin{cases} -ax^2 & \text{при } c < 0 \text{ и } a \neq 0 \\ \dfrac{a-x}{cx} & \text{при } c > 0 \text{ и } a = 0 \\ \dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 9

$$F=\begin{cases} ax^2+b^2x & \text{при } a<0 \text{ и } x\neq 0 \\ x-\dfrac{a}{x-c} & \text{при } a>0 \text{ и } x=0 \\ 1+\dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 10

$$F=\begin{cases} ax^2-bx+c & \text{при } x<3 \text{ и } b\neq 0 \\ \dfrac{x-a}{x-c} & \text{при } x>3 \text{ и } b=0 \\ \dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 11

$$F=\begin{cases} ax^2+\dfrac{b}{c} & \text{при } x<1 \text{ и } c\neq 0 \\ \dfrac{x-a}{(x-c)^2} & \text{при } x>1.5 \text{ и } c=0 \\ \dfrac{x^2}{c^2} & \text{в остальных случаях} \end{cases}$$

Вариант 12

$$F=\begin{cases} ax^3+b^2+c & \text{при } x<0.6 \text{ и } b+c\neq 0 \\ \dfrac{x-a}{x-c} & \text{при } x>0.6 \text{ и } b+c=0 \\ \dfrac{x}{c}+\dfrac{x}{a} & \text{в остальных случаях} \end{cases}$$

Вариант 13

$$F=\begin{cases} ax^2+b & \text{при } x-1<0 \text{ и } b-x\neq 0 \\ \dfrac{x-a}{x} & \text{при } x-1>0 \text{ и } b+x=0 \\ \dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 14

$$F=\begin{cases} -ax^3-b & \text{при } x+c<0 \text{ и } a\neq 0 \\ \dfrac{x-a}{x-c} & \text{при } x+c>0 \text{ и } a=0 \\ \dfrac{x}{c}+\dfrac{c}{x} & \text{в остальных случаях} \end{cases}$$

Вариант 15

$$F=\begin{cases} -ax^2+b & \text{при } x<0 \text{ и } b\neq 0 \\ \dfrac{x}{x-c}+5.5 & \text{при } x>0 \text{ и } b=0 \\ \dfrac{x}{-c} & \text{в остальных случаях} \end{cases}$$

Вариант 16

$$F=\begin{cases} a(x+c)^2-b & \text{при } x=0 \text{ и } b\neq 0 \\ \dfrac{x-a}{-c} & \text{при } x=0 \text{ и } b=0 \\ a+\dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 17

$$F=\begin{cases} ax^2-cx+b & \text{при } x+10<0 \text{ и } b\neq 0 \\ \dfrac{x-a}{x-c} & \text{при } x+10>0 \text{ и } b=0 \\ \dfrac{-x}{a-c} & \text{в остальных случаях} \end{cases}$$

Вариант 18

$$F=\begin{cases} ax^2+bx^2 & \text{при } x<0 \text{ и } b\neq 0 \\ \dfrac{x-a}{x-c} & \text{при } x>0 \text{ и } b=0 \\ \dfrac{x+5}{c(x-10)} & \text{в остальных случаях} \end{cases}$$

Вариант 19

$$F=\begin{cases} a(x+7)^2-b & \text{при } x<5 \text{ и } b\neq 0 \\ \dfrac{x-ac}{ax} & \text{при } x>5 \text{ и } b=0 \\ \dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 20

$$F=\begin{cases} -\dfrac{2x-c}{cx-a} & \text{при } x<0 \text{ и } b\neq 0 \\ \dfrac{x-a}{x-c} & \text{при } x>0 \text{ и } b=0 \\ -\dfrac{x}{c}-\dfrac{c}{2x} & \text{в остальных случаях} \end{cases}$$

Лабораторная работа 5. Обработка числовых последовательностей

Цель работы

Целью работы является изучение структуры данных одномерный массив.

Задание на лабораторную работу

Задания на лабораторную работу приводятся в каждом варианте. При написании программ можно использовать как динамические, так и нединамические массивы. Размерность последних задаётся именованной константой.

Примечание: массивы, созданные с помощью классов (таких, как например, array или vector из библиотеки STL) использовать запрещается.

Примечание: если использовались операторы динамического выделения памяти, то следует вставить дополнительный код, обнаруживающий утечки памяти. См. раздел «Как обнаружить утечки памяти?». Все утечки памяти должны быть устранены.

Варианты заданий

Вариант 1

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. сумму отрицательных элементов массива;
2. произведение элементов массива, расположенных между максимальным и минимальным элементами.

Упорядочить элементы массива по возрастанию.

Вариант 2

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. сумму положительных элементов массива;
2. произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.

Упорядочить элементы массива по убыванию.

Вариант 3

В одномерном массиве, состоящем из n целых элементов, вычислить:

1. произведение элементов массива с чётными номерами;
2. сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом – все отрицательные (элементы, равные 0, считать положительными).

Вариант 4

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. сумму элементов массива с чётными номерами;

2. сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Сжать массив, удалив из него все элементы, модуль которых не превышает 1. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 5

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. максимальный элемент массива;
2. сумму элементов массива, расположенных до последнего положительного элемента.

Сжать массив, удалив из него все элементы, модуль которых находится в интервале $[a, b]$. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 6

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. минимальный элемент массива;
2. сумму элементов массива, расположенных между первым и последним положительными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом – все остальные.

Вариант 7

В одномерном массиве, состоящем из n целых элементов, вычислить:

1. номер максимального элемента массива;
2. произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечётных позициях, а во второй половине – элементы, стоявшие в чётных позициях.

Вариант 8

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. номер минимального элемента массива;
2. сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом – все остальные.

Вариант 9

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. максимальный по модулю элемент массива;
2. сумму элементов массива, расположенных между первым и вторым положительными элементами.

Преобразовать массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

Вариант 10

В одномерном массиве, состоящем из n целых элементов, вычислить:

1. минимальный по модулю элемент массива;
2. сумму модулей элементов массива, расположенных после первого элемента, равного нулю.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в чётных позициях, а во второй половине – элементы, стоявшие в нечётных позициях.

Вариант 11

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. номер минимального по модулю элемента массива;
2. сумму модулей элементов массива, расположенных после первого отрицательного элемента.

Сжать массив, удалив из него все элементы, величина которых находится в интервале $[a, b]$. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 12

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. номер максимального по модулю элемента массива;
2. сумму модулей элементов массива, расположенных после первого положительного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале $[a, b]$, а потом – все остальные.

Вариант 13

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. количество элементов массива, лежащих в диапазоне $[a, b]$;
2. сумму элементов массива, расположенных после максимального элемента.

Упорядочить элементы массива по убыванию модулей элементов.

Вариант 14

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. количество элементов массива, равных 0;
2. сумму элементов массива, расположенных после минимального элемента.

Упорядочить элементы массива по возрастанию модулей элементов.

Вариант 15

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. количество элементов массива, больших C;

2. произведение элементов массива, расположенных после максимального по модулю элемента.

Преобразовать массив таким образом, чтобы сначала располагались все отрицательные элементы, а потом – все положительные (элементы, равные 0, считать положительными).

Вариант 16

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. количество отрицательных элементов массива;
2. сумму модулей элементов массива, расположенных после минимального по модулю элемента.

Заменить все отрицательные элементы массива их квадратами и упорядочить элементы массива по возрастанию.

Вариант 17

В одномерном массиве, состоящем из n целых элементов, вычислить:

1. количество положительных элементов массива;
2. сумму элементов массива, расположенных после последнего элемента, равного 0.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых не превышает 1, а потом – все остальные.

Вариант 18

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. количество элементов массива, меньших C;
2. сумму целых частей элементов массива, расположенных после последнего отрицательного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, отличающиеся от максимального не более чем на 20%, а потом – все остальные.

Вариант 19

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. произведение отрицательных элементов массива;
2. сумму положительных элементов массива, расположенных после максимального элемента.

Изменить порядок следования элементов в массиве на обратный.

Вариант 20

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. произведение положительных элементов массива;
2. сумму элементов массива, расположенных до минимального элемента.

Упорядочить по возрастанию отдельно элементы, стоящие на чётных местах, и элементы, стоящие на нечётных местах.

Лабораторная работа 6. Обработка числовых матриц

Цель работы

Целью работы является изучение структуры данных двумерный массив.

Задание на лабораторную работу

Задания на лабораторную работу приводятся в каждом варианте. При написании программ можно использовать как динамические, так и нединамические массивы. Размерность последних задаётся именованной константой.

Примечание: массивы, созданные с помощью классов (таких, как например, array или vector из библиотеки STL) использовать запрещается.

Примечание: если использовались операторы динамического выделения памяти, то следует вставить дополнительный код, обнаруживающий утечки памяти. См. раздел «Как обнаружить утечки памяти?». Все утечки памяти должны быть устранены.

Варианты заданий

Вариант 1

Дана целочисленная прямоугольная матрица. Определить:

1. количество строк, не содержащих ни одного нулевого элемента;
2. максимальное из чисел, встречающихся в заданной матрице более одного раза.

Вариант 2

Дана целочисленная прямоугольная матрица.

1. Определить количество столбцов, не содержащих ни одного нулевого элемента.
2. Характеристикой строки целочисленной матрицы назовём сумму её положительных чётных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик.

Вариант 3

Дана целочисленная прямоугольная матрица. Определить:

1. количество столбцов, содержащих хотя бы один нулевой элемент;
2. номер строки, в которой находится самая длинная серия одинаковых элементов.

Вариант 4

Дана целочисленная квадратная матрица. Определить:

1. произведение элементов в тех строках, которые не содержат отрицательных элементов;
2. максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

Вариант 5

Дана целочисленная квадратная матрица. Определить:

1. сумму элементов в тех столбцах, которые не содержат отрицательных элементов;
2. минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

Вариант 6

Дана целочисленная прямоугольная матрица. Определить:

1. сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент;
2. номера строк и столбцов всех седловых точек матрицы.

Примечание. Матрица A имеет седловую точку A_{ij}, если A_{ij} является минимальным элементов в -й строке и максимальным в j-м столбце.

Вариант 7

Для заданной матрицы размером N на N

1. Найти такие k, что k-я строка матрицы совпадает с k-м столбцом.
2. Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.

Вариант 8

Характеристикой столбца целочисленной матрицы назовём сумму модулей его отрицательных нечётных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик.

Найти сумму элементов в тех столбцах, которые содержат хотя бы один отрицательный элемент.

Вариант 9

Соседями элемента A_{ij} в матрице назовём элементы A_{kl} с $i-1 \le k \le i+1, j-1 \le l \le j+1, (k,l) \ne (i,j)$. Операция сглаживания матрицы даёт новую матрицу того же размера, каждый элемент которой получается как среднее арифметическое имеющихся соседей соответствующего элемента исходной матрицы. Построить результат сглаживания заданной вещественной матрицы размером 10 на 10.

В сглаженной матрице найти сумму модулей элементов, расположенных ниже главной диагонали.

Вариант 10

Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей. Подсчитать количество локальных минимумов заданной матрицы размером N на N.

Найти сумму модулей элементов, расположенных выше главной диагонали.

Вариант 11

Коэффициенты системы линейных уравнений заданы в виде прямоугольной матрицы. С помощью допустимых преобразований привести систему к треугольному виду.

Найти количество строк, среднее арифметическое элементов которых меньше заданной величины.

Вариант 12

Уплотнить заданную матрицу, удаляя из неё строки и столбцы, заполненные нулями.

Найти номер первой из строк, содержащих хотя бы один положительный элемент.

Вариант 13

Осуществить циклический сдвиг элементов прямоугольной матрицы на n элементов вправо или вниз (в зависимости от введённого режима). n может быть больше количества элементов в строке или столбце.

Вариант 14

Осуществить циклический сдвиг элементов квадратной матрицы размерности N на N вправо на k элементов таким образом: элементов 1-й строки сдвигаются в последний столбец сверху вниз, из него – в последнюю строку справа налево, из неё – в первый столбец снизу верх, из него – в первую строку; для остальных элементов аналогично.

Вариант 15

Дана целочисленная прямоугольная матрица. Определить номер первого из столбцов, содержащих хотя бы один нулевой элемент.

Характеристикой строки целочисленной прямоугольной матрицы назовём сумму её отрицательных чётных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с убыванием характеристик.

Вариант 16

Упорядочить строки целочисленной прямоугольной матрицы по возрастанию количества одинаковых элементов в каждой строке.

Найти номер первого из столбцов, не содержащих ни одного отрицательного элемента.

Вариант 17

Путём перестановки элементов квадратной вещественной матрицы добиться того, чтобы её максимальный элемент находился в левом верхнем углу, следующий по величине – в позиции (2,2), следующий по величине – в позиции (3,3) и т.д., заполнить, таким образом, всю главную диагональ.

Найти номер первой их строк, не содержащих ни одного положительного элемента.

Вариант 18

Дана целочисленная прямоугольная матрица. Определить:

1. количество строк, содержащих хотя бы один нулевой элемент;
2. номер столбца, в котором находится самая длинная серия одинаковых элементов.

Вариант 19

Дана целочисленная квадратная матрица. Определить:

1. сумму элементов в тех строках, которые не содержат отрицательных элементов;
2. минимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

Вариант 20

Дана целочисленная прямоугольная матрица. Определить:

1. количество отрицательных элементов в тех строках, которые содержат хотя бы один нулевой элемент;
2. номера строки и столбцов всех седловых точек матрицы.

Примечание. Матрица A имеет седловую точку A_{ij}, если A_{ij} является минимальным элементов в -й строке и максимальным в j-м столбце.

Лабораторная работа 7. Обработка текстовых данных

Цель работы

Целью работы является изучение представления строковых данных в языке С и алгоритмов их обработки.

Задание на лабораторную работу

Разработать функцию, которая выполняет ту обработку символьной строки, которая определена в Вашем индивидуальном задании.

Примечание: строки представляются как массив символов. При реализации функции запрещается пользоваться функциями обработки строк библиотек языка С (например, функцией srtlen стандартной библиотеки С, функциями библиотек "string.h" и "ctype.h").

Примечание: если использовались операторы динамического выделения памяти, то следует вставить дополнительный код, обнаруживающий утечки памяти. См. раздел «Как обнаружить утечки памяти?». Все утечки памяти должны быть устранены.

Варианты заданий

Вариант 1

Функция подсчитывает количество слов в строке.

Вариант 2

Функция дописывает одну строку в конец другой.

Вариант 3

Функция выделяет первое слово из строки.

Вариант 4

Функция сравнивает две строки, игнорируя различия в регистрах.

Вариант 5

Функция разбивает строку на две части: до первого вхождения заданного символа и после него.

Вариант 6

Функция выравнивает строку по левому краю до заданной длины.

Вариант 7

Функция копирует строку в другую строку заданной длины и размещает текст первой строки по центру второй строки.

Вариант 8

Функция удаляет из строки заданное количество символов, начиная с заданной позиции.

Вариант 9

Функция определяет в строке номер позиции слова с заданным номером.

Вариант 10

Функция заменяет символы строки из одного заданного алфавита на символы другого алфавита.

Вариант 11

Функция находит последнее вхождение в строку заданной подстроки.

Вариант 12

Функция доводит длину строки до заданной, вставляя пробелы между словами.

Вариант 13

Функция находит в строке первый символ, который входит в другую заданную строку.

Вариант 14

Функция удаляет из строки слово с заданным номером.

Вариант 15

Функция перезаписывает символы строки заданным количеством символов другой строки, начиная с заданной позиции.

Вариант 16

Функция находит в строке первый символ, который не входить в другую заданную строку.

Вариант 17

Функция находит в строке первый символ, который не входить в другую заданную строку.

Вариант 18

Функция удаляет из начала и из конца строки заданный символ.

Вариант 19

Функция выделяет из строки заданное количество слов, начиная со слова с заданным номером.

Вариант 20

Функция выравнивает строку по правому краю до заданной длины.

Вариант 21

Функция вставляет в строку другую строку, начиная с заданной позиции.

Вариант 22

Функция переводит строку, содержащую десятичное представление целого числа, в строку, содержащую его шестнадцатеричное представление.

Вариант 23

Функция находит первое вхождение в строку заданной подстроки.

Вариант 24

Функция дописывает одну строку у начало другой.

Вариант 25

Функция заменяет в строке все множественные вхождения заданного символа одним.

Вариант 26

Функция заменяет в строке одну заданную комбинацию символов другой заданной комбинацией.

Вариант 27

Функция переписывает все символы строки в обратном порядке.

Вариант 28

Функция определяет длину слова с заданным номером.

Вариант 29

Функция копирует строку в другую строку заданное количество раз.

Вариант 30

Функция выделяет из заданной строки подстроку заданной длины, начиная с заданной позиции.

Лабораторная работа 8.Обработка данных в файлах

Цель работы

Целью работы является изучение файловых операций ввода-вывода.

Задание на лабораторную работу

Работа выполняется в двух вариантах:

1. С использованием библиотек потокового ввода-вывода (<fstream>).
2. С использованием библиотек стандартного ввода-вывода (<cstdio>)

Задания на лабораторную работу приводятся в каждом варианте.

Примечание: для того чтобы открыть файлы с русскими буквами, необходимо включить поддержку русского языка. Смотри «Приложение 9. Печать русских букв в среде Visual C++ 2008».

Примечание: если использовались операторы динамического выделения памяти, то следует вставить дополнительный код, обнаруживающий утечки памяти. См. раздел «Как обнаружить утечки памяти?». Все утечки памяти должны быть устранены.

Варианты заданий

Вариант 1

Написать программу, которая считывает из текстового файла три предложения и записывает их в другой файл в обратном порядке.

Вариант 2

Написать программу, которая считывает текст из файла и записывает в другой файл только те предложения, которые содержат введённое с клавиатуры слово.

Вариант 3

Написать программу, которая считывает текст из файла и записывает в другой файл только те строки, которые содержат двузначные числа.

Вариант 4

Написать программу, которая считывает английский текст из файла и записывает в другой файл те слова, которые начинаются с гласных букв.

Вариант 5

Написать программу, которая считывает текст из файла и записывает его в другой файл, меняя местами каждые два соседних слова.

Вариант 6

Написать программу, которая считывает текст из файла и записывает в другой файл только те предложения, которые не содержат запятых.

Вариант 7

Написать программу, которая считывает текст из файла и определяет, сколько в нём слов, состоящих из не более чем четырёх букв.

Вариант 8

Написать программу, которая считывает текст из файла и записывает в другой файл только цитаты (предложения, заключённые в кавычки).

Вариант 9

Написать программу, которая считывает текст из файла и записывает в другой файл те предложения, которые состоят из заданного количества слов.

Вариант 10

Написать программу, которая считывает английский текст из файла и записывает в другой файл слова текста, начинающиеся и оканчивающиеся на гласные буквы.

Вариант 11

Написать программу, которая считывает текст из файла и записывает в другой файл строки, не содержащие двузначные числа.

Вариант 12

Написать программу, которая считывает текст из файла и записывает в другой файл предложения, начинающиеся с тире, перед которым могут находиться только пробельные символы.

Вариант 13

Написать программу, которая считывает английский текст из файла и записывает его в другой файл, заменив каждую первую букву слов, начинающихся с гласной буквы, на прописную.

