

Документ подписан простой электронной подписью

Информация о владельце:

ФИО: Локтионова Оксана Геннадьевна

Должность: проректор по учебной работе

Дата подписания: 02.09.2021 17:07:00

Уникальный программный ключ:

0b817ca911e6668abb13a5d426d39e5f1c11eabbf75e943df4a4851fda56d089

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ

Проректор по учебной работе

О.Г. Локтионова

«02» 09 2021



**ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ФУНКЦИЙ НА
ЯЗЫКЕ C++**

Методические указания по выполнению лабораторной работы по
дисциплине "Программирование на языках высокого уровня" для
студентов направления подготовки 09.03.04 "Программная инженерия"

Курск 2017

УДК 681.3.06(071.8)

Составители: Т.М. Белова, В.Г. Белов

Рецензент

Кандидат технических наук, доцент кафедры программной инженерии ЮЗГУ И.Н. Ефремова

Программирование с использованием функций на языке C++: методические указания по выполнению лабораторной работы по дисциплине "Программирование на языках высокого уровня" для студентов направления подготовки 09.03.04 "Программная инженерия" / Юго-Зап. гос. ун-т; сост. Т.М. Белова, В.Г. Белов. Курск, 2017. – 24 с.

Содержат основные теоретические положения и приемы разработки программ с использованием функций на языке C++, пример решения типовой задачи, индивидуальные задания и контрольные вопросы к защите лабораторной работы.

Методические указания соответствуют требованиям рабочей программы по дисциплине "Программирование на языках высокого уровня".

Предназначены для студентов направления подготовки 09.03.04 «Программная инженерия» дневной и заочной форм обучения.

Текст печатается в авторской редакции.

Подписано в печать *27.12.17*. Формат 60x84 1/16.

Усл. печ. л. *14*. Уч.-изд. л. *13*. Тираж 100 экз. Заказ *4393*. Бесплатно.

Юго-Западный государственный университет
305040, Курск, ул.50 лет Октября, 94.

Программирование с использованием функций на языке C++

Цель работы — изучение структуры функции, механизма взаимодействия программы с функциями, рекурсивного метода организации вычислений, получение навыков программирования алгоритмов средней сложности с использованием функций.

Основные понятия

Если в программе встречаются повторяющиеся фрагменты, то целесообразно эти фрагменты оформить как подпрограмму. Подпрограммы представляют собой относительно самостоятельные фрагменты программы, оформленные особым образом и снабженные именем. Упоминание этого имени в тексте программы называется вызовом подпрограммы. Все идентификаторы, описанные внутри программы, локализуются в ней. Все имена в пределах подпрограммы, в которой они объявлены, должны быть уникальными и не могут совпадать с именем самой подпрограммы.

Использование подпрограмм позволяет уменьшить объем памяти, занимаемый кодом программы в целом. Однако время выполнения программы при этом несколько возрастает, так как появляются временные издержки на вызов подпрограмм и возврат из подпрограмм.

В C++ подпрограммы представлены функциями.

Функции

Каждую свою функцию программисту необходимо описать, т. е. указать ее заголовок и тело.

В заголовке объявляются тип возвращаемого функцией результата, имя функции и формальные параметры, если они есть.

Формат заголовка функции:

тип результата < имя функции>(список параметров)

тип результата < имя функции>()

Примеры заголовка функции:

double a (double x, int b)

void f(void) или void fl()

Если тип возвращаемого значения не указан, он по умолчанию считается равным **int**. Если функция не возвращает никакого значения, то тип возвращаемого значения объявляется **void**.

Тело функции представляет собой локальный блок, по структуре аналогичный программе:

```
тип результата <имя функции>(формальные параметры) {
    <операторы тела функции>
}
```

Как правило, хотя формально не обязательно, помимо описания функции в текст программы включается также *прототип* функции – ее предварительное объявление. Прототип представляет собой тот же заголовок функции, но с точкой с запятой («;») в конце. Кроме того, в прототипе можно не указывать имена формальных параметров.

Введение в программу прототипов функций преследует несколько целей:

1. Позволяет использовать в данном модуле функцию, описанную в каком-нибудь другом модуле.
2. Последовательность размещения в модуле описания функции безразлична, т. е. вызывать функцию можно до ее описания.

Вызов функции осуществляется упоминанием имени функции в соответствующих выражениях наряду с переменными и константами. Оператор вызова функции состоит из имени (идентификатора функции) и списка фактических параметров, отделенных друг от друга запятыми и заключенными в круглые скобки.

Допускается также вызов функции, не использующий возвращаемого ею значения.

Например:

```
double y=a (x, b); // используется возвращаемое функцией значение
a (x, b); // возвращаемое функцией значение игнорируется
```

Выход из функции может осуществляться следующими способами:

1. Если функция не должна возвращать никакого значения, то выход из нее происходит или по достижении закрывающей ее тело фигурной скобки, или при выполнении оператора *return*.

2. Если же функция должна возвращать некоторое значение, то нормальный выход из нее осуществляется оператором

```
return <выражение>;
```

где выражение должно формировать возвращаемое значение и соответствовать типу, объявленному в заголовке функции.

Например:

```
double fsum(double x1, double x2, int a)
{
    return a *(x1+x2);
}
```

Ниже приведен пример функции, не возвращающей никакого значения:

```
void sprint(AnsiString s)
{
    if (s == "") return;
    ShowMessage(s);
}
```

Прервать выполнение функции можно также генерацией какого-то исключения. Наиболее часто в этих целях используют процедуру **Abort**, не связанную с каким-то сообщением об ошибке. Применение функции **Abort** выводит управление сразу вверх из всех вложенных друг в друга вызовов функций.

Локальные и глобальные переменные

Все объявления в теле функции носят *локальный* характер. Объявленные переменные доступны только внутри данной функции. Если их имена совпадают с именами каких-то глобальных переменных модуля, то эти внешние переменные становятся невидимыми и недоступными. В этих случаях получить доступ к глобальной переменной можно, поставив перед ее именем два двоеточия («::»), т. е. применив унарную операцию разрешения области действия. Например, выражение **::i** означает *глобальную* переменную *i*, даже если в функции она объявлена локальной.

Локальные переменные не просто видны только в теле функции, но по умолчанию они существуют только внутри функции, создаваясь в момент вызова и уничтожаясь в момент выхода из функции. Если требуется этого избежать, соответствующие переменные объявляются со спецификацией *static*.

Передача параметров в функции по значению и по ссылке

Список *формальных* параметров состоит из имен параметров и указаний на их тип. Например, в заголовке

```
double fsum (double x1, double x2, int x3)
```

указано три параметра **x1**, **x2**, **a** и определены их типы. Вызов такой функции может иметь вид

```
double y = fsum (z, x2, x3);
```

Это только один из способов передачи параметров в функцию, называемый *передачей по значению*. Работает он так. В момент вызова функции в памяти создаются временные переменные **x1**, **x2**, **a**, и в них копируются значения аргументов **z**, **x2** и константы **5**. На этом связь между переменными и аргументами разрывается. Можно изменять внутри функции значения **x1**, **x2** и **x3**, но это не отразится на значениях аргументов.

К недостаткам такой передачи параметров по значению относятся:

- затраты времени на копирование значений и затраты памяти на хранение копии;
- невозможность из функции изменять значение некоторых аргументов, что во многих случаях желательно.

Возможен и другой способ передачи параметров – *вызов по ссылке*. В случае вызова по ссылке оператор вызова дает вызываемой функции возможность прямого доступа к передаваемым данным, а также возможность изменения этих данных. Вызов по ссылке исключает расходы на копирование больших объемов данных, но ослабляет защищенность, потому что вызываемая функция может испортить передаваемые в нее данные.

Вызов по ссылке можно осуществить двумя способами – с помощью *ссылочных параметров* и с помощью *указателей*. Чтобы показать, что параметр функции передан по ссылке, после типа параметра в прототипе функции ставится символ амперсанда **&**. Такое же обозначение используется в списке типов параметров в заголовке функции. При вызове такой функции

достаточно указать имя переменной и она будет передана по ссылке. Реально в функцию будет передана не сама переменная, а ее адрес, полученный операцией адресации `&`. Упоминание в теле вызываемой функции переменной по имени ее параметра в действительности является обращением к исходной переменной в вызывающей функции, и эта исходная переменная может быть изменена непосредственно вызываемой функцией.