Вариант 14

Написать программу, которая считывает текст из файла и записывает его в другой файл, заменив цифры от 0 до 9 на слова «ноль, «один», …, «девять», начиная каждое предложение с новой строки.

Вариант 15

Написать программу, которая считывает текст из файла, находит самое длинное слово и определяет, сколько раз оно встретилось в тексте.

Вариант 16

Написать программу, которая считывает текст из файла и записывает в другой файл сначала вопросительные, а затем восклицательные предложения.

Вариант 17

Написать программу, которая считывает текст из файла и записывает его в другой файл, добавляя после каждого предложения, сколько раз встретилось в нём введённое с клавиатуры слово.

Вариант 18

Написать программу, которая считывает текст из файла и записывает в другой файл все его предложения в обратном порядке.

Вариант 19

Написать программу, которая считывает текст из файла и записывает в другой файл сначала предложения, начинающиеся с однобуквенных слов, а затем все остальные.

Вариант 20

Написать программу, которая считывает текст из файла и записывает в другой файл те предложения, которые содержат максимальное количество знаков пунктуации.

Приложение 1. Рекомендованный список литературы

1. Стандарт языка C++:ISO/IEC 14882-2011(E). Information Technology - Programming languages - C++. Последняя публичная версия черновика: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2011/n3242.pdf
2. ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения».
3. Павловская Т. А. С/С++. Программирование на языке высокого уровня - СПб. : ПИТЕР, 2005. - 461 с.
4. Культин Николай. С/С++ в задачах и примерах, 2-е издание. – СПб.: БХВ-Петербург, 2003. – 288 с.: ил.
5. Дейтел Х., Дейтел П.Как программировать на С++: Пятое издание/Пер. с англ. – М.: Бином, 2008. – 1456 с.: ил.
6. Шилдт Герберт. Полный справочник по С++, 4-е издание. – М.: Вильямс, 2009. – 800 с.: ил.
7. Керниган Брайан В., Пайк Роб. Практика программирования/Пер. с англ. – СПб.: Невский диалект, 2001. – 381 с.: ил.
8. Немцова, Т. И. Программирование на языке высокого уровня. Программирование на языке С++: учеб. пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев ; под ред. Л.Г. Гагариной. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2018. — 512 с. — (Среднее профессиональное образование). - ISBN 978-5-16-102802-5. - Текст : электронный. - URL: https://new.znanium.com/catalog/product/918098 (дата обращения: 11.02.2020)
9. Кузин, А. В. Программирование на языке Си : учебное пособие / А.В. Кузин, Е.В. Чумакова.— М. : ФОРУМ : ИНФРА-М, 2019. — 144 с. — (Высшее образование). - ISBN 978-5-16-102926-8. - Текст : электронный. - URL: https://new.znanium.com/catalog/product/1007488 (дата обращения: 11.02.2020)
10. Парфенов, Д. В. Язык Си: кратко и ясно: Учебное пособие / Д.В. Парфенов. - Москва : Альфа-М: НИЦ ИНФРА-М, 2020. - 320 с. - ISBN 978-5-16-101327-4. - Текст : электронный. - URL: https://new.znanium.com/catalog/product/1046077 (дата обращения: 11.02.2020)

Приложение 2. Титульный лист

ГУАП
КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

к.т.н, доц..		А.В. Туманова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ

ВЫЧИСЛЕНИЕ МАТЕМАТИЧЕСКИХ ВЫРАЖЕНИЙ

по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ(А)

СТУДЕНТ(КА) ГР.	09.02.2022	
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

Приложение 3. Пример оформления отчёта

ГУАП

КАФЕДРА № 43

ОТЧЁТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

доц., канд. техн. наук		А.В. Туманова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №5

ОБРАБОТКА ЧИСЛОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛА

СТУДЕНТКА ГР.	4636	09.02.2022	Т.С. Наумовец
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

1. Цель работы

Целью работы является изучение структуры данных одномерный массив.

2. Задание

Согласно варианту №1 в одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. сумму отрицательных элементов массива;
2. произведение элементов массива, расположенных между максимальным и минимальным элементами;
3. упорядочить элементы массива по возрастанию.

3. Описание созданных функций

Для реализации задания нам потребуются следующие функции:

Имя: sum_neg
Назначение: вычислить сумму отрицательных элементов массива
Входные данные:

- m – массив вещественных элементов;
- n – количество элементов в массиве.

Выходные данные:

- sum – сумма отрицательных элементов массива.

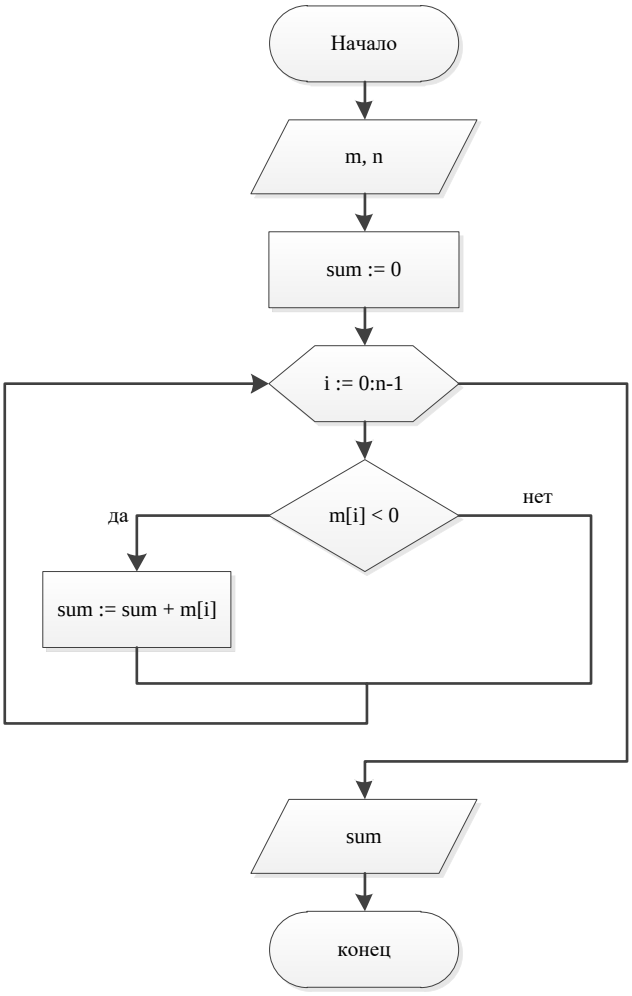
Побочный эффект: отсутствует.
Тестовые данные:

M	N	sum
4 2 1	3	0
5 -6 -7 11	4	-13

Прототип: double sum_neg(const double m[], const size_t n);

Псевдокод
Сумма = 0
Для каждого элемента в массиве
    Если он отрицательный
        То прибавить его значение к сумме

Блок-схема



Имя: prod_minmax

Назначение: произведение элементов массива, расположенных между максимальным и минимальным элементами.

Входные данные:

- m – массив вещественных элементов;
- n – количество элементов в массиве.

Выходные данные:

- prod – произведение элементов массива, расположенных между максимальным и минимальным элементами.

Побочный эффект: отсутствует.

Тестовые данные:

M	N	prod
2 2 2 2	4	0
5 0 3 2 10 6	6	6
1 2 3 5 0 4	6	0

Прототип: double prod_minmax(const double m[], const size_t n);

Псевдокод

Вставить псевдокод.

Блок-схема

Вставить блок-схему.

Имя: sort
Назначение: упорядочить элементы массива по возрастанию.
Входные данные:

- m – массив вещественных элементов;
- n – количество элементов в массиве.

Выходные данные:

- m – упорядоченный массив.

Побочный эффект: отсутствует.
Тестовые данные:

M	N	m
1 2 3 4	4	1 2 3 4
4 3 2 1	4	1 2 3 4
1 2 3 5 0 4	6	0 1 2 3 4 5

Прототип: void sort(double m[], const size_t n);

Псевдокод

Вставить псевдокод.

Блок-схема

Вставить блок-схему.

4. Текст программы

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

double sum_neg(const double m[], const size_t n);
double prod_minmax(const double m[], const size_t n);
void sort(double m[], const size_t n);

int main()
{
	setlocale(LC_ALL, "russian");

	size_tn;
	cout << "Введите размер массива: ";
	cin>>n;

	double *m = new double[n];
```

```
	for (size_t i=0; i<n; i++)
	{
		cout << "Введите m[" << i << "]: ";
		cin >> m[i];
	}

	cout << "Сумма отрицательных элементов равна " << sum_neg(m, n) << endl;
	cout << "Произведение чисел между минимумом и максимумом равно " <<
prod_minmax(m, n) << endl;
	cout << "Отсортированный массив: " << endl;
	sort(m, n);
	for (size_t i=0; i<n; i++)
		cout << "m[" << i << "] = " << m[i] << endl;

	delete [] m;
	return 0;
}

double sum_neg(const double m[], const size_t n)
{
	double sum = 0;
	for (size_t i=0; i<n; i++)
		if (m[i] < 0)
			sum += m[i];
	return sum;
}

double prod_minmax(const double m[], const size_t n)
{
	size_t pos_min = 0;
	for (size_t i=1; i<n; i++)
		if (m[i] < m[pos_min])
			pos_min = i;

	size_t pos_max = 0;
	for (size_t i=1; i<n; i++)
		if (m[i] > m[pos_max])
			pos_max = i;

	if (pos_min > pos_max)
	{
		size_t t = pos_min;
		pos_min = pos_max;
		pos_max = t;
	}

	if (pos_min == pos_max - 1)
		return 0;

	double prod = 1;
	for (size_t i=pos_min+1; i<=pos_max-1; i++)
		prod *= m[i];
	return prod;
}
```

```
void sort(double m[], const size_t n)
{
	bool is_swap;
	do
	{
		is_swap = false;
		for (size_t i=0; i<n-1; i++)
			if (m[i] > m[i+1])
			{
				double t = m[i];
				m[i] = m[i+1];
				m[i+1] = t;
				is_swap = true;
			}
	}
	while (is_swap);
}
```

5. Пример выполнения программы

Ниже показан пример выполнения программы.

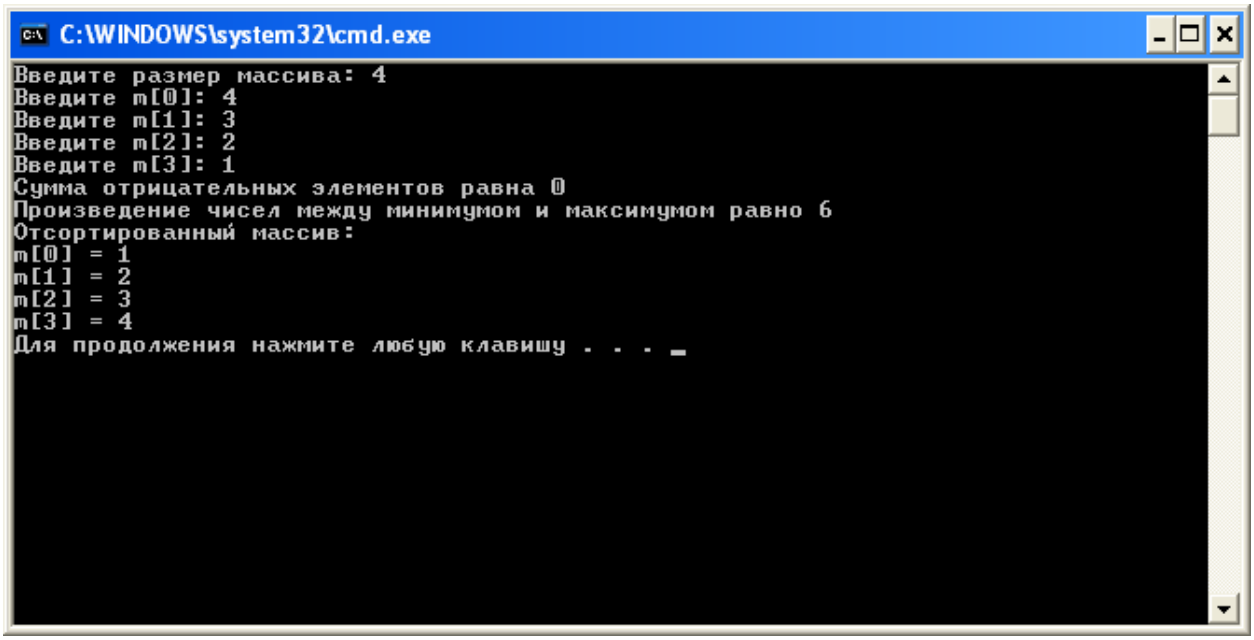


Рис. 1 – Пример выполнения программы

Видно, что результаты расчётов совпадают с тестовыми данными.

6. Анализ результатов и выводы

В результате выполнения лабораторной работы были изучены статические одномерные массивы. Научились передавать массивы, указатели и ссылки в качестве параметров функции, возвращать указатели на массив.

К достоинствам программы можно отнести:

- Каждое задание реализовано в виде отдельной функции, что позволяет использовать эти функции в других проектах.
- За счёт динамического выделения памяти программа использует минимальное количество оперативной памяти.

Из недостатков можно отметить:

- Не производится проверка входных данных.
- Сортировка реализована не самым быстрым алгоритмом, что на больших массивах приводит к более медленной работе.

Приложение 4. Типовые элементы блок-схем

Блок-схема – описание алгоритма, в котором отдельные шаги изображаются в виде блоков различной формы, соединенных между собой линиями, указывающими направление выполнения последовательности. Правила выполнения схем определяются: ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения»[2].

В этом стандарте указаны набор доступных графических символов (раздел 2.2.2) и правила их применения (раздел 4), которые носят довольно общий характер. С одной стороны, это позволяет описывать потоки управления самой разнообразной формы вне зависимости от исполняющего устройства. С другой стороны, подобная свобода приводит к непониманию как их использовать начинающими программистами и отсутствию связи с процедурным программированием.

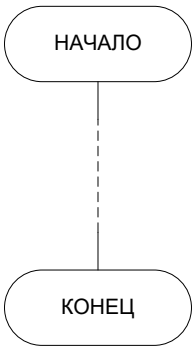
В процедурном программировании, предусматривается ограниченный набор операторов, с помощью которых можно представить любой алгоритм. В большинстве языков программирования это:

- оператор присваивания;
- условный оператор;
- оператор выбора;
- цикл с предусловием;
- цикл с постусловием;
- цикл с параметром;
- вызов подпрограммы.

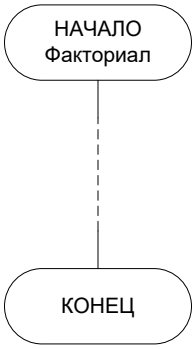
Рассмотрим, как они могут быть описаны с помощью блок-схем.

Блоки начала и завершения программы

Обозначаются овалами или прямоугольниками с сильно закруглёнными краями. Присутствуют в каждой программе и подпрограмме.

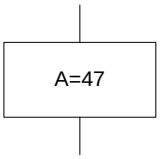


Для удобства после слова «НАЧАЛО» можно указать имя подпрограммы:

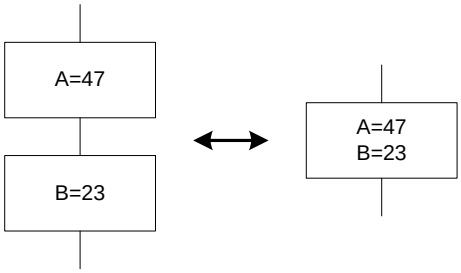


Оператор присваивания

Оператор присваивания обозначается в виде прямоугольника, внутри которого записан сам оператор.



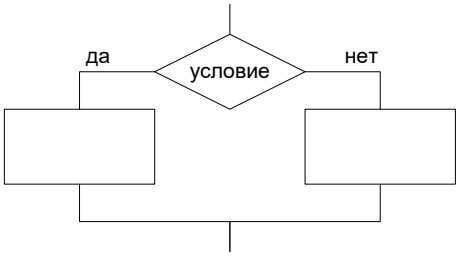
Для более короткой записи без потери смысла можно объединять несколько подряд идущих блоков вычислений в один. Представления слева и справа обозначают одно и тоже.



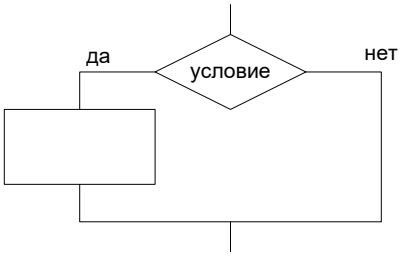
Иногда оператор присваивания обозначается через «:=», в других случаях используя «=». Стандарт ничего не говорит по этому поводу. Обычно, разработчики используют ту запись, которая принята в языке программирования, на котором будет реализован алгоритм.

Условный оператор

Общий вид условного оператора показан ниже. Следует заметить, что разрешается писать только слова «да» и «нет» после ветвления.

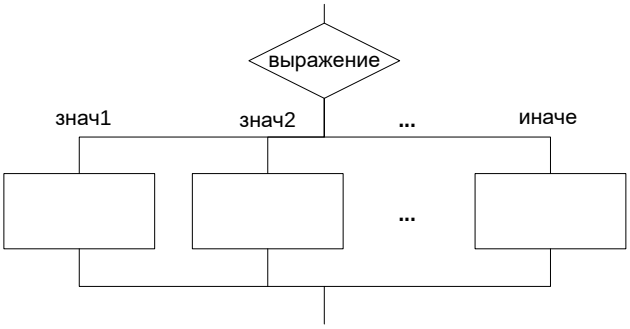


В случае краткой формы условного оператора он может быть записан следующим образом:



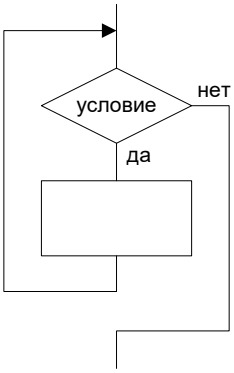
Оператор выбора

Оператор выбора записывается следующим образом:



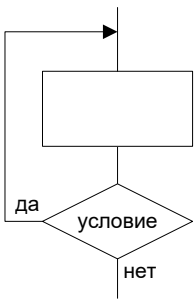
Цикл с предусловием

Цикл с предусловием записываются следующим образом:



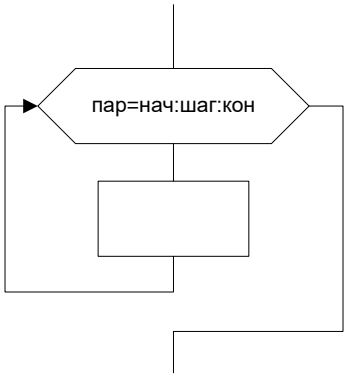
Цикл с постусловием

Цикл с постусловием записываются следующим образом:



Цикл с параметром

Цикл с параметром записываются следующим образом:

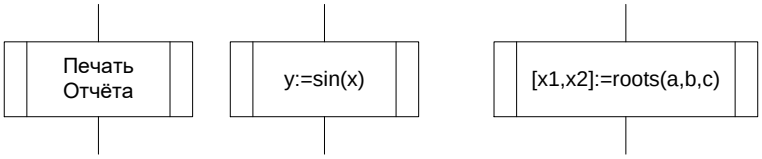


Внутри блока указывают изменяющийся параметр (имя переменной), начальное и конечное значения, а также шаг изменения параметра. Если шаг не указан, то считается равным 1.

Вызов подпрограммы

Вызов подпрограммы обозначается следующим образом:

В стандарте не говорится о способах записи входных и выходных параметров. Но предлагается следующий вариант. Входные параметры записываются в круглых скобках после имени подпрограммы. Выходные – в квадратных скобках перед именем подпрограммы. Примеры вызовов, где функция возвращает 0, 1 и 2 параметров:



Приложение 5. Расширенная таблица ASCII

Для кодировки 866 расширенная таблица ASCII выглядит следующим образом:

[	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺ 1	☻ 2	♥ 3	♦ 4	♣ 5	♠ 6	• 7	8	○ 9	10	11	12	13	14	15
1	► 16	◄ 17	18	! 19	20	21	‗ 22	23	↑ 24	↓ 25	→ 26	← 27	28	↔ 29	▲ 30	▼ 31
2	32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
3	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
4	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
5	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
6	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
7	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	}] 125	~ 126	⌂ 127
8	Ç 128	ü 129	é 130	â 131	ä 132	à 133	å 134	ç 135	ê 136	ë 137	è 138	ï 139	î 140	ì 141	Ä 142	Å 143
9	É 144	æ 145	Æ 146	ô 147	ö 148	ò 149	û 150	ù 151	ÿ 152	Ö 153	Ü 154	¢ 155	£ 156	¥ 157	₧ 158	ƒ 159
A	160	í 161	ó 162	ú 163	ñ 164	Ñ 165	ª 166	º 167	¿ 168	⌐ 169	¬ 170	½ 171	¼ 172	¡ 173	« 174	» 175
B	░ 176	▒ 177	▓ 178	| 179	┤ 180	╡ 181	╢ 182	╖ 183	╕ 184	╣ 185	║ 186	╗ 187	╝ 188	╜ 189	╛ 190	┐ 191
C	└ 192	┴ 193	┬ 194	├ 195	─ 196	┼ 197	╞ 198	╟ 199	╚ 200	╔ 201	╩ 202	╦ 203	╠ 204	═ 205	╬ 206	╧ 207
D	╨ 208	╤ 209	╥ 210	╙ 211	╘ 212	╒ 213	╓ 214	╫ 215	216	┘ 217	┌ 218	▉ 219	▄ 220	▌ 221	▐ 222	▀ 223
E	α 224	ß 225	Γ 226	π 227	Σ 228	σ 229	µ 230	τ 231	Φ 232	Θ 233	Ω 234	δ 235	∞ 236	φ 237	ε 238	∩ 239
F	≡ 240	± 241	≥ 242	≤ 243	⌠ 244	⌡ 245	÷ 246	≈ 247	° 248	∙ 249	· 250	√ 251	ⁿ 252	² 253	■ 254	□ 255

Кодировка символов, предложенная IBM (соответствует ASCII - кодировке)

Альтернативная кодовая таблица

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺ 1	☻ 2	♥ 3	♦ 4	♣ 5	♠ 6	• 7	8	○ 9	10	11	12	13	14	15
1	► 16	◄ 17	18	19	20	21	‗ 22	23	↑ 24	↓ 25	→ 26	← 27	28	↔ 29	▲ 30	▼ 31
2	32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
3	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
4	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
5	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
6	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
7	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	}] 125	~ 126	⌂ 127
8	А 128	Б 129	В 130	Г 131	Д 132	Е 133	Ж 134	З 135	И 136	Й 137	К 138	Л 139	М 140	Н 141	О 142	П 143
9	Р 144	С 145	Т 146	У 147	Ф 148	Х 149	Ц 150	Ч 151	Ш 152	Щ 153	Ъ 154	Ы 155	Ь 156	Э 157	Ю 158	Я 159
A	а 160	б 161	в 162	г 163	д 164	е 165	ж 166	з 167	и 168	й 169	к 170	л 171	м 172	н 173	о 174	п 175
B	░ 176	▒ 177	▓ 178	│ 179	┤ 180	╡ 181	╢ 182	╖ 183	╕ 184	╣ 185	║ 186	╗ 187	╝ 188	╜ 189	╛ 190	┐ 191
C	└ 192	┴ 193	┬ 194	├ 195	─ 196	┼ 197	╞ 198	╟ 199	╚ 200	╔ 201	╩ 202	╦ 203	╠ 204	═ 205	╬ 206	╧ 207
D	╨ 208	╤ 209	╥ 210	╙ 211	╘ 212	╒ 213	╓ 214	╫ 215	216	┘ 217	┌ 218	▉ 219	▄ 220	▌ 221	▐ 222	▀ 223
E	р 224	с 225	т 226	у 227	ф 228	х 229	ц 230	ч 231	ш 232	щ 233	ъ 234	ы 235	ь 236	э 237	ю 238	я 239
F	Ё 240	ё 241	≥ 242	≤ 243	⌠ 244	⌡ 245	÷ 246	≈ 247	° 248	∙ 249	· 250	√ 251	ⁿ 252	² 253	■ 254	⌂ 255

Для кодировки 1251 расширенная таблица ASCII выглядит следующим образом:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				©	Ё	§	Є	·		°						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2		!	"	#	$	%	&		(	)	*	+				/
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}]	~	□
	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	Ђ	Ѓ	‚	ѓ	„	…	†	‡		‰	Љ	‹	Њ	Ќ	Ћ	Џ
	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	ђ	‘	’	"	"		–	—		™	љ	›	њ	ќ	ћ	џ
	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A		Ў	ў	J	¤	Ґ	¦	§	Ё	©	Є	«		-	®	Ї
	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	°	±	І	і	ґ	µ		·	ё	№	є	»	j	S	s	ї
	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	▪ 223
E	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Приложение 6. Таблица приоритетов операций

Таблица приоритетов определяет приоритет в цепочках выражений, когда порядок вычислений не был явно задан скобками. В стандарте языка C++ указано [1 пункт 5.1 стр. 87] «The precedence of operators is not directly specified, but it can be derived from the syntax» или «приоритеты операторов явно не определены, но вытекают из грамматики». Поэтому ниже приводится таблица приоритетов операций, полученных из грамматики.

Также приведём значения некоторых терминов в области программирования:

- **Арность** операции – количество операндов, над которыми данная операция вычисляется. Слово образовалось из названий предикатов небольшой арности (унарный – один аргумент, бинарный – два, тернарный – три).
- **Ассоциативность** (очерёдность) операторов – это последовательность их выполнения (или направление вычисления), реализуемое, когда операторы имеют одинаковый приоритет и отсутствует явное (с помощью скобок) указание на очерёдность их выполнения. При этом различается левая ассоциативность, при которой вычисление выражения происходит слева-направо, и правая ассоциативность – справа-налево. Соответствующие операторы называют левоассоциативными и правоассоциативными.

Приоритет	Оператор	Описание	Арность	Ассоциативность
1 Наивысший	`::`	Изменение области видимости	1	Слева направо
2	`++`	Суффиксный инкремент	1	
	`--`	Суффиксный декремент	1	
	`()`	Вызов функции	2	
	`[]`	Взятие элемента массива	1	
	`.`	Выбор элемента по ссылке	2	
	`->`	Выбор элемента по указателю	2	
	`typeid()`	RTTI (только C++; см typeid)	1	
	`const_cast`	Преобразование типов (C++)	2	
	`dynamic_cast`	Преобразование типов (C++)	2	
	`reinterpret_cast`	Преобразование типов (C++)	2	
	`static_cast`	Преобразование типов (C++)	2	
3	`++`	Префиксный инкремент	1	Справа налево
	`--`	префиксный декремент	1	
	`+`	Унарный плюс	1	
	`-`	Унарный минус	1	
	`!`	Логическое НЕ	1	
	`~`	Побитовое НЕ	1	
	`(`*`type`*`)`	Приведение типов	2	
	`*`	Разыменование указателя	1	
	`&`	Взятие адреса	1	
	`sizeof`	Size-of (размер)		
	`new, new[]`	Выделение динамической памяти (C++)	1	

Приоритет	Оператор	Описание	Арность	Ассоциативность
	`delete, delete[]`	Освобождение динамической памяти (C++)	1	
4	`.*`	Указатель на член (C++)	2	Слева направо
	`->*`	Указатель на член (C++)	2	
5	`*`	Умножение	2	
	`/`	Деление	2	
	`%`	Взятие модуля (остаток от деления)	2	
6	`+`	Сложение	2	
	`-`	Вычитание	2	
7	`<<`	Побитовый сдвиг влево	2	
	`>>`	Побитовый сдвиг вправо	2	
8	`<`	Менее	2	
	`<=`	Менее или равно	2	
	`>`	Более	2	
	`>=`	Более или равно	2	
9	`==`	Равенство	2	
	`!=`	Неравенство	2	
10	`&`	Побитовое И	2	
11	`^`	Побитовое исключающее или (XOR)	2	
12	`|`	Побитовое ИЛИ	2	
13	`&&`	Логическое И	2	
14	`||`	Логическое ИЛИ	2	
15	`?:`	Условный оператор	3	Справа налево
16	`=`	Прямое присваивание	2	Справа налево
	`+=`	Присваивание со сложением	2	
	`-=`	Присваивание с вычитанием	2	
	`*=`	Присваивание с умножением	2	
	`/=`	Присваивание с делением	2	
	`%=`	Присваивание с взятием остатка от деления	2	
	`<<=`	Присваивание с побитовым сдвигом влево	2	
	`>>=`	Присваивание с побитовым сдвигом вправо	2	
	`&=`	Присваивание с побитовым И	2	
	`^=`	Присваивание с побитовым XOR	2	
	`|=`	Присваивание с побитовым OR	2	
17	`throw`	Оператор создания исключения (C++)		
18	`,`	Последовательное вычисление	2	Слева направо

Приложение 7. Перечень функций стандартной библиотеки C++

Подключение библиотек

Библиотеки языка C++ это набор файлов, которые содержат описания функций. Подключая библиотеку можно вызывать те функции, которые определены в этой библиотеке. Подключение библиотек происходит с помощью директивы «#include»

```
#include <iostream> // Подключение библиотеки потокового ввода-вывода
#include <cmath>    // Подключение библиотеки математических функций
```

В описании библиотек приводится только назначение функций. За более полной информацией необходимо обратиться к документации по языку программирования.

Библиотека утверждений«cassert» (assert.h)

Функции проверки утверждений	
assert	Проверить утверждение

Более подробное описание находится, например, здесь: http://ru.wikipedia.org/wiki/Assert.h.

Библиотека посимвольной обработки«cctype» (ctype.h)

Функции проверки категории символа	
isalnum	Проверяет, является ли аргумент буквой или цифрой
isalpha	Проверяет, является ли аргумент буквой
iscntrl	Проверяет, является ли аргумент управляющим символом
isdigit	Проверяет, является ли аргумент цифрой
isgraph	Проверяет, является ли аргумент символом, имеющим графическое представление
islower	Проверяет, является ли аргумент буквой в нижнем регистре
isprint	Проверяет, является ли аргумент символом, который может быть напечатан
ispunct	Проверяет, является ли аргумент символом, имеющим графическое представление, но не являющимся при этом буквой или цифрой
isspace	Проверяет, является ли аргумент разделительным символом
isupper	Проверяет, является ли аргумент буквой в верхнем регистре
isxdigit	Проверяет, является ли аргумент цифрой шестнадцатеричной системы счисления
Функции изменения регистра	
tolower	Изменить прописную букву на строчную («большую» на «маленькую»)
toupper	Изменить строчную букву на прописную («маленькую» на «большую»)

ASCII код	Символы	isalnum	isalpha	iscntrl	isdigit	isgraph	islower	isprint	ispunct	isspace	isupper	isxdigit
0x00 .. 0x08	NUL, (другие управляющие коды)			x								
0x09 .. 0x0D	(управляющие коды, перемещающие курсор: '\t','\f','\v','\n','\r')			x						x		
0x0E .. 0x1F	(другие управляющие коды)			x								
0x20	пробел (' ')							x		x		
0x21 .. 0x2F	!"#$%&'()*+,-./					x		x	x			
0x30 .. 0x39	01234567890	x			x	x		x				x

0x3a .. 0x40	:;<=>?@					x		x	x			
0x41 .. 0x46	ABCDEF	x	x			x		x			x	x
0x47 .. 0x5A	GHIJKLMNOPQRSTUVWXYZ	x	x			x		x			x	
0x5B .. 0x60	[\]^_`					x		x	x			
0x61 .. 0x66	abcdef	x	x			x	x	x				x
0x67 .. 0x7A	ghijklmnopqrstuvwxyz	x	x			x	x	x				
0x7B .. 0x7E	{|}~					x		x	x			
0x7F	(DEL)			x								

Библиотека региональных настроек«clocale» (locale.h)

Данная библиотека используется для задач связанных из локализацией. Для правильной обработки символов национальных алфавитом следует подключить данную библиотеку и установить параметры, используя «setlocale».

Функции локализации	
setlocale	Установить или получить текущие региональные настройки
localeconv	Получить конкретные параметры региональных настроек

Библиотека математических функций «cmath» (math.h)

Все тригонометрические функции вычисляют в радианах, а не в градусах. Чтобы перейти от одной размерности к другой, можно воспользоваться следующими формулами:

$$\text{град} = \frac{180}{\text{p}}\text{рад}$$

$$\text{рад} = \frac{\text{p}}{180}\text{град}$$

Тригонометрические функции	
cos	Вычисляет значение косинуса
sin	Вычисляет значение синуса
tan	Вычисляет значение тангенса
acos	Вычисляет значение арккосинуса
asin	Вычисляет значение арксинуса
atan	Вычисляет значение арктангенса
atan2	Вычисляет значение арктангенса с двумя параметрами
Гиперболические функции	
cosh	Вычисляет значение гиперболического косинуса
sinh	Вычисляет значение гиперболического синуса
tanh	Вычисляет значение гиперболического тангенса
Экспоненциальные и логарифмические функции	
exp	Вычисление экспоненты
frexp	Разбивает число с плавающей точкой на мантиссу и показатель степени
ldexp	Умножение числа с плавающей точкой на целую степень двух
log	Вычисляет значение натурального логарифма
log10	Вычисляет значение десятичного логарифма
modf	Извлекает целую и дробную части (с учетом знака) из числа с плавающей точкой
Возведение в степень	
pow	Возведение в степень
sqrt	Вычисляет значение квадратного корня
Округление, абсолютное значение, остаток от деления	
ceil	Округление вверх
fabs	Вычисляет значение абсолютной величины (числа с плавающей точкой)
floor	Округление вниз
fmod	Вычисление остатка от деления нацело (числа с плавающей точкой)

Константа π

Стандарт языка C++ [1] не определяет эту константу. Поэтому есть два варианта:

1) определить самостоятельно:

```
const double PI =3.141592653589793238462;
const float PI_F=3.14159265358979f;
```

или вычислить например, через арктангенс

```
#include <cmath>
const double PI = std::atan(1.0)*4;
```

2) некоторые компиляторы (обычно) определяют константу M_PI, но нужно сделать так:

```
#include <cmath>
const double PI = std::atan(1.0)*4;
```

а затем использовать константу M_PI. Например:

```
#define _USE_MATH_DEFINES
#include <cmath>

int main()
{
	double x = cos(M_PI/2);
	return 0;
}
```

Библиотека ввода-вывода «cstdio» (stdio.h)

Операции над файлами	
remove	Удалить файл
rename	Переименовать файл
tmpfile	Открыть временный файл
tmpnam	Создать уникальное имя для временного файла
Доступ к файлу	
fclose	Закрыть файл
fflush	Принудительно опустошает буфер вывода, записывая его содержимое в файл
fopen	Открыть файл
freopen	Заново открыть файл в другом режиме или другой файл
setbuf	Установить буфер вывода для заданного файла
setvbuf	Установить режим буферизации для заданного файла
Форматированный ввод-вывод	
fprintf	Форматированный вывод в файл
fscanf	Форматированный ввод из файла
printf	Форматированный вывод на экран
scanf	Форматированный ввод с клавиатуры
sprintf	Форматированный вывод в строку
sscanf	Форматированный ввод из строки
Посимвольный ввод-вывод	
fgetc	Получить следующий символ из файла
fgets	Получить следующую строку из файла
fputc	Записать символ в файл
fputs	Записать строку в файл
getchar	Прочитать символ с клавиатуры

gets	Прочитать строку с клавиатуры
putchar	Вывести символ на экран
puts	Вывести строку на экран
ungetc	Возвратить символ обратно в буфер чтения файла
Прямой (блочный) ввод-вывод	
fread	Прочитать блок данных из файла
fwrite	Записать блок данных в файл
Управление текущей позицией в файле	
fseek	Установить текущую позицию в файле
ftell	Получить текущую позицию в файле
rewind	Установить позицию в файле на начало
Обработка ошибок	
clearerr	Очистить индикатор ошибок
feof	Проверить, достигнут конец файла
ferror	Проверить наличие ошибки
perror	Вывести сообщение об ошибке

Библиотека дополнительных функций«cstdlib» (stdlib.h)

Преобразование строковых данных	
atof	Преобразовать значение строки в вещественное число двойной точности
atoi	Преобразовать значение строки в целое
atol	Преобразовать значение строки в длинное целое
Создание псевдослучайных чисел	
rand	Создать псевдослучайное число
srand	Инициализировать датчик псевдослучайных чисел
Целочисленная арифметика	
abs	Абсолютное значение (для целых чисел)
div	Целочисленное деление с остатком

Библиотека обработки строк «cstring» (string.h)

Копирование	
strcpy	Копировать одну строку в другую
strncpy	Копировать заданное число символов одной строки в другую
strxfrm	Копировать одну строку в другую с использованием региональных настроек
Слияние (конкатенация)	
strcat	Дописать одну строку к другой
strncat	Дописать заданное число символов из одной строки к другой
Сравнение	
strcmp	Сравнить две строки
strcoll	Сравнить две строки, используя региональные настройки
strncmp	Сравнить заданное число первых символов двух строк
Поиск	
strchr	Найти первое вхождение символа в строку
strcspn	Найти первое вхождение символа из заданного множества в строку
strpbrk	Осуществить последовательный поиск всех вхождений символов заданного множества в строке
strrchr	Найти последнее вхождение символа в строку
strspn	Определить сколько символов заданного множества встречается с начала строки
strstr	Найти первое вхождение подстроки в строку
strtok	Разбить строку на части
Другие	
strlen	Вычислить длину строки

Приложение 8. Создание нового проекта в среде Visual C++ 2008

1. Запустите Visual Studio 2008. (Меню «Все программы» - «Microsoft Visual Studio 2008» - « Microsoft Visual Studio 2008»)
2. Вызовите мастера создания нового проекта. (Меню «Файл» - «Создать» - «Проект»)
3. Выберите тип проекта «Visual C++» - «Win32». Выберите шаблон проекта «Консольное приложение Win32». Укажите имя проекта, например, «lab1» или «spiski». Снимите галочку «Создать каталог для решения». Нажмите «ОК».

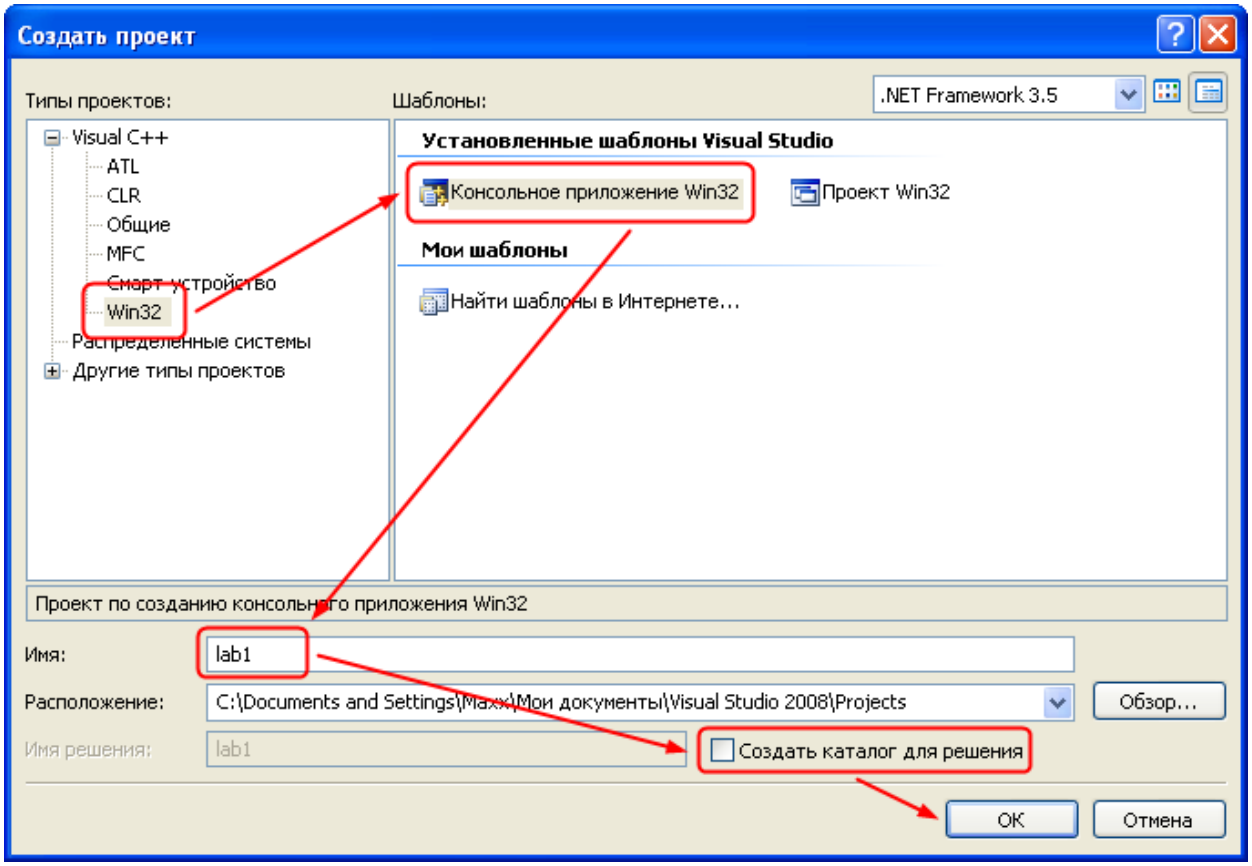


Рис. 2. Выбор типа создаваемого проекта

После нажатия на кнопку «ОК» в папке, указанной в поле «Расположение», будет создана папка с именем проекта. Эту папку с именем проекта можно скопировать на флэшку или любой другой носитель информации. При сдаче лабораторной работы в учебной лаборатории открывается проект из папки, и студент показывает сделанную программу.

Также папку с проектом можно копировать и перемещать в любую другую папку, например, с целью резервного копирования или унося результаты работы из учебной лаборатории домой.

4. В открывшемся окне «Мастер приложений Win32» нажмите кнопку «Далее».

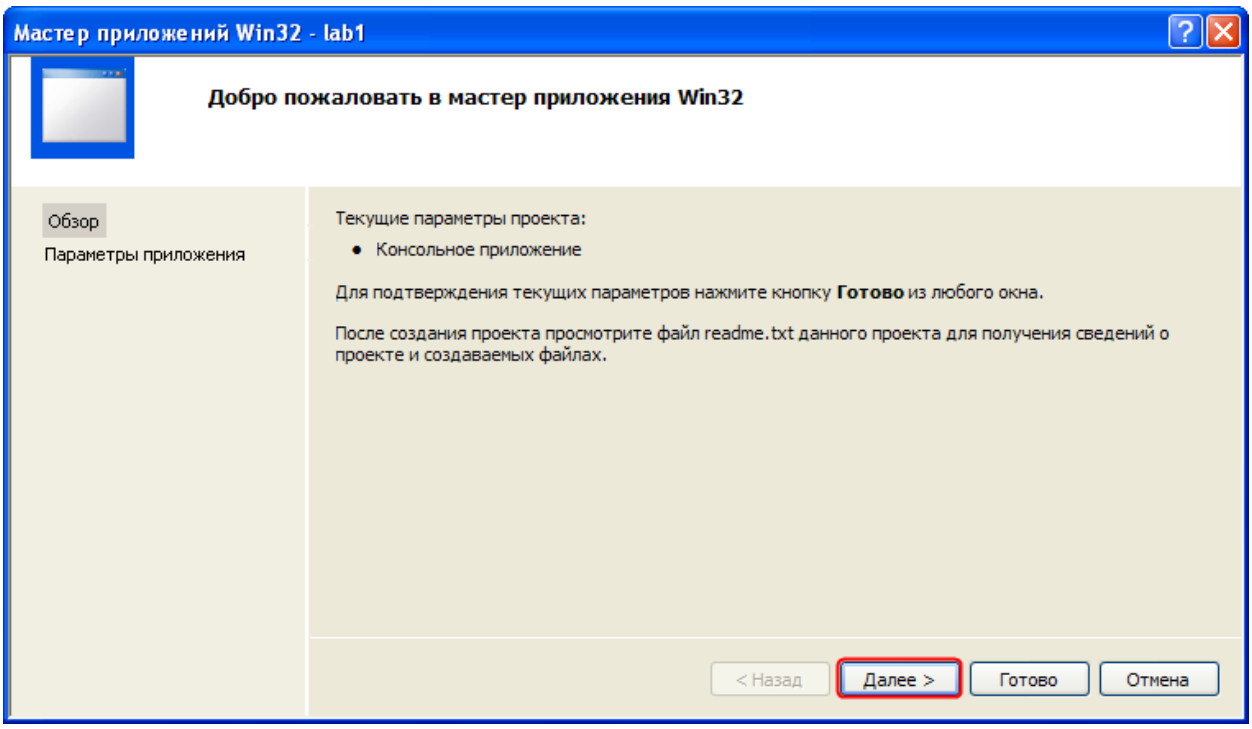


Рис. 3. Мастер приложений. Вкладка «Обзор»

5. Во вкладке «Параметры приложения» необходимо выбрать тип приложения «Консольное приложение». В дополнительных параметрах указать «Пустой проект». А затем нажать кнопку «Готово».

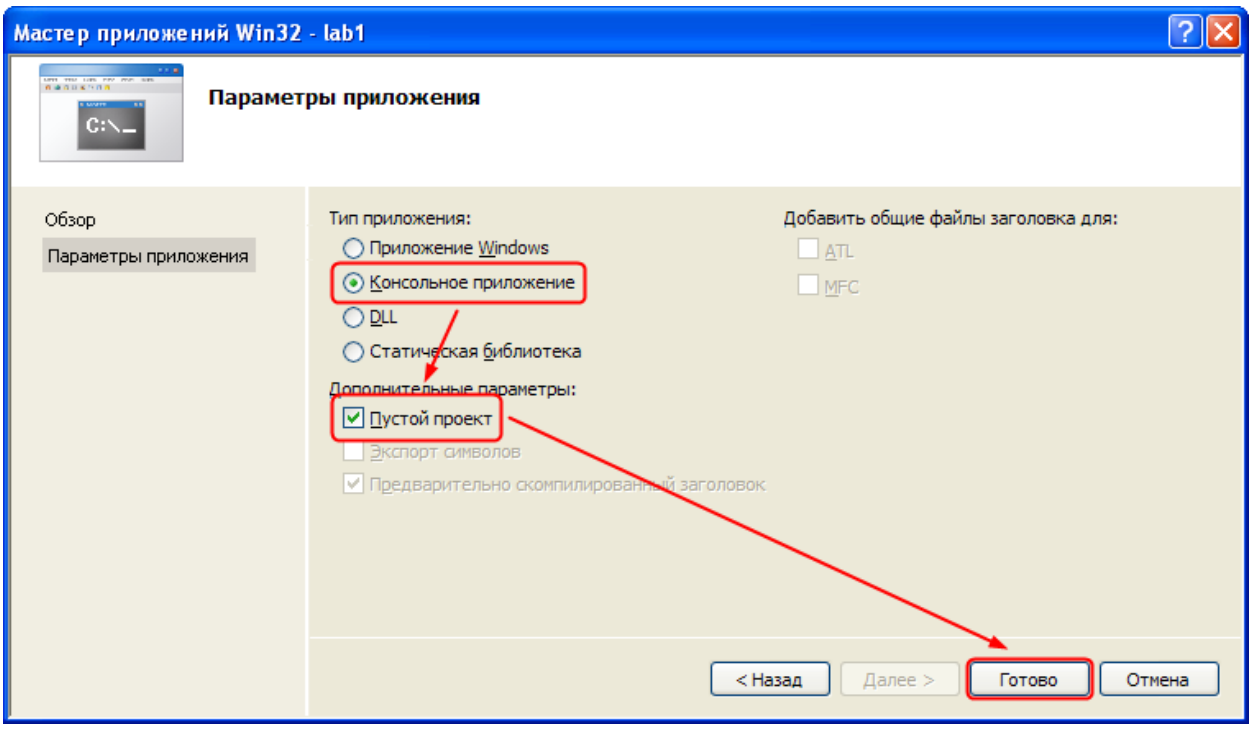


Рис. 4. Мастер приложений. Вкладка «Параметры приложения»

6. В обозревателе решений необходимо добавить новый файл. Для этого правой кнопкой мыши нажмите на «Файлы исходного кода» в окне «Обозреватель решений». В появившемся меню выбрать «Добавить» и «Создать элемент…».

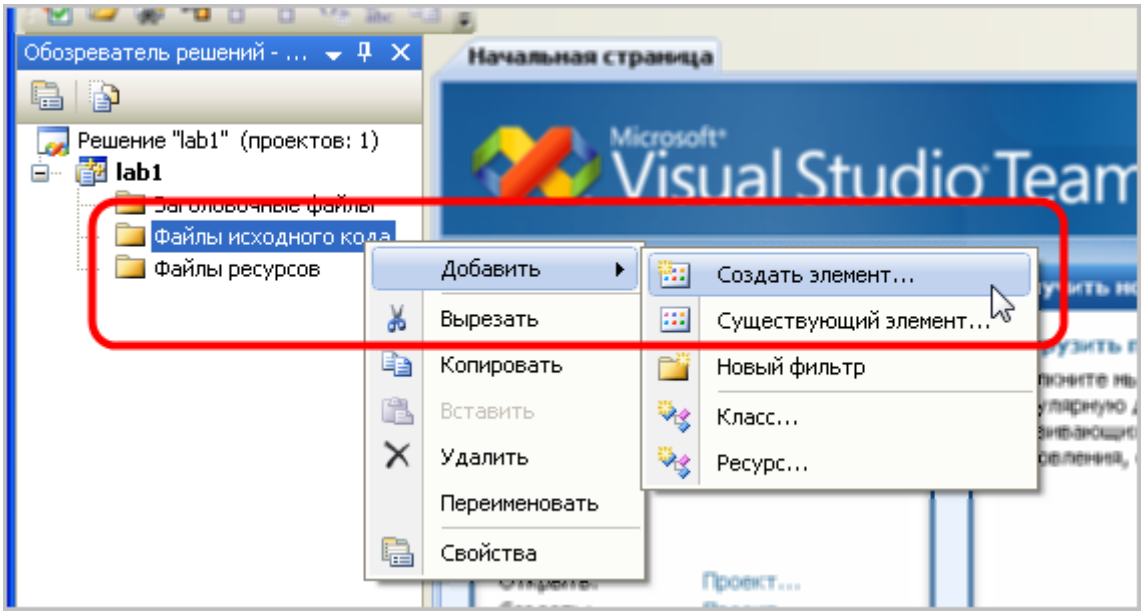


Рис. 5. Вызов мастера «Добавление нового элемента»

7. В мастере «Добавление нового элемента» выбрать категорию «VisualC++» - «Код». В установленных шаблонах выбрать «Файл C++ (.cpp)». Указать имя файла, например, «main» (расширение файла «.cpp» можно опустить). Нажать кнопку «Добавить».

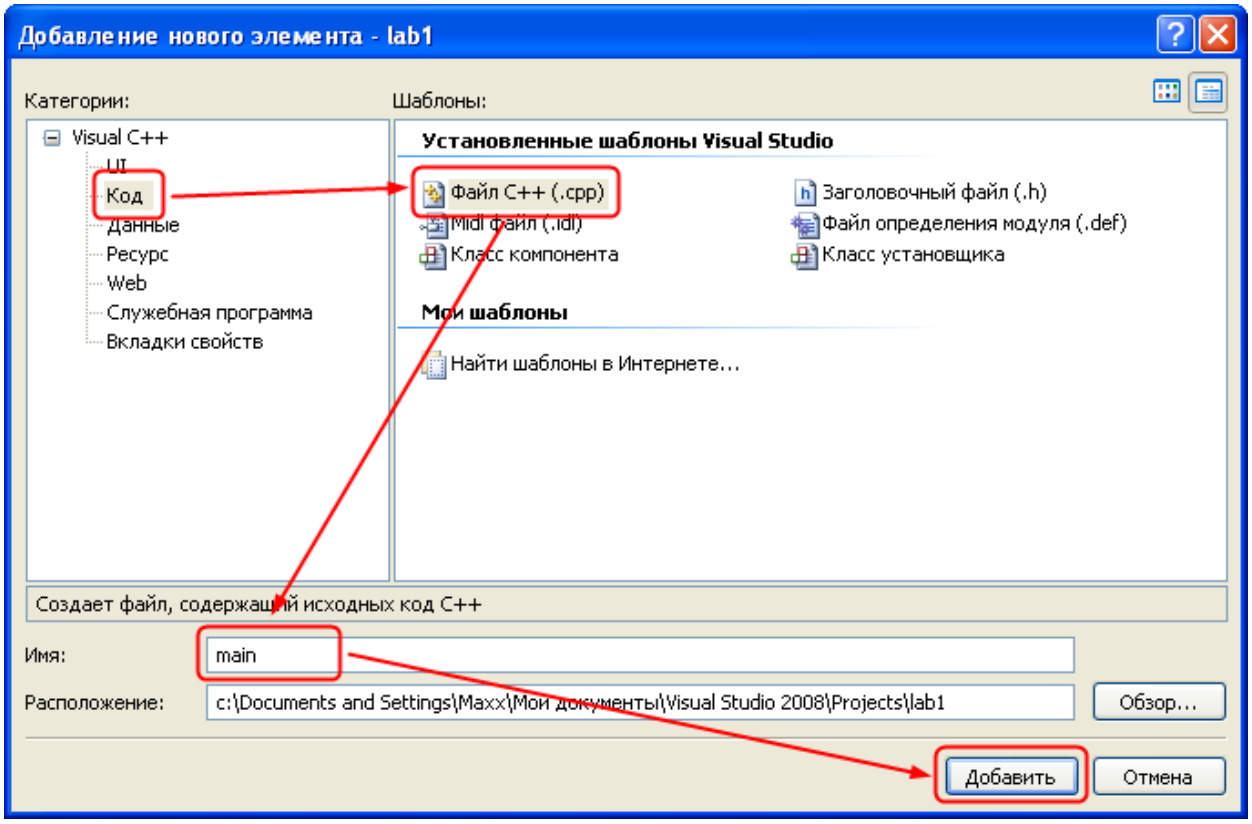


Рис. 6. Добавление файла в проект

8. В папке «Файлы исходного кода программы» окна «Обозреватель решений» появиться новый элемент, а в рабочей области среды откроется новый файл. Далее необходимо набрать исходный код программы.

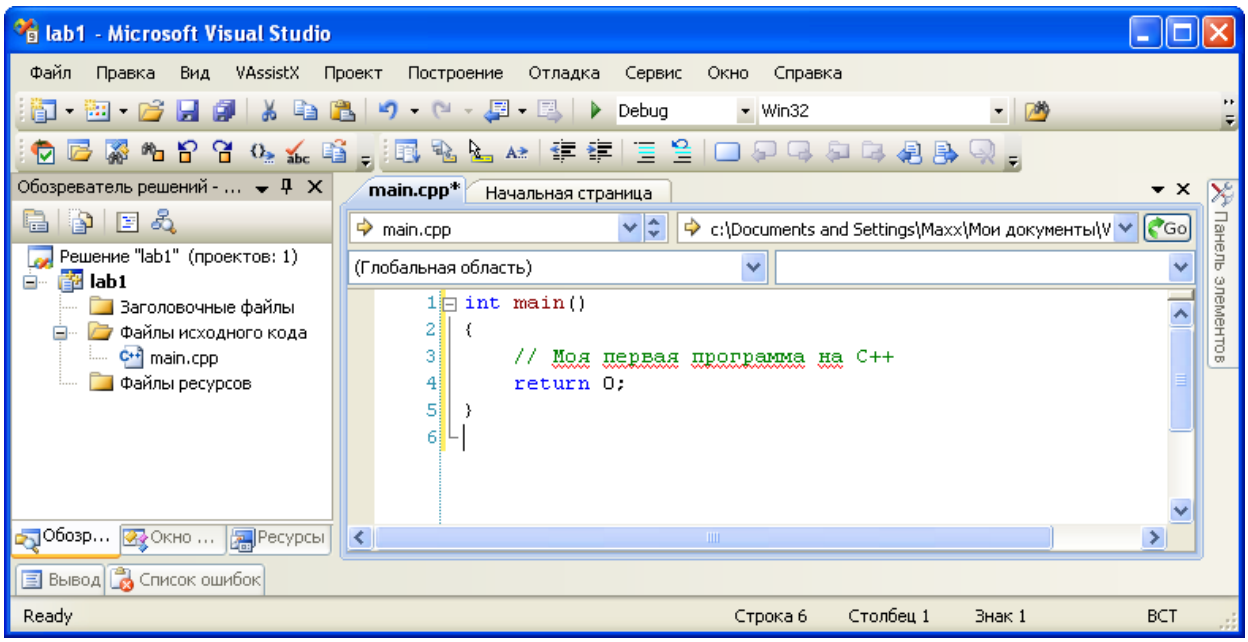


Рис. 7. Главное окно среды после добавления нового файла в проект

Приложение 9. Отладка кода проекта в среде Visual C++

Отладка – этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Чтобы понять, где возникла ошибка, приходится:

- узнавать текущие значения переменных;
- и выяснять, по какому пути выполнялась программа.

Существуют две взаимодополняющие технологии отладки.

Использование отладчиков – программ, которые включают в себя пользовательский интерфейс для пошагового выполнения программы: оператор за оператором, функция за функцией, с остановками на некоторых строках исходного кода или при достижении определённого условия.

Вывод текущего состояния программы с помощью расположенных в критических точках программы операторов вывода – на экран, файл, принтер или громкоговоритель. Вывод отладочных сведений в файл называется журналированием и используется при отладке больших программ.

Рассмотрим основные возможности среды Visual C++ для отладки программ на следующем примере:

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

int sum1n(int n)
{
	int i, s=0;
	for (i=0; i<n; i++)
		s += i;
	return s;
}

int main()
{
	int sum;
	sum = sum1n(10);
	cout<<"sum = "<<sum<<endl;
	return 0;
}
```

Режимы запуска программы

Программа в среде Visual Studio может быть запущена в двух режимах:

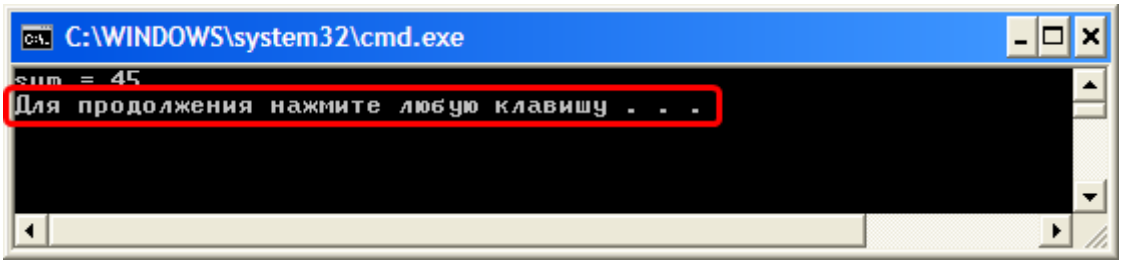
- в режиме отладки (горячая клавиша F5, меню «Отладка» - «Начать отладку»);
- без отладки (горячая клавиша Ctrl+F5, меню «Отладка» - «Запуск без отладки»).