Например:

```
void square(int &); // Прототип функции вычисления квадрата
void square(int &a) // Заголовок функции
{
    a *=a;          // Изменение значения параметра
}
```

Вызываться подобная функция может обычным способом передачей в нее имени аргумента.

Например:

```
int x1=2;
square (x1);
```

В результате подобного вызова переменная `x1` получит значение 4.

Альтернативной формой передачи параметров по ссылке является использование *указателей*. Тогда адрес переменной передается в функцию операцией косвенной адресации `«*»`. В списке параметров подобной функции перед именем переменной указывается символ `«*»`, что свидетельствует о том, что передается не сама переменная, а указатель на нее. В теле функции тоже перед именем параметра ставится символ операции разыменования `«*»`, чтобы получить доступ через указатель к значению переменной. При вызове функции в нее должна передаваться в качестве аргумента не сама переменная, а ее адрес, получаемый с помощью операции адресации `&`.

Рассмотрим функция `square` с передачей параметра по ссылке с помощью указателя:

```
void square(int *); // Прототип функции вычисления квадрата
void square(int *a) // Заголовок функции
{
    *a *= *a;       // Изменение значения параметра
}
```

Вызов функции может осуществляться так:

```
int x1=2;
square(&x1); //в результате значение x1 = 4
```

При передаче параметров с помощью ссылки необходимо строго следить за соответствием типов формальных и фактических параметров, иначе будет создаваться вспомогательная переменная и связь между фактическими и формальными параметрами прервется.

При работе с *массивами* компилятор всегда переходит к выражению с указателем. В языке *C++ Builder* имя массива является константным указателем адреса элемента массива с нулевым индексом. Отсюда следует, что при использовании имени массива в качестве параметра функции допускается изменять значения элементов массива. Массив нельзя передать как параметр по значению.

Например, если функция *f* должна принимать массив как параметр, ее прототип может иметь вид

```
void f (int a[]);
```

Обращение к такой функции может быть записано так:

```
const n=10;
int a[n];
...
f (a); // вызов функции с параметром массивом
```

Если необходимо запретить изменения в функции параметров, то при описании формальных параметров перед именем переменной следует добавить ключевое слово *const*:

```
void f (const int *a, float *b);
```

Передача параметров по умолчанию

Обычно при вызове функции в нее передается конкретное значение каждого параметра. Но можно указать, что параметр является параметром по умолчанию, и присвоить ему значение по умолчанию. Задавать параметры по умолчанию можно в прототипе функции или в заголовке функции. Параметры по умолчанию должны быть указаны при первом упоминании имени функции, обычно в прототипе. Для задания значения параметру после имени

параметра ставится символ «=», после которого записывается значение по умолчанию.

Аргументы по умолчанию должны быть самыми последними в списке параметров функции. Например:

```
void f(double a, char s = '*', int i = 2);
```

Здесь параметры *s* и *i* являются параметрами по умолчанию.

Если при вызове функции параметр по умолчанию не указан, то в функцию автоматически передается его значение по умолчанию. Пропускать при вызове можно только некоторое число последних параметров в списке.

Правильный вызов функции:

```
f(2.5, '/', 5);
```

```
f(2.5, '/');
```

```
f(2.5);
```

Неправильный вызов функции:

```
f(2.5, ,5);
```

Функции с переменным числом параметров

Иногда в функцию требуется передавать некоторое число фиксированных параметров плюс неопределенное число дополнительных параметров. Такие функции называются *функциями с переменным числом параметров*. В этом случае заголовок функции имеет вид

```
тип имя_функции (список_аргументов,...)
```

Многоточие сообщает компилятору, что контроль типов и количества параметров при вызове функции проводить не следует.

При реализации механизма определения количества параметров и их типов используется два подхода:

- 1) один из параметров определяет число дополнительных параметров функции;
- 2) в список аргументов последним задается параметр-индикатор, указывающий на окончание списка параметров.

Переход от одного параметра к другому в обоих случаях осуществляется с помощью указателей.

Рекурсия

Рекурсия – это такой способ организации вычислительного процесса, при котором подпрограмма в ходе выполнения составляющих ее операторов обращается сама к себе.

При выполнении правильно организованной рекурсивной подпрограммы осуществляется многократный переход от некоторого текущего уровня организации алгоритма к нижнему уровню последовательно до тех пор, пока не будет получено тривиальное решение поставленной задачи.

Структура рекурсивной процедуры может принимать три разных формы:

1. Форма с выполнением действий до рекурсивного вызова (с выполнением действий *s* на рекурсивном спуске):

```
unsigned rec(unsigned n)
{
    s; // действия s выполняются на рекурсивном спуске
    if (условие)
        rec(n);
}
```

2. Форма с выполнением действий после рекурсивного вызова (с выполнением действий *s* на рекурсивном возврате):

```
unsigned rec(unsigned n)
{
    if (условие)
        rec(n);
    s; // действия s выполняются на рекурсивном возврате
}
```

3. Форма с выполнением действий как до, так и после рекурсивного вызова (с выполнением действий как на рекурсивном спуске, так и на рекурсивном возврате):

```
unsigned rec(unsigned n)
{
    s1;
    if (условие)
        rec(n);
    s2;
}
```

```

unsigned rec(unsigned n)
{
    if (условие)
    {
        s1;
        rec(n);
        s2;
    }
}

```

Все формы рекурсивных процедур находят применение на практике. Глубокое понимание рекурсивного механизма и умение управлять им является необходимым качеством квалифицированного программиста.

Пример программирования с использованием подпрограмм

Задание. Вычислить функцию

$$S = \frac{\cos x}{1!} + \dots + \frac{\cos Nx}{N!},$$

где N – количество членов ряда.

Структура программы будет включать в себя два модуля – головную программу с именем Project1 и модуль с именем Unit1, связанный с основной формой.

Реализуем подпрограмму вычисления факториала двумя способами – в виде функции и рекурсивным методом.

1. На рисунке 1 – разработка алгоритма:

– входные данные: x – вещественная переменная, являющаяся аргументом функции $\cos(x)$; n – целочисленная переменная, обозначающая количество членов ряда;

– выходные данные: s – вещественная переменная, значение которой есть сумма членов ряда;

– промежуточные данные: k – целочисленная переменная, используемая как счетчик цикла; r – целочисленная переменная.