Запуск программы в режиме отладки означает, что:

- выполнение программы остановится при достижении первой точки останова;
- во время выполнения программы можно устанавливать новые точки останова, что позволяет переходить в режим пошаговой отладки в нужный момент в нужном месте;

- переходить в режим пошагового выполнения программы при возбуждении исключительных ситуаций;
- другое.

Чтобы проверить выполнение программы, не удаляя точки останова, программа запускается в режиме без отладки. Другой особенностью этого режима является ожидание нажатия любой клавиши перед завершением программы. Очень часто начинающим программистам непонятно, почему окно с программой автоматически закрывается. Чтобы предотвратить автоматическое закрытие, программа запускается в режиме без отладки.



По окончании работы программы пользователь видит сообщение «Для продолжения нажмите любую клавишу …».

Пошаговое выполнение программы

В пошаговом режиме в программе выполняется одна строчка на шаг. Если в одной строке размещаются два и более операторов, то все эти операторы будут выполнены за один шаг. Для упрощения дальнейшего изложения будем предполагать, что в одной строке программы находится один оператор. Тогда термины строка программы и оператор программы будут эквивалентны.

На рисунке 8 показано, как отмечается текущая строка, операторы которой будут выполнены на следующем шаге.

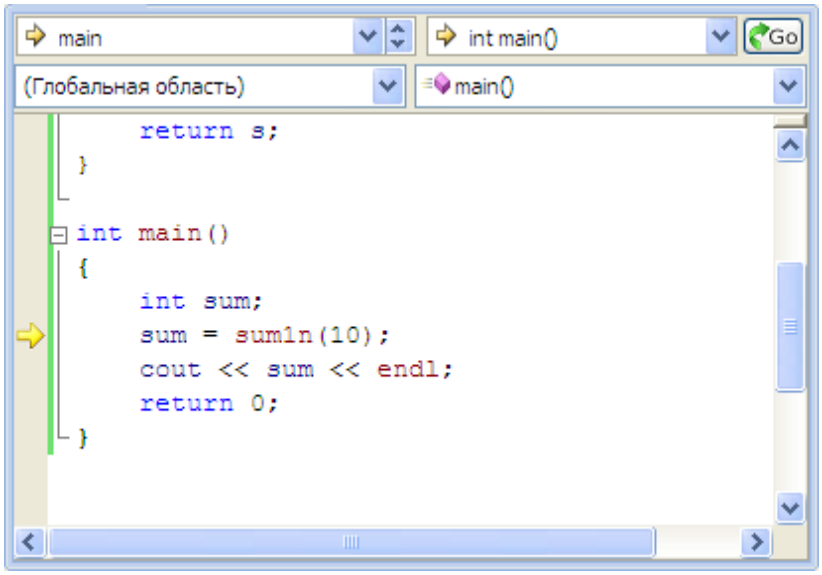


Рис. 8. Пометка текущей строки в режиме отладки с помощью жёлтой стрелки

Существует три типа шагов, которые можно выполнять:

1. Шаг с заходом (горячая клавиша F11, меню «Отладка» - «Шаг с заходом»).
2. Шаг с выходом (горячая клавиша Shift+F11, меню «Отладка» - «Шаг с выходом»).

3. Шаг с обходом(горячая клавиша F10, меню «Отладка» - «Шаг с обходом»).

Шаг с заходом обозначает, что если в текущей строке имеется вызов функции, то:

- происходит вычисление параметров и вызов функции;
- управление передаётся в функцию (рисунок 9).

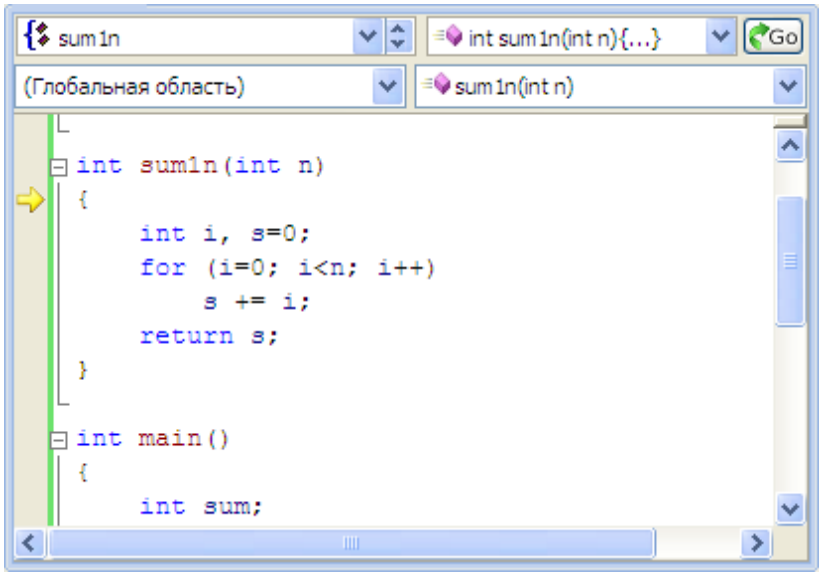


Рис. 9. Передача управления в функцию с остановкой перед первым оператором

Шаг с выходом обозначает, что все операторы текущей функции выполняются, а программа приостанавливает своё выполнение на операторе, где происходил вызов текущей функции. Например, выполнение шага с выходом внутри функции «sum1n» приведёт к выполнению всех операторов (согласно логике программы) этой функции и приостановке на операторе вызова этой функции внутри функции «main» (рисунок 10). Приостановка именно на этом операторе позволяет отладить другие функции, вызов которых находится в этой же строке.

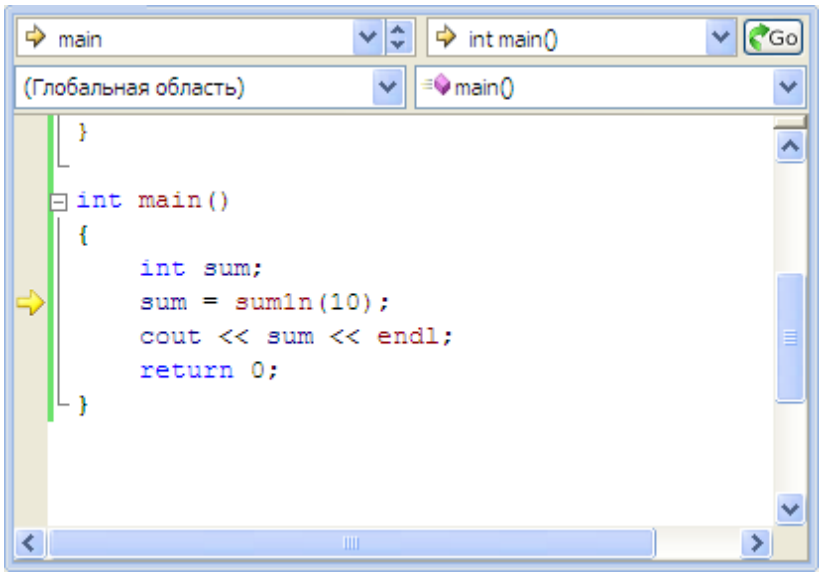


Рис. 10. Передача управления из функции на оператор где она вызывалась

Шаг с обходом обозначает, что выполняется текущий оператор (с вызовом всех входящих в него функций), а управление приостанавливается на следующем.

Точки останова

Иногда программу пошагово выполнять бывает очень утомительно и отладку нужно провести, начиная с определённого оператора. Таким оператором может быть оператор цикла «for» внутри функции «sum1n». Для этого на операторе цикла ставится точка останова:

- курсор перемещается на строку, где программа должна приостановиться, дойдя до этого места;
- выставляем точку останова, нажимая F9 или используя меню «Отладка» - «Точка останова» (рисунок 11).

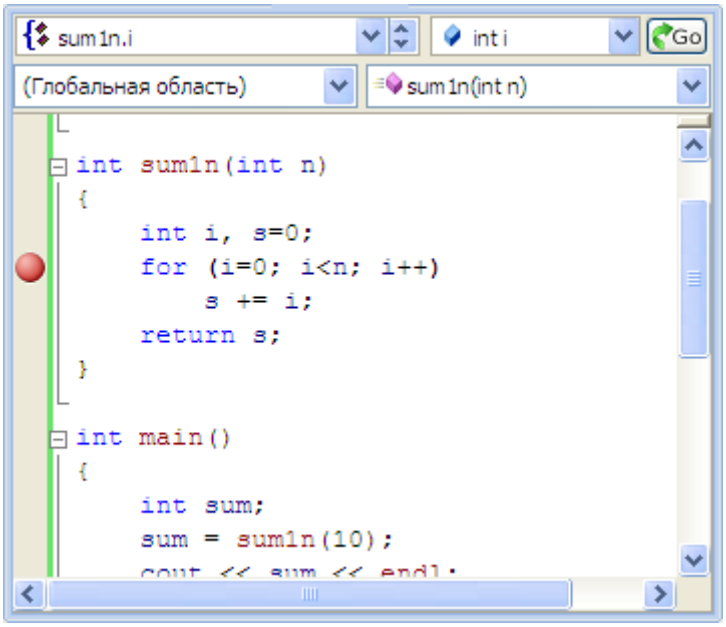
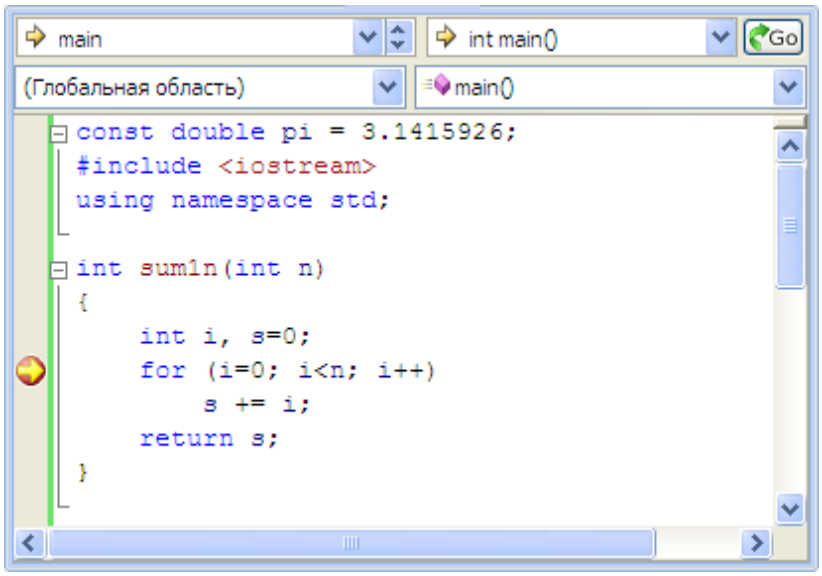


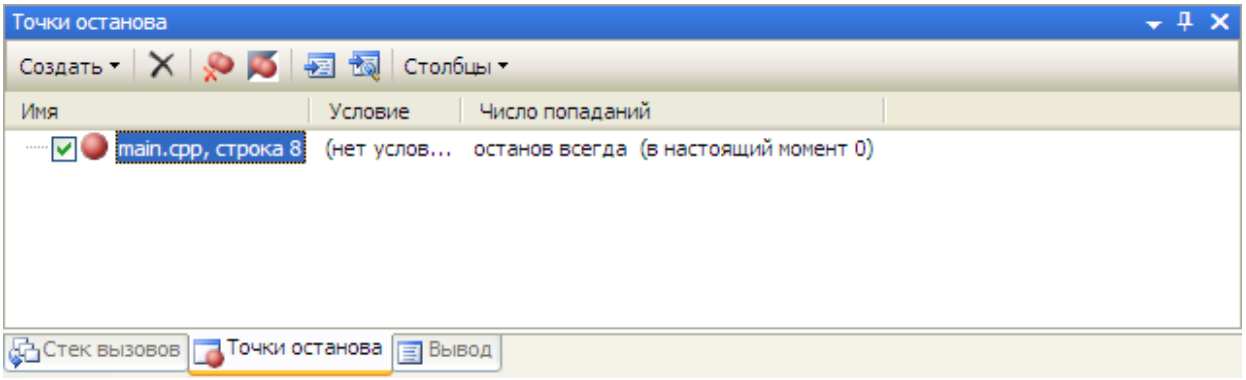
Рис. 11. Установка точки останова на заданном операторе

Затем программа запускается в режиме отладки. Как только происходит достижение оператора с точкой останова, то выполнение программы прерывается и отладка переходит в пошаговый режим.



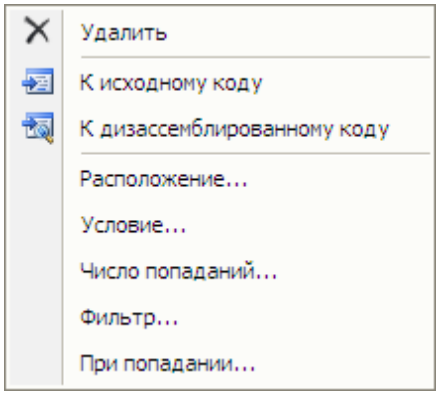
Снятие точки останова происходит точно также: если в этой строке была установлена точка останова, то нажатие F9 приведёт к её удалению.

Перечень всех точек останова можно получить через меню «Отладка» - «Окна» - «Точки останова» или с помощью горячей клавиши Alt+F9.



С помощью этого окна можно временно отключать, изменять, удалять точки останова. Хочется остановиться на следующих возможностях этого окна:

- Установка и снятие галочки приводит к включению и отключению точки останова без её удаления. Это позволяет сохранять информацию об условиях срабатывания, числа попаданий и т.д.
- Нажатие правой кнопки мыши на выбранной точке останова приведёт к открытию меню управления.



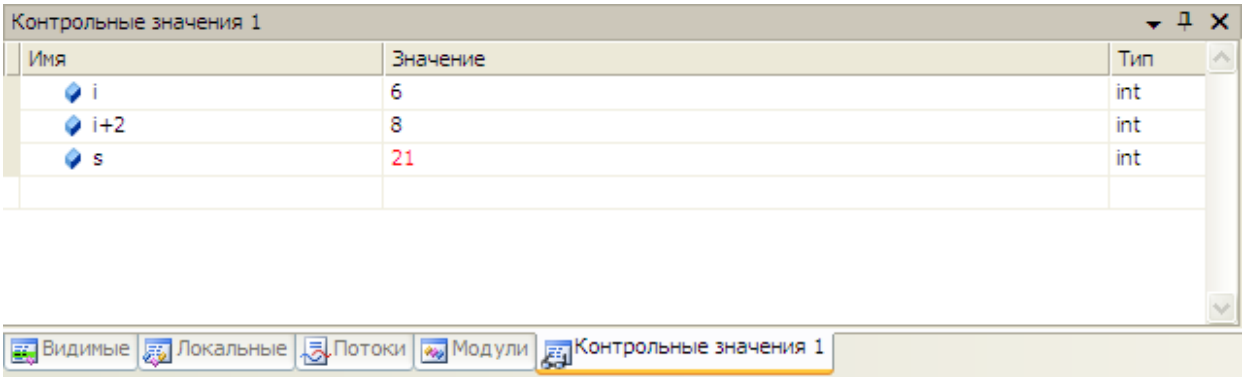
Используя меню управлению можно:

- Изменить расположение точки останова (пункт «Расположение…»).
- Изменить условие остановки программы (пункт «Условие…»). Данный пункт полезен, когда требуется остановиться на заданном операторе при выполнении определённого условия, например, при достижении счётчиком цикла значения 80.
- Изменить условие остановки программы согласно количеству раз выполнения оператора с этой точкой останова (пункт «Число попаданий…»).
- Выставлять дополнительные правила срабатывания (пункт «Фильтр»). Например, точка останова срабатывает для процесса с заданным номером (типичная ситуация для параллельного программирования).
- Задать дополнительные действия при срабатывании точки останова (пункт «При попадании…»). Например, напечатать сообщение или выполнить макрос.