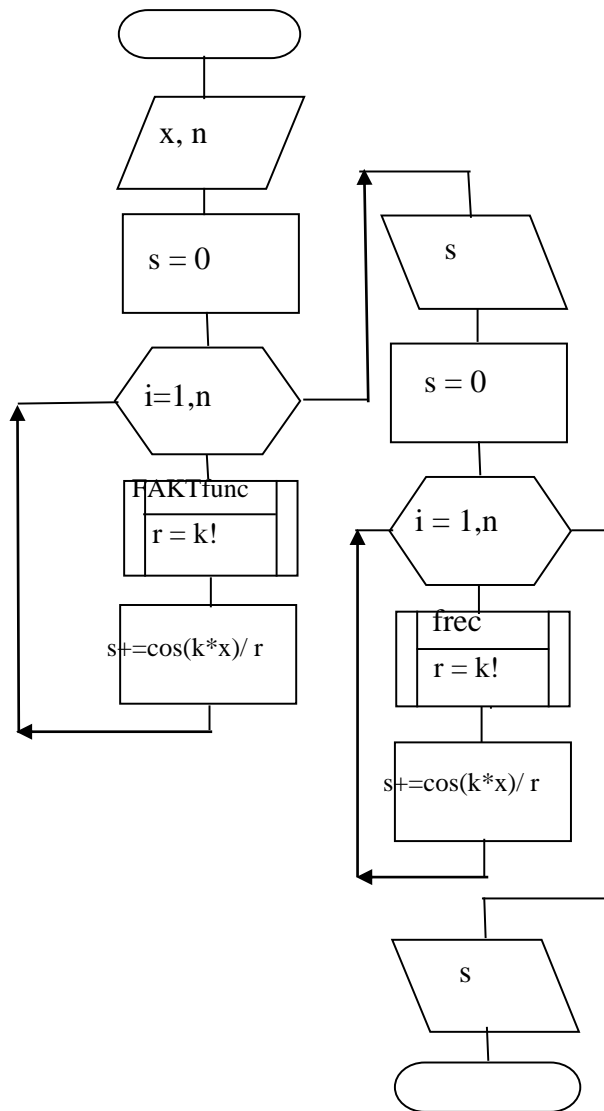


Рисунок 1 – Схема алгоритма основной программы

2. На рисунке 2 – функция ФАКТfunc, которая вычисляет $k!$:

- входные данные: k – целочисленная переменная;
- выходные данные: r – целочисленная переменная, являющаяся значением $k!$;
- промежуточные данные: i – целочисленная переменная, используемая как счетчик цикла.

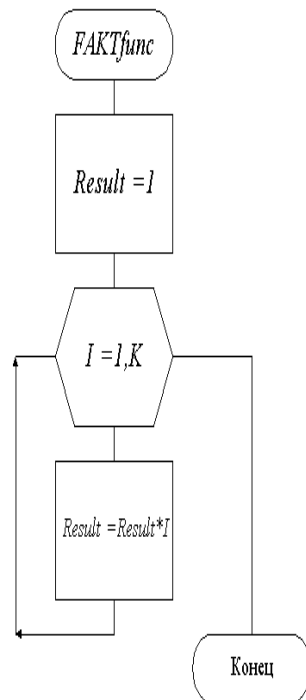


Рисунок 2 – Схема алгоритма функции ФАКТfunc

3. На рисунке 3 – функция **frec**, которая вычисляет $k!$:

- входные данные: k – целочисленная переменная;
- выходные данные: *frec* – имя функции вещественного типа;
- промежуточные данные: i – целочисленная переменная, используемая как счетчик цикла.

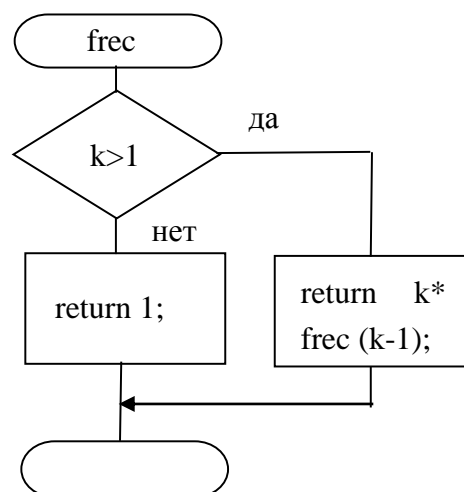


Рисунок 3 – Схема алгоритма функции *frec*

4. Разработка формы – таблица 1, рисунок 4.

Используемые компоненты

Имя компонента	Страница палитры компонент	Настраиваемое свойство	Значение
1. Form1	–	Caption	
2. StaticText1	Additional	Caption	Введите N
3. StaticText2	Additional	Caption	Результат S
4. Edit1	Standard	Text	
5. Edit2	Standard	Text	
6. Edit3	Standard	Text	
7. Edit4	Standard	Text	
8. Edit5	Standard	Text	
9. Button1	Standard	Caption	Вычисление функции