Доступ к переменным

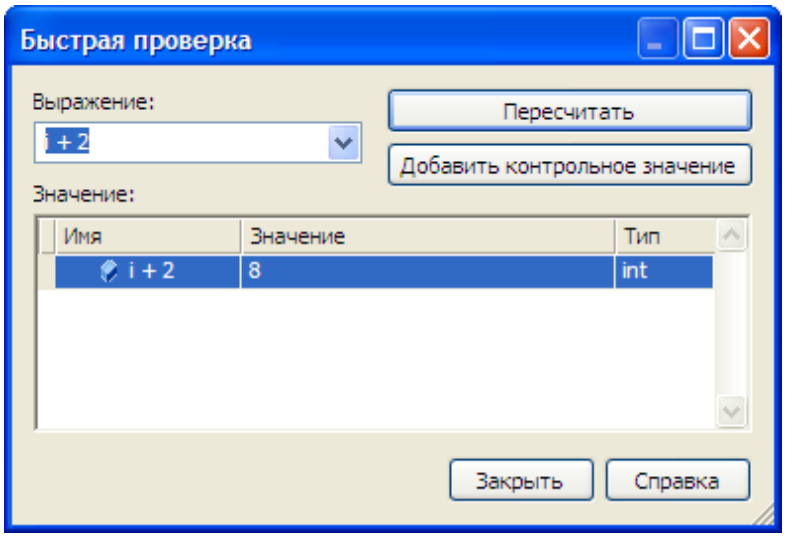
Во время отладки программист может посмотреть значения переменных в окнах «Видимые», «Локальные» и «Контрольные значения». Все они доступны в режиме отладки:

- В окне «Видимые» (меню «Отладка» - «Окна» - «Видимые») отображаются переменные, которые использовались на прошлом шаге и будут использоваться в следующем (текущем) шаге. Название несколько вводит в заблуждение, поскольку можно было бы подумать, что здесь отображаются все переменные в текущей области видимости. Только в этой вкладке показано, какие значения возвращают функции при выполнении операторов.
- В окне «Локальные» (меню «Отладка» - «Окна» - «Локальные») отображаются все локальные переменные текущей функции.
- В окне «Контрольные значения» (меню «Отладка» - «Окна» - «Контрольные значения» - «Контрольные значения 1/2/3/4») отображаются значения переменных, список которых программист определяет сам. При этом программист может не просто ввести имя переменной, но и целое выражение (например, «i+2»). К сожалению, в выражениях нельзя использовать вызовы функций.



При необходимости пользователь может изменить значения переменных. Для этого в колонке значение напротив нужной переменной пользователь, нажав дважды левой клавишей мыши, может присвоить новое значение этой переменной.

Проверку значения того или иного выражения можно осуществить в окне «Быстрая проверка» меню «Отладка».



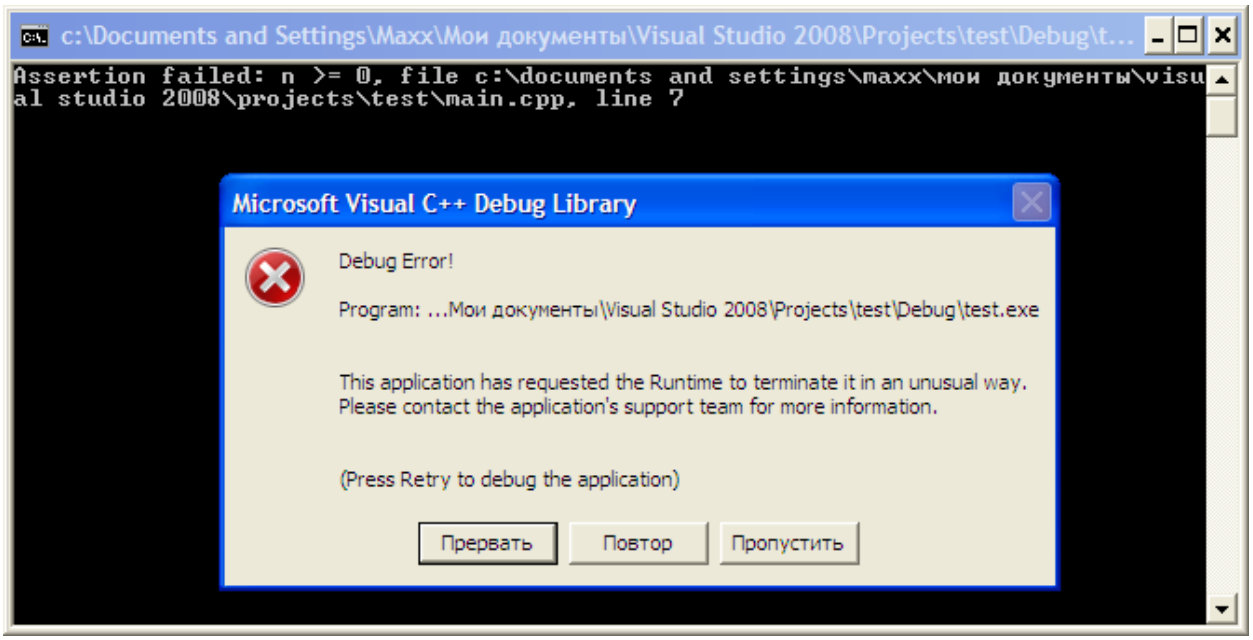
Утверждения

Одним из способов отладки программы является использование утверждений. Утверждение – это логическое условие, которое должно быть всегда истинно. Если такое условие ложно, то значит, имеются ошибки в логике программы.

Например, в функции «sum1n» параметр «n» не должен принимать отрицательные значения. Поскольку тип выбран неверно, то такая ситуация может возникнуть, хотя и не должна. Внедрим в код утверждение, которое проверяет, что параметр «n» не должен принимать отрицательные значения. Для этого подключим библиотеку «cassert» и добавим проверку условия с помощью макроса «assert».

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

int sum1n(int n)
{
	assert(n>= 0);
	int i, s=0;
	for (i=0; i<n; i++)
		s += i;
	return s;
}

int main()
{
	int sum;
	sum = sum1n(-1);
	cout<<"sum = "<<sum<<endl;
	return 0;
}
```

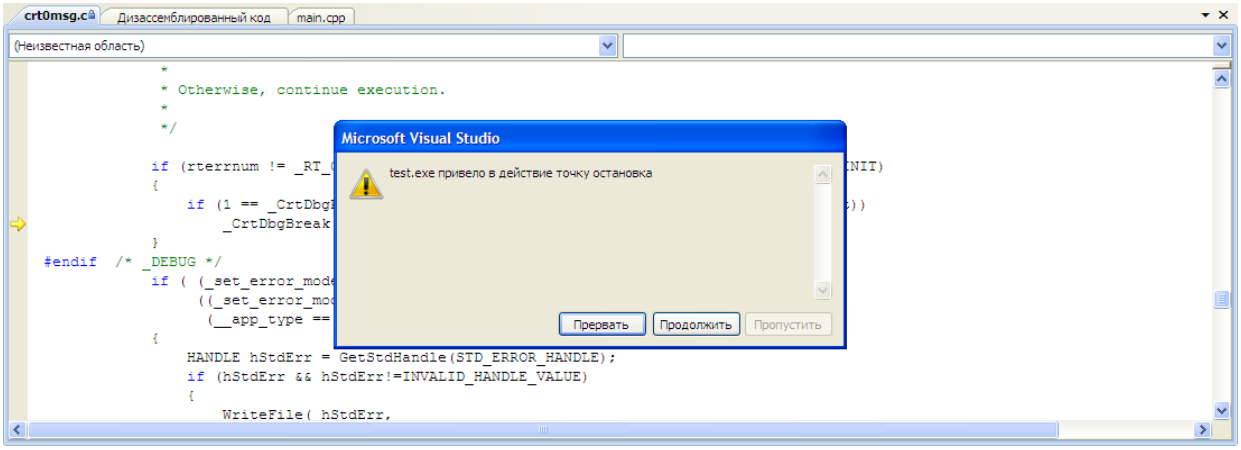
При выполнении такой программы пользователь увидит следующее окно:



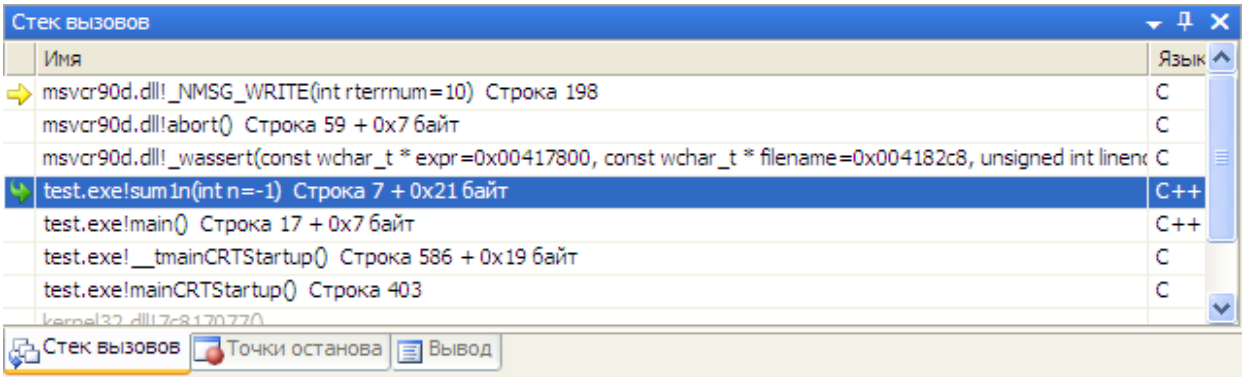
В окне приложения сообщается, какое условие было нарушено и в каком месте программы. Пользователю предлагается выбор:

- Кнопка «Прервать» позволяет прервать выполнение программы и перейти к редактированию исходного кода программы.
- Кнопка «Повтор» (рекомендуется) позволяет приостановить выполнение программы и перейти в режим пошаговой отладки.
- Кнопка «Пропустить» позволяет выполнить программу дальше, не обращая внимания на неправильные данные.

При нажатии на кнопку «Повтор» пользователь увидит следующее:



Далее необходимо нажать кнопку «Прервать» и просмотрев стек вызовов определить, в каком месте программы произошёл сбой.



Из рисунка видно, что сбой произошёл в файле «test.exe» в функции «sum1n» в строке 7. Указаны значения параметров функции «sum1n» («int n=-1»). Далее можно посмотреть, откуда была вызвана эта функция. В данном случае вызов был произведён из функции «main».

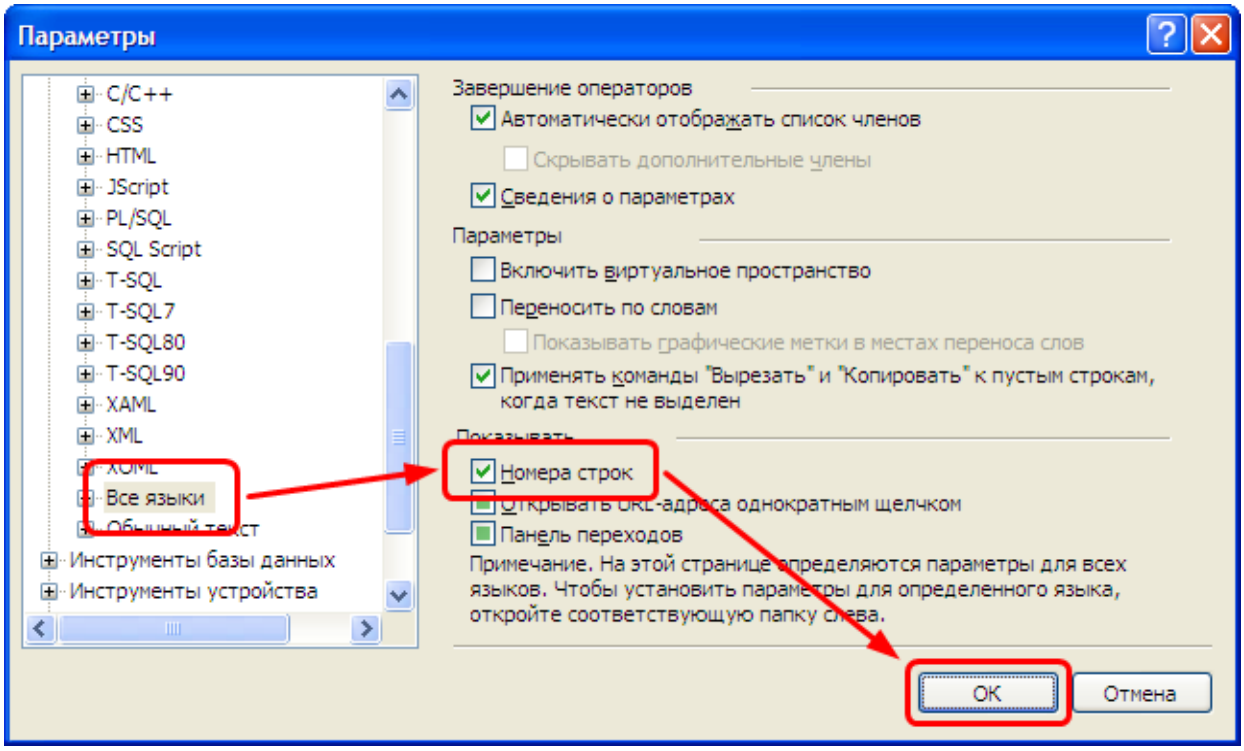
Таким образом, записывая утверждения в различных частях программы, мы можем быстрее выявить некорректное поведение и аномальные значения переменных.

Может возникнуть впечатление, что утверждения замедляют работу программы, проверяя каждый раз условия. Это верно для программ конфигурации «Debug». Для конфигурации «Release» утверждения не компилируются, и программа выполняется без дополнительных проверок.

Приложение 8. Полезные советы по работе в среде VisualC++ 2008

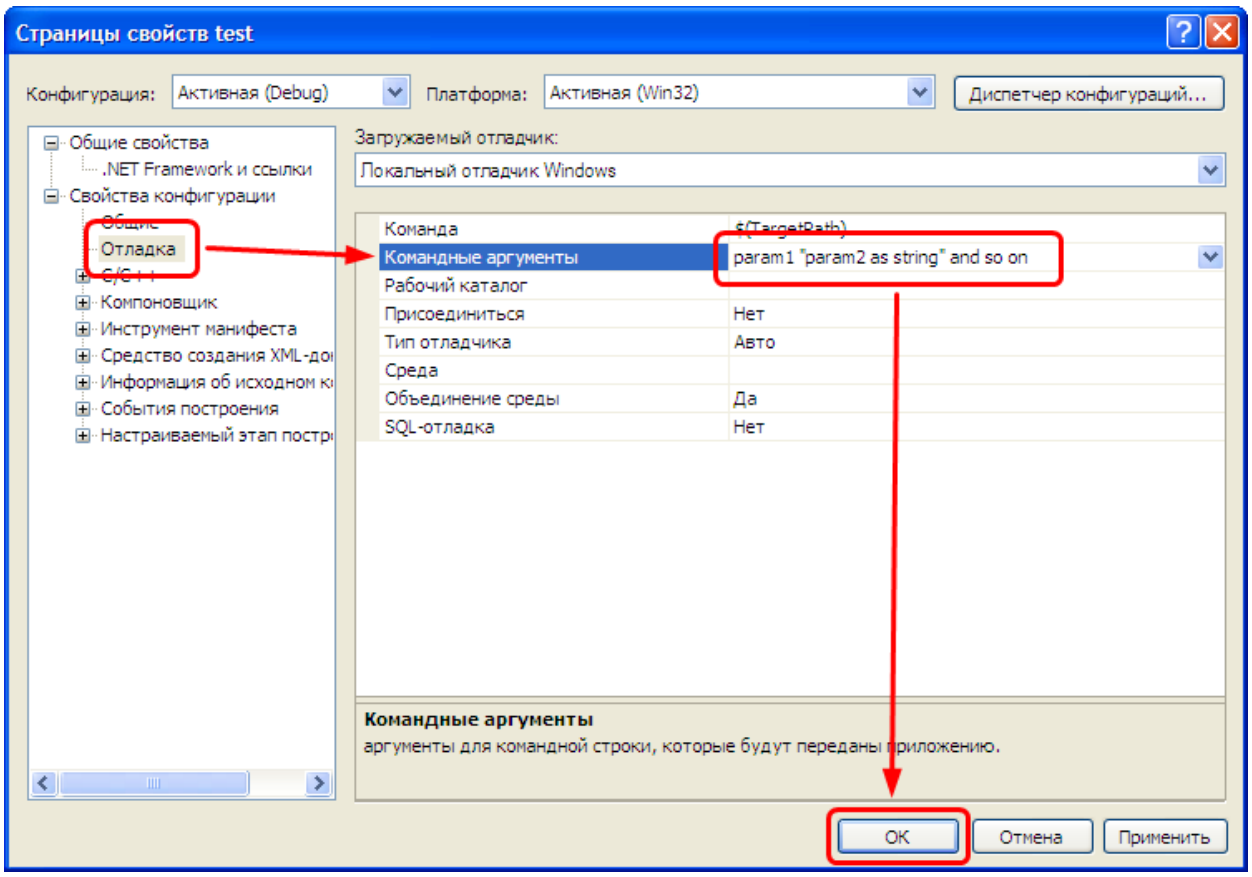
Как включить отображение номеров строк?

Вызвать окно параметров среды Visual Studio, используя меню «Сервис» - «Параметры». В левой части выбрать «Текстовый редактор» - «Все языки». Затем в правой части отметить «Показывать номера строк» и нажать кнопку «ОК».



Как задать параметры командной строки при отладке?

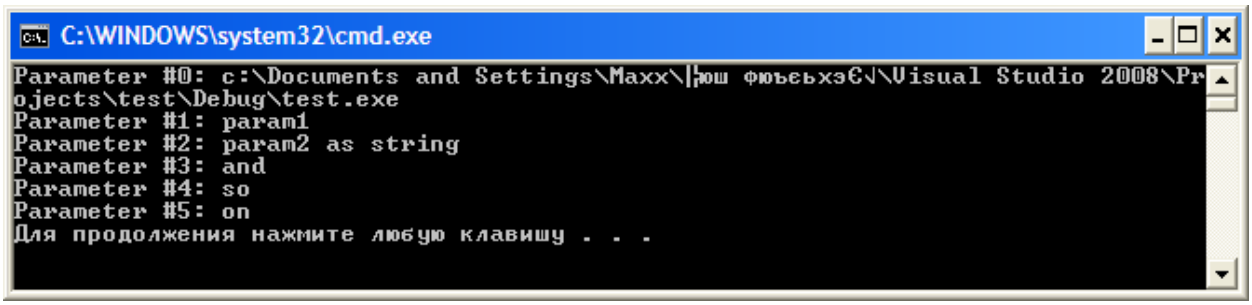
Используя меню («Проект» - «Свойства») или горячие клавиши (Alt+F7) открыть свойства проекта. Слева в дереве конфигурации выбрать «Свойства конфигурации» - «Отладка». В правом части указать командные аргументы. При этом считается, что аргументы программы разделяются пробелами. Если аргумент содержит пробелы, то значение такого аргумента заключается в двойные кавычки. Таким образом на рисунке показано, что в проекте заданы 5 параметров со значениями «param1», «param2 asstring», «and», «so» и «on». После чего нажать на кнопку «ОК». Запуск программы в режиме отладки будет эквивалентен запуску программы из командной строки с этими параметрами.



Доступ к параметрам происходит через параметры функции «main». Таким образом программа

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

int main(int argc, char *argv[])
{
	for (int i=0; i<argc; i++)
		cout<<"Parameter #"<<i<<": "<<argv[i] <<endl;
	return 0;
}
```

отобразит следующее



Как видно из результатов выполнения параметр с индексом 0 всегда определён и содержит полный путь к файлу программы.

Почему окно программы по завершению автоматически закрывается?

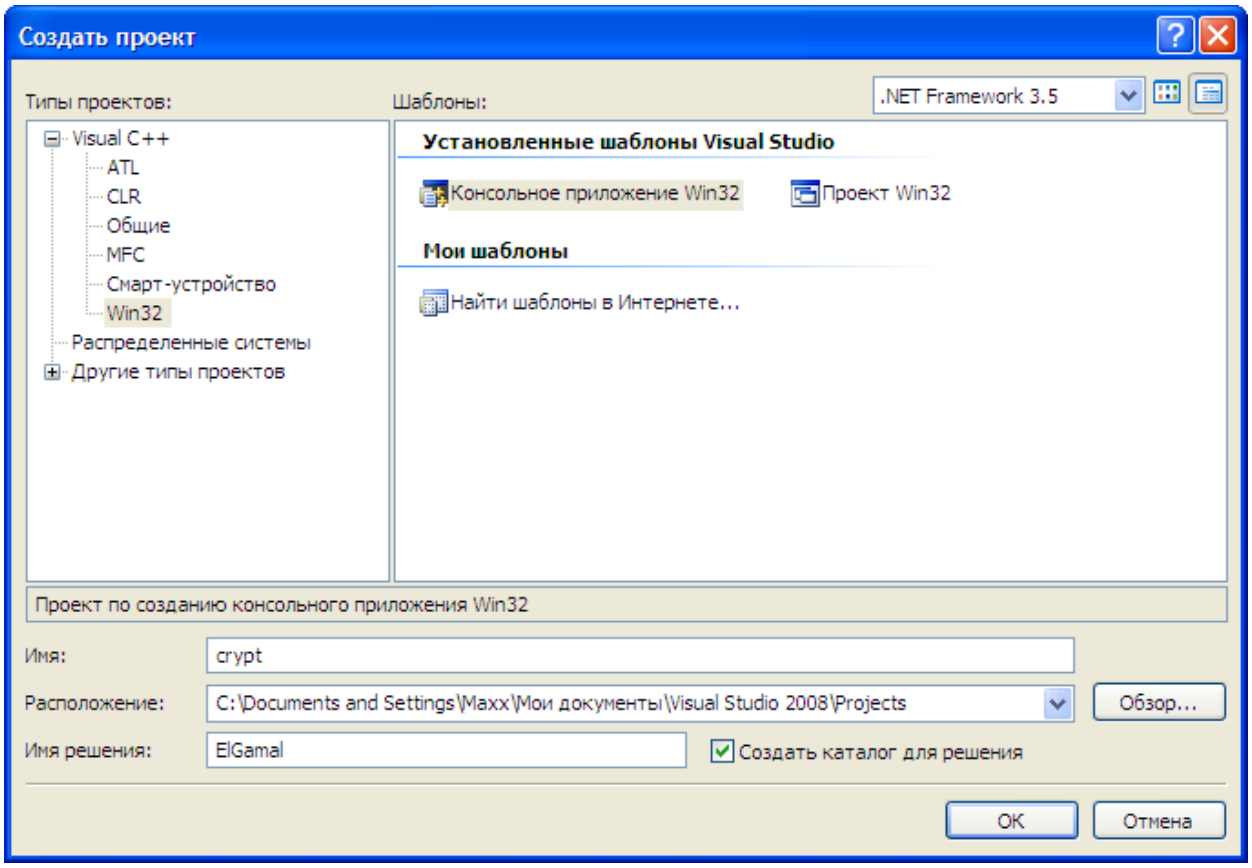
Смотри описание режимов запуска программы (раздел «Режимы запуска программы»).

Как создать решение с несколькими проектами?

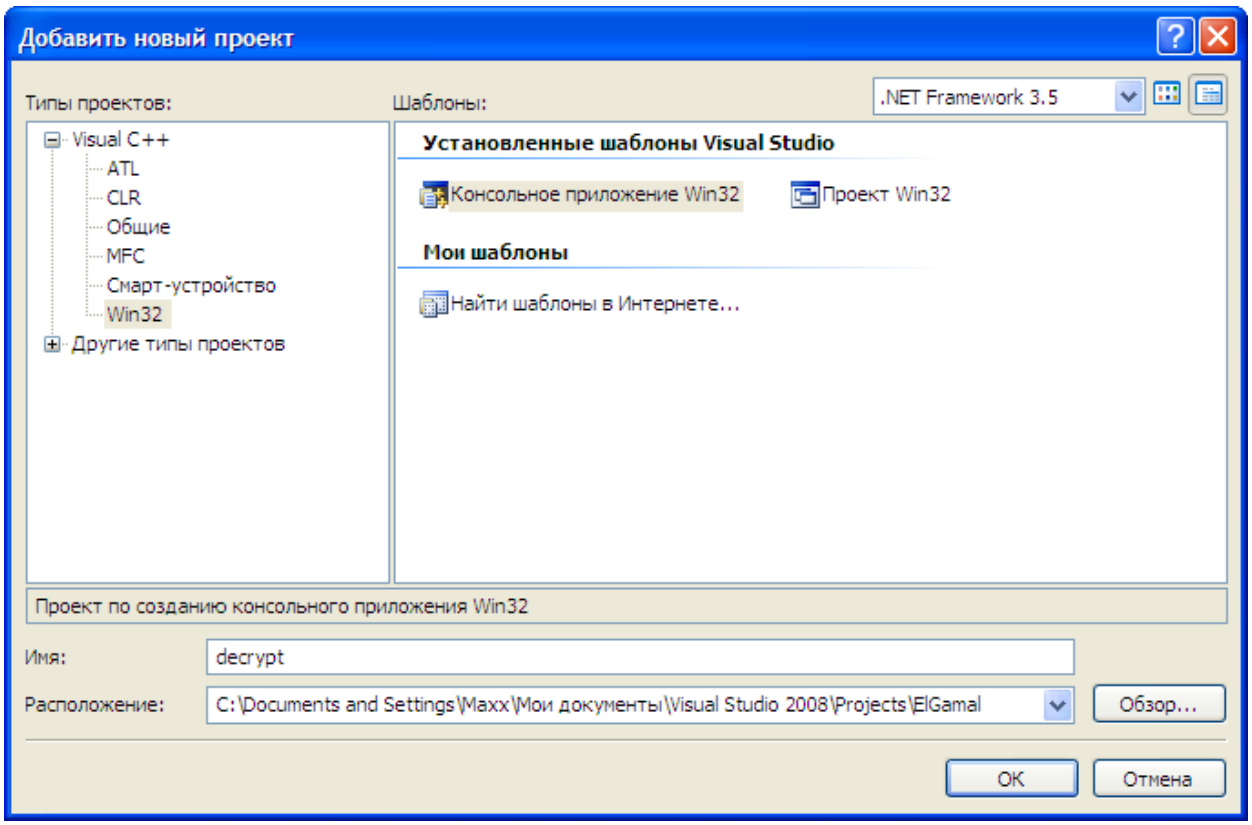
Visual Studio поддерживает два уровня иерархии кода: проект и решение. Проект – это отдельная программа, которая состоит из одного или нескольких файлов. В результате компиляции проекта получается, как правило, исполняемый файл. Решение – это несколько проектов, которые связаны между собой логически. Эти проекты могут иметь общие файлы, а могут и не иметь.

В качестве примера можно привести шифрование по алгоритму Эль-Гамаля. В решение «ElGamal» будут входить два проекта: «crypt» - шифрация данных и «decrypt» - дешифрование. Последовательность действий будет следующей:

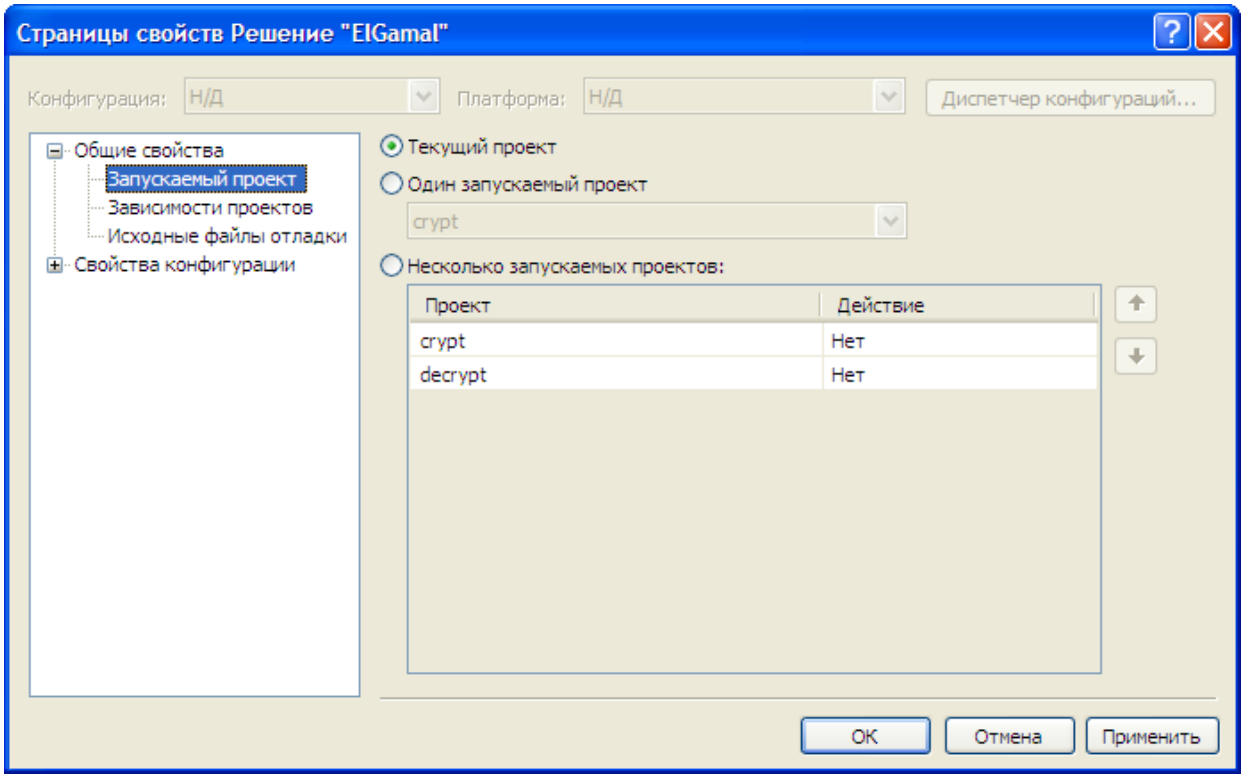
1. Создать решение и первый проект (в данном случае для шифрации данных).
   1.1. Меню «Файл» - «Создать» - «Проект».
   1.2. Выбрать тип проекта (в данном случае «Консольное приложение Win32»).
   1.3. Ввести имя первого проекта (в данном случае «crypt»).
   1.4. Поставить галочку «Создать каталог для решения» и ввести его имя (в данном случае «ElGamal»).



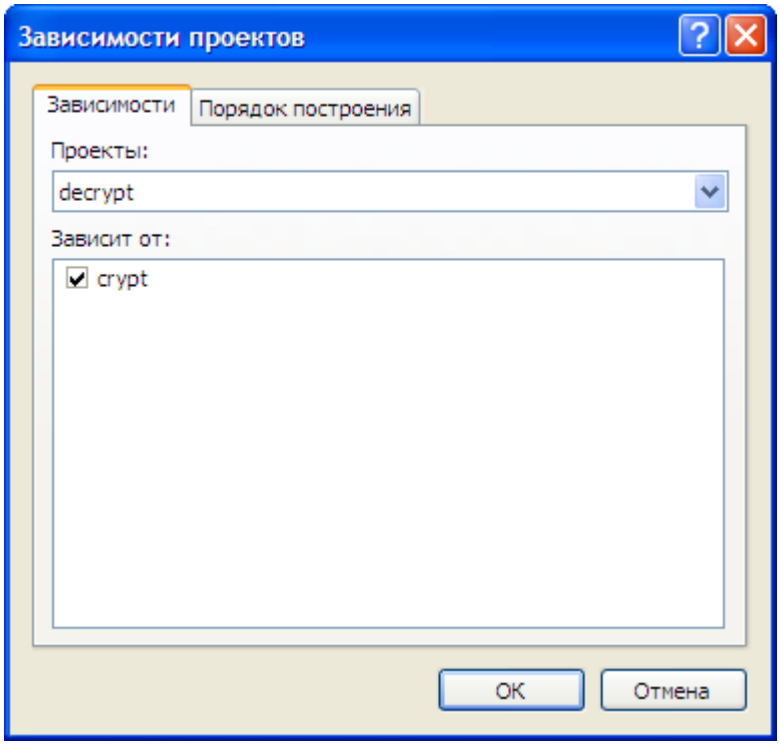
2. Создать проект «decrypt» и добавить его в решение.
   2.1. Меню «Файл» - «Добавить» - «Создать проект».
   2.2. Выбрать тип проекта (в данном случае «Консольное приложение Win32»).
   2.3. Ввести имя второго проекта (в данном случае «decrypt»).



3. Установить запускаемые проекты.
   3.1. Меню «Проект» - «Назначить запускаемые проекты».
   3.2. Выбрать запускаемые проекты. Обычно, это текущий проект. Так проще отлаживать и нет необходимости сразу две программы запускать.

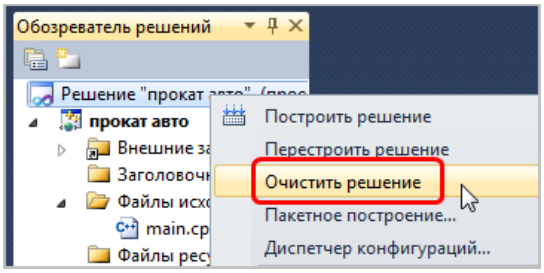


4. Установить зависимости между проектами. Один проект может быть библиотекой (например, библиотека для работы с большими числами произвольного размера), которую используют другие проекты.
   4.1. Меню «Проект» - «Зависимости проектов».
   4.2. Выбрать зависимый проект в выпадающем списке «Проекты».
   4.3. Поставить галочки напротив проектов, от которых он зависит.



Как уменьшить размер проекта?

При работе с проектом Visual Studio создаёт временные файлы. Они нужны только на этапе отладки проекта. После того, как работа с проектом завершена, рекомендуется убрать все лишние файлы (оставив файлы с исходным кодом). Для этого нужно нажать правой кнопкой на решении или проекте (если их в решение несколько), а затем из контекстного меню выбрать «Очистить решение».



Очистка проекта желательна, если нужно отправить проект по электронной почте (например, преподавателю для проверки). Тогда архив будет иметь гораздо меньший размер.

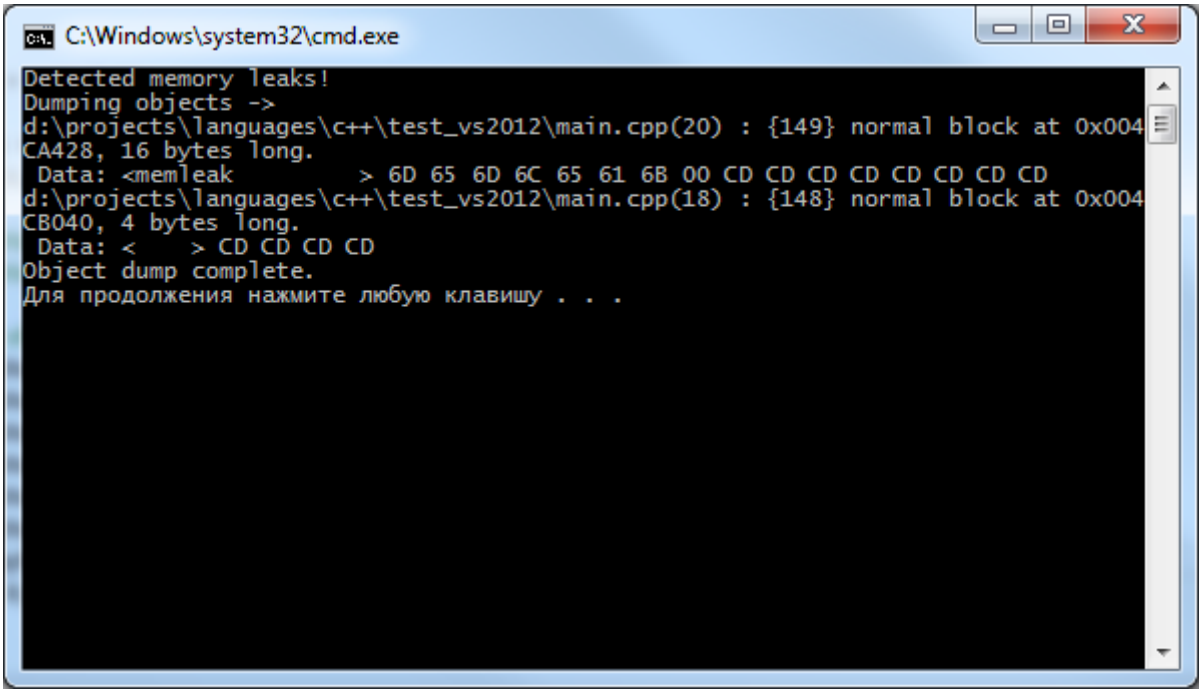
Иногда возникает ситуация, что проект был скомпилирован на одной машине, а затем перенесён на другую, но компиляция на другой машине не выполняется из-за «странных» ошибок. Обычно, очистка проекта и повторная сборка решают эту проблему.

Как обнаружить утечки памяти?

При работе с динамической памятью по разным причинам, но по вине программиста, могут возникать утечки памяти. Для каждого компилятора имеются свои средства их обнаружения. В следующей программе показано, как обнаружить утечки памяти для компиляторов Visual C++ версий 2008, 2010, 2012.

```
// Для обнаружения утечек памяти
#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>

#ifdef _DEBUG
#ifndef DBG_NEW
#define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
#define newDBG_NEW
#endif
#endif

// Далее обычный код
#include <iostream>

int main()
{
	int *p = new int;

	char *s = new char[16];
	strcpy(s, "memleak");

	// delete p;
	// delete [] s;

	// Для обнаружения утечек памяти
	_CrtSetReportMode( _CRT_WARN, _CRTDBG_MODE_FILE );
	_CrtSetReportFile( _CRT_WARN, _CRTDBG_FILE_STDOUT );
	_CrtSetReportMode( _CRT_ERROR, _CRTDBG_MODE_FILE );
	_CrtSetReportFile( _CRT_ERROR, _CRTDBG_FILE_STDOUT );
	_CrtSetReportMode( _CRT_ASSERT, _CRTDBG_MODE_FILE );
	_CrtSetReportFile( _CRT_ASSERT, _CRTDBG_FILE_STDOUT );
	_CrtDumpMemoryLeaks();
	return 0;
}
```

Если в программе имеются утечки памяти, то по завершению работы программы будет отображена соответствующая информация:



Если утечек нет, то подобный вывод отсутствует. Самой важной информацией в выводимых сообщениях является отображение имени файла и номера строки, где были выделены неосвобождённые блоки. Более детально ознакомиться со структурой формата вывода можно по следующей ссылке http://msdn.microsoft.com/ru-ru/library/vstudio/x98tx3cf%28v=vs.110%29.aspx.

Приложение 9. Печать русских букв в среде Visual C++ 2008

Описание проблемы

При выводе информации на русском языке Visual C++ некорректно отображает русские символы.