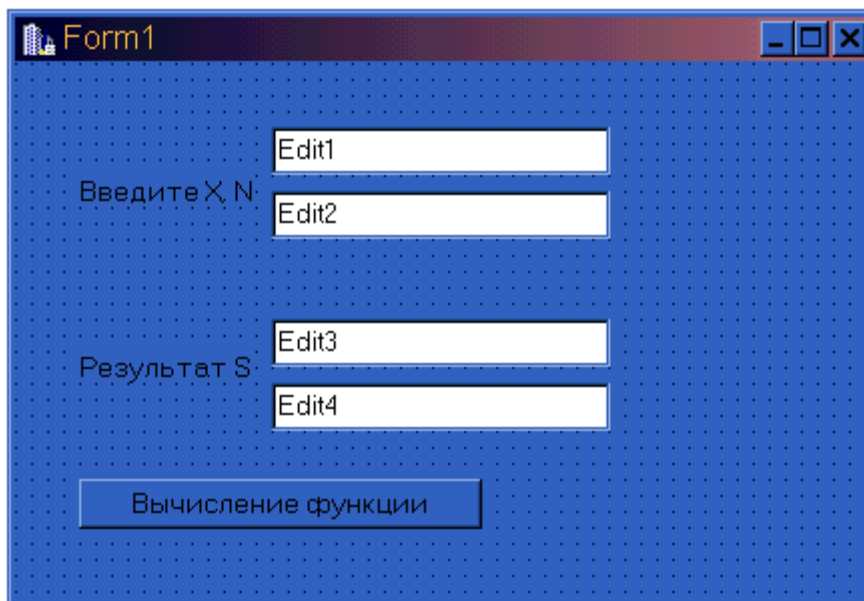


Рисунок 4 – Изображение формы

Текст программы

```
//-----
#include <vcl.h>
#pragma hdrstop
#include<math.h>
#include "bn.h"
```

```

//-----
#pragma resource "*.dfm"
TForm1 *Form1;
unsigned long ФАКТfunc(int k);
unsigned long frec (int k);
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
// вычисление k! с помощью функции ФАКТfunc
unsigned long ФАКТfunc(int k) {
    unsigned long result = 1;
    for (int i=1; i<k; i++)
        result * = i;
    return result;
}
// вычисление k! с помощью рекурсивной функции ФАКТrec
unsigned long frec (int k) {
    if (k>1)
        return k* frec(k-1);
    else
        return 1;
}

void __fastcall TForm1::Button1Click(TObject *Sender) {
    double s=0;
    unsigned long r;
    double x=StrToFloat (Edit1->Text);
    int n=StrToInt (Edit2->Text);
    for(int k = 1; k < n; k++) {
        r = ФАКТfunc(k);
        s+=cos(k*x)/ (double) r ;
    }
    Edit3->Text = FloatToStr (s);
    s =0;
}

```

```

for(int k = 1; k < n; k++){
r = frec (k);
s += cos(k*x)/(double) r;
}
Edit4->Text = FloatToStr(s);
}

```

Индивидуальные задания

Каждому студенту нужно решить две задачи первого и второго уровней сложности.

Задачи первого уровня сложности

1. Опишите функцию целого типа `min_elem(a, n)`, которая находит минимальный элемент в целочисленном одномерном массиве a размером n . С помощью этой функции найдите минимальные элементы в целочисленных массивах b и c , размером nb и nc , соответственно.

2. Опишите функцию целого типа `sum_elem(a, n)`, которая находит сумму чисел, расположенных между максимальным и минимальным числами (в сумму включить и оба эти числа) в целочисленном одномерном массиве a размером n . С помощью этой функции найдите сумму элементов в целочисленном массиве b размером nb .

3. Опишите функцию целого типа `num_elem(a, n)`, которая находит порядковый номер максимального элемента в одномерном массиве вещественных чисел a размером n . С помощью этой функции найдите порядковый номер максимального элемента в массиве вещественных чисел b размером nb .

4. Опишите функцию `num_elem(a, n, nmin, nmax)`, которая находит порядковые номера минимального и максимального элементов в одномерном массиве вещественных чисел a размером n . С помощью этой функции найдите порядковые номера минимального и максимального элементов в массиве вещественных чисел b размером nb .

5. Опишите функцию вещественного типа `sum_elem(a, n)`, которая находит сумму минимального и максимального элементов в одномерном массиве вещественных чисел a размером n . С помощью этой функции найдите сумму минимального и максимального элементов в массиве вещественных чисел b размером nb .