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

int main()
{
	cout << "Привет!" << endl;
	return 0;
}
```

Ниже показан результат выполнения этой программы:

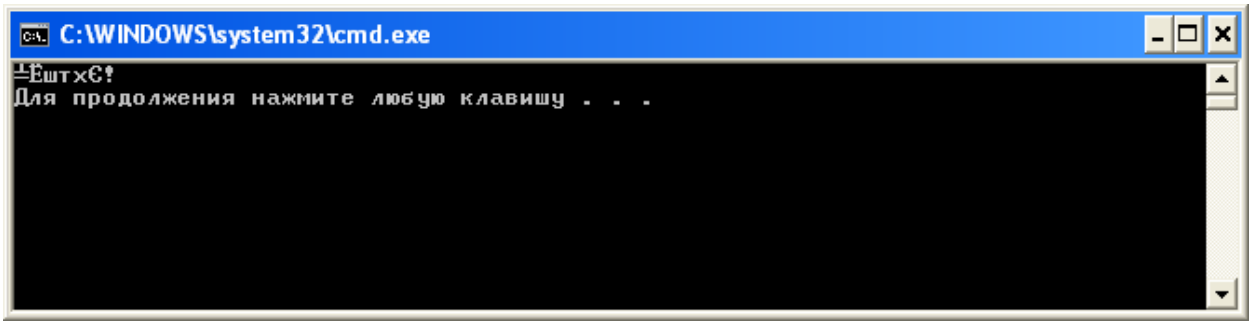


Рис. 12. Пример некорректного отображения русских букв

Проблема возникает из-за того, что используются различные кодировки текста для приложений Windows и консольных приложений. В качестве Windows-приложения выступает «VisualStudio», в котором набирается русский текст (слово «Привет»). В качестве консольного приложения выступает программа. В первом случае используется кодировка 1251, во втором – 866.

Есть несколько способов решения данный проблемы.

Способ 1

Не использовать русские буквы, а писать сообщения на транслите (например, «Privet!») или английском языке.

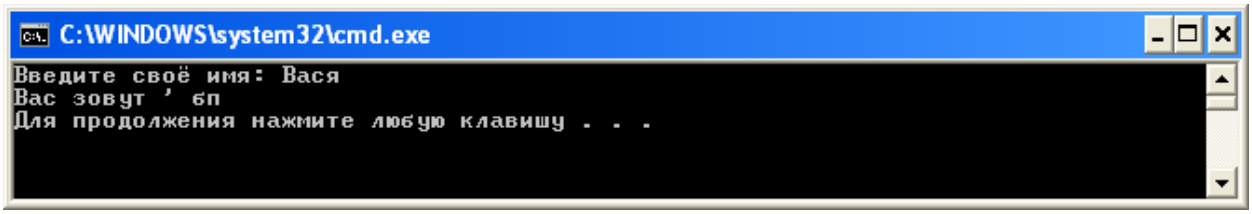
Способ 2

Использование функции «setlocale» позволяет частично решить проблему, а именно **только** корректного **вывода** информации. Сама по себе функция «setlocale» призвана установить региональные настройки. Использование региональных настроек позволяет корректно сравнивать слова, написанные на алфавите отличном от английского, открывать файлы, содержащие буквы национального алфавита, использовать понятные пользователю разделители разрядов в числе и разделителей даты, времени и т.д.

```
#include <iostream>
#include <string>
```

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

int main()
{
	string name;
	setlocale(LC_ALL, "rus");
	cout<<"Введите своё имя: ";
	cin>>name;
	cout<<"Вас зовут "<<name<<endl;
	return 0;
}
```

Пример выполнения:

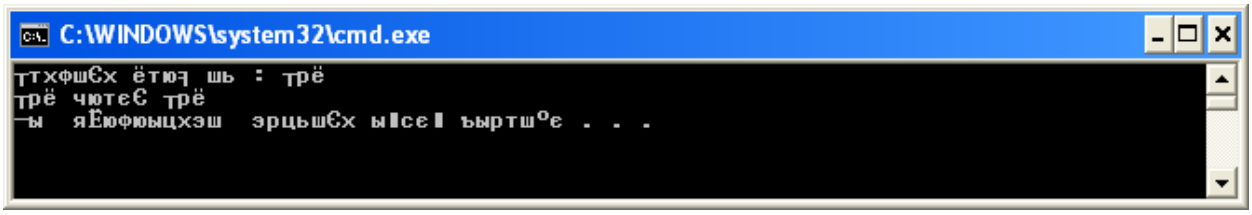


Способ 3

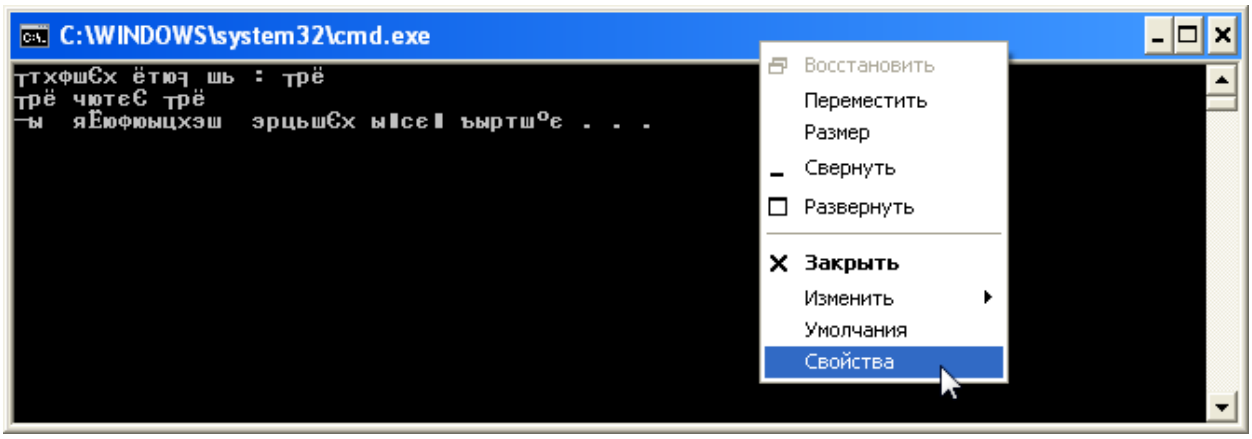
Ещё одним способом изменения кодировки является использование функций «SetConsoleCP» и «SetConsoleOutputCP», которые изменяют кодировку входных и выходных данных соответственно.

```
#include <iostream>
#include <string>
#include <Windows.h>
using namespace std;

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);

	string name;
	cout<<"Введите своё имя: ";
	cin>>name;
	cout<<"Вас зовут "<<name<<endl;
	return 0;
}
```

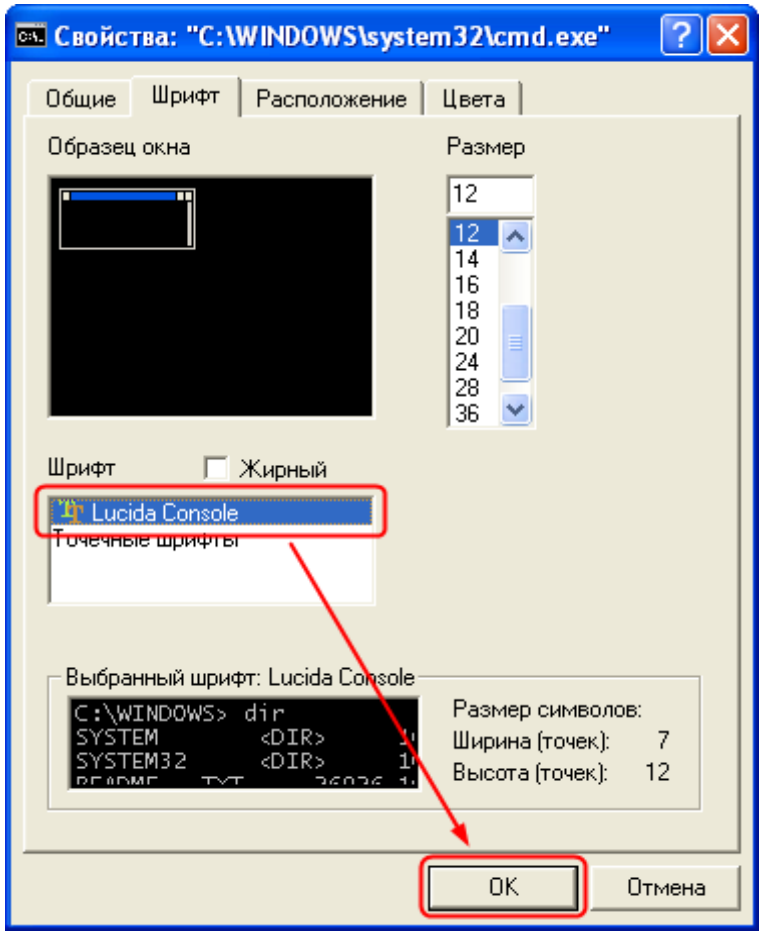
Однако при запуске данного примера мы увидим следующее:

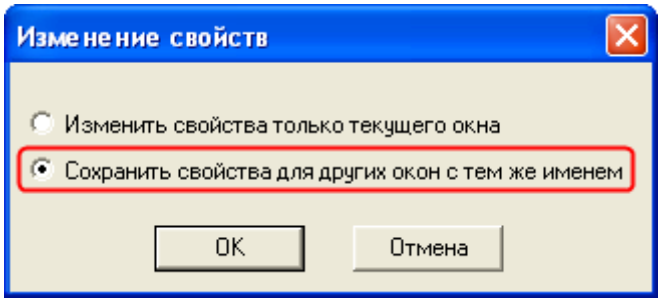


Некорректное отображение информации связано с тем, что выбран неправильный шрифт. Для того, чтобы изменить шрифт необходимо правой кнопкой мыши нажать на заголовок окна и выбрать меню «Свойства».

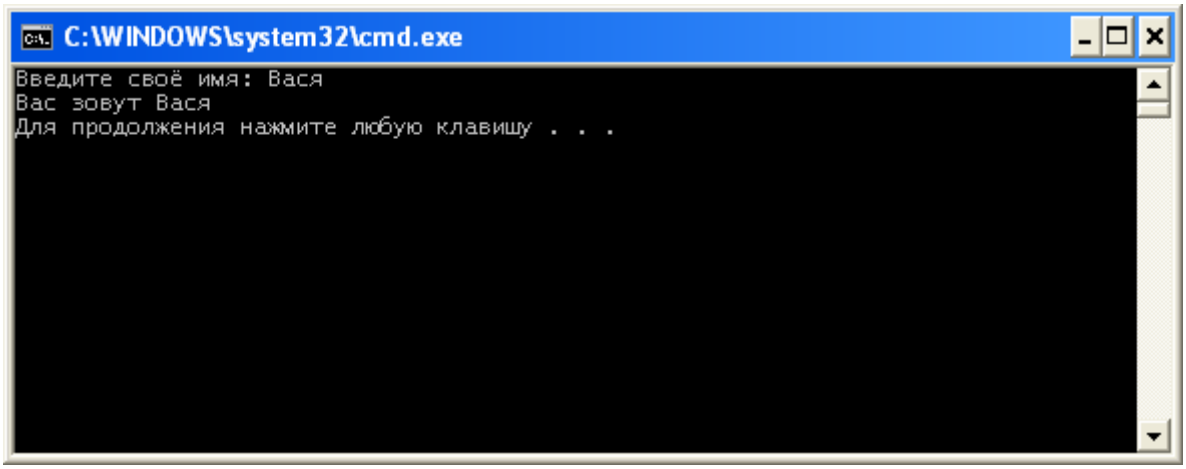


Выбрать шрифт «LucidaConsole» и нажать «ОК».





Теперь те же данные отображаются и обрабатываются корректно.



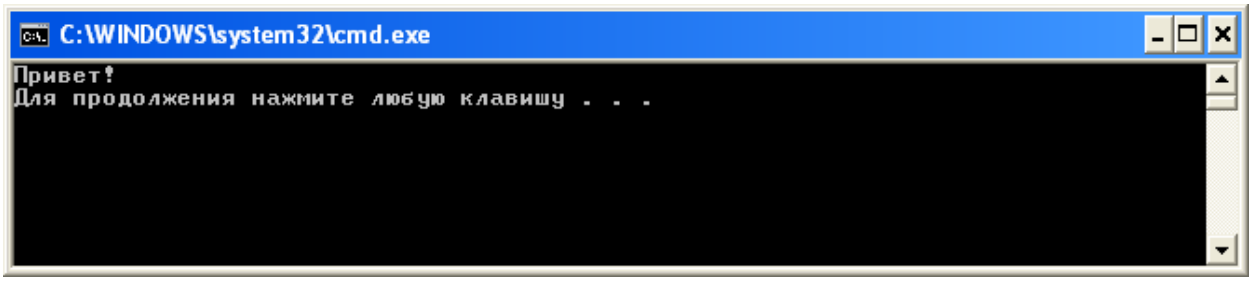
Способ 4

Производить ручное преобразование информации с использованием функций «OemToChar» (для вводимых данных) и «CharToOem» (для выводимых данных).

Пример:

```
#include <iostream>
#include <Windows.h>
using namespace std;

int main()
{
	char str[100];
	CharToOem(L"Привет!", str);

	cout << str << endl;
	return 0;
}
```

Результат работы:



Выводы

При выполнении лабораторных работ следует использовать способ 2 (если используется только вывод) или способ 3 (использует ввод и вывод информации на русском языке).

Ниже представлены выводы безотносительно лабораторных работ.

Важно понимать, что для приложений, написанных в VisualStudio, родной является кодировка Windows. Поэтому правильно будет конвертировать входные данные при вводе и обрабатывать в родной кодировке.

Наиболее правильным является ручное преобразование данных при вводе и выводе информации. Такая программа будет более сложная, но и более контролируема.

Наиболее легким и правильным вариантом является использование функций «SetConsoleCP», «SetConsoleOutputCP» и «setlocale». В этом случае ввод и вывод, а также обработка информации производятся автоматически. Недостатком этого способа можно считать ручное установление шрифта консоли.

Если программа обрабатывает только числовые данные, то наиболее простым и подходящим вариантом будет использование «setlocale».

В любом случае ручное преобразование данных не стоит игнорировать. Ниже представлен код файла «russian.h», который позволяет упростить использование функций «OemToChar»и «CharToOem».

```
#pragmaonce

/*
	Версия:		1.0
	Автор:		ГУАП, Кафедра 43, Суслов М.Н.
	Дата: 6.12.2009 г.

	Данный файл позволяет переконвертировать символы национального алфавита
	для корректного отображения в консольных приложениях.

	Эта проблема возникает из-за того, что используются различные кодировки
	для приложений Windows (в данном случае «Visual Studio», в котором
	набирается русский текст) и консольных приложений (в данном случае
	разрабатываемая программа, в которой отображается русский текст,
	набранный в другой кодировке).

	Для решения этой проблемы имеются следующие функции:
	const char* dos2winA(const char* cstr);
	const wchar_t* dos2winW(const wchar_t* wstr);
	const char* win2dosA(const char* cstr);
	const wchar_t* win2dosW(const wchar_t* wstr);

	А также вспомогательные:
	const std::string dos2win(const std::string &str);
	const std::wstring dos2win(const std::wstring &str);
	const std::string win2dos(const std::string &str);
	const std::wstring win2dos(const std::wstring &str);

	Пример:
	-------------------------------------------------------------
	#include <iostream>
	using namespace std;
	#include "russian.h"

	int main()
	{
		cout << win2dosA("Привет!") << endl;
		win2dosA(NULL);
		return 0;
	}
	-------------------------------------------------------------
```

```
      Чтобы освободить занятую память в конце программы следует вызвать
      использованные функции с параметром NULL. Поэтому программа может
      выглядеть следующим образом:
      ------------------------------------------------------------
      #include <iostream>
      using namespace std;
      #include "russian.h"

      void rus_exit()
      {
            win2dosA(NULL);
            win2dosW(NULL);
            dos2winA(NULL);
            dos2winA(NULL);
      }

      int main()
      {
            atexit(rus_exit);
            cout << win2dosA("Введите своё имя: ");
            return 0;
      }
------------------------------------------------------------*/

#include <windows.h>
#include <cstring>
#include <exception>
#include <string>

const char* dos2winA(const char* cstr)
{
      static char* cbuf = NULL;

      if (cbuf != NULL)
      {
            delete [] cbuf;
            cbuf = NULL;
      }

      if (cstr != NULL)
      {
            cbuf = new char[strlen(cstr)+1];
            if (!OemToCharA(cstr, cbuf))
                  throw new exception("russian.h: dos2winA: OemToCharA");
      }

      return cbuf;
}

const wchar_t* dos2winW(const wchar_t* wstr)
{
      static wchar_t* wbuf = NULL;

      if (wbuf != NULL)
      {
```

```
            delete [] wbuf;
            wbuf = NULL;
      }

      if (wstr != NULL)
      {
            size_t len = wcslen(wstr) + 1;
            char* cbuf = new char[len];

            size_t num_can;
            if (wcstombs_s(&num_can, cbuf, len, wbuf, len-1))
                  throw new exception("russian.h: dos2winW: wcstombs_s");
            if (num_can != len)
                  throw new exception("russian.h: wcstombs_s");

            if (!OemToCharW(cbuf, wbuf))
                  throw new exception("russian.h: dos2winW: OemToCharW");

            delete [] cbuf;
            cbuf = NULL;
      }

      return wbuf;
}

const char* win2dosA(const char* cstr)
{
      static char* cbuf = NULL;

      if (cbuf != NULL)
      {
            delete [] cbuf;
            cbuf = NULL;
      }

      if (cstr != NULL)
      {
            cbuf = new char[strlen(cstr)+1];
            if (!CharToOemA(cstr, cbuf))
                  throw new exception("russian.h: win2dosA: CharToOemA");
      }

      return cbuf;
}

const wchar_t* win2dosW(const wchar_t* wstr)
{
      static wchar_t* wbuf = NULL;

      if (wbuf != NULL)
      {
            delete [] wbuf;
            wbuf = NULL;
      }
```

```
	if (wstr != NULL)
	{
		size_t len = wcslen(wstr) + 1;
		char* cbuf = new char[len];

		if (!CharToOemW(wstr, cbuf))
			throw new exception("russian.h: win2dosW: CharToOemW");

		size_t num_can;
		if (mbstowcs_s(&num_can, wbuf, len, cbuf, len-1))
			throw new exception("russian.h: win2dosW: mbstowcs_s");
		if (num_can != len)
			throw new exception("russian.h: win2dosW: mbstowcs_s");

		delete [] cbuf;
		cbuf = NULL;
	}

	return wbuf;
}

#ifdef UNICODE
#define dos2win dos2winW
#define win2dos win2dosW
#else
#define dos2win dos2winA
#define win2dos win2dosA
#endif

const std::string dos2win(const std::string &str)
{
	return std::string(dos2winA(str.c_str()));
}

const std::wstring dos2win(const std::wstring&str)
{
	return std::wstring(dos2winW(str.c_str()));
}

const std::string win2dos(const std::string &str)
{
	return std::string(win2dosA(str.c_str()));
}

const std::wstring win2dos(const std::wstring &str)
{
	return std::wstring(win2dosW(str.c_str()));
}
```