6. Опишите функцию целого типа `min_elem(a, n)`, которая находит минимальный нечетный элемент в целочисленном одномерном массиве a раз-

мером n . С помощью этой функции найдите минимальные нечетные элементы в целочисленных массивах b и c , размером nb и nc , соответственно.

7. Опишите функцию целого типа $\text{min_elem}(a, n)$, которая находит максимальный четный элемент в целочисленном одномерном массиве a размером n . С помощью этой функции найдите максимальные четные элементы в целочисленных массивах b и c , размером nb и nc , соответственно.

8. Опишите функцию вещественного типа $\text{sum_elem}(a, n)$, которая находит среднее арифметическое минимального и максимального элементов в одномерном массиве вещественных чисел a размером n . С помощью этой функции найдите среднее арифметическое минимального и максимального элементов в массиве вещественных чисел b размером nb .

9. Опишите функцию $\text{min_max_elem}(a, n, \text{min}, \text{max})$, которая находит минимальный и максимальный элементы в одномерном массиве вещественных чисел a размером n . С помощью этой функции найдите минимальный и максимальный элементы в массиве вещественных чисел b размером nb .

10. Опишите функцию $\text{min_max_elem}(a, n, \text{min}, \text{max})$, которая находит минимальный элемент среди элементов, имеющих четный индекс, и максимальный элемент среди элементов, имеющих нечетный индекс, в одномерном массиве вещественных чисел a размером n . С помощью этой функции найдите минимальный и максимальный элементы в массиве вещественных чисел b размером nb .

11. Опишите функцию $\text{inv_elem}(a, n)$, которая инвертирует (изменяет порядок следования на обратный) элементы в одномерном массиве вещественных чисел a размером n . С помощью этой функции инвертировать элементы в массиве вещественных чисел b размером nb .

12. Опишите функцию целого типа $\text{sum_elem}(a, n)$, которая находит произведение чисел, расположенных между максимальным и минимальным числами (в произведение включить и оба эти числа) в целочисленном одномерном массиве a размером n . С помощью этой функции найдите произведение элементов в целочисленном массиве b размером nb .

13. Опишите функцию вещественного типа $\text{mul_elem}(a, n)$, которая находит произведение минимального и максимального элементов в одномерном массиве вещественных чисел a размером n . С помощью этой функции найдите произведение минимального и максимального элементов в массиве вещественных чисел b размером nb .

14. Опишите функцию $\text{copy_elem}(a, n, x, r, m)$, которая копирует из одномерного массива вещественных чисел a размером n в массив r m чисел, значения которых больше x . С помощью этой функции копируйте из массива

вещественных чисел b размером nb в результирующий массив числа, значения которых больше x . Предусмотреть ситуацию, когда таких элементов нет.

15. Опишите функцию $\text{copy_elem}(a, n, x, y, r, m)$, которая копирует из одномерного массива целых чисел a размером n в массив r m чисел, значения которых принадлежат интервалу $[x, y]$. С помощью этой функции копируйте из массива целых чисел b размером nb в результирующий массив числа, значения которых принадлежат интервалу $[x, y]$.

16. Опишите функцию целого типа $\text{sum_elem}(a, n)$, которая находит сумму чисел, расположенных от начала массива до максимального в целочисленном одномерном массиве a размером n . С помощью этой функции найдите сумму элементов, расположенных от начала массива до максимального в целочисленном массиве b размером nb .

17. Опишите функцию целого типа $\text{sum_elem}(a, n)$, которая находит сумму чисел, расположенных от максимального элемента до конца массива в целочисленном одномерном массиве a размером n . С помощью этой функции найдите сумму элементов, расположенных от максимального элемента до конца массива в целочисленном массиве b размером nb .

18. Опишите функцию целого типа $\text{mul_elem}(a, n)$, которая находит произведение чисел, расположенных от начала массива до минимального в целочисленном одномерном массиве a размером n . С помощью этой функции найдите произведение элементов, расположенных от начала массива до минимального в целочисленном массиве b размером nb .

19. Опишите функцию целого типа $\text{mul_elem}(a, n)$, которая находит произведение чисел, расположенных от максимального элемента до конца массива в целочисленном одномерном массиве a размером n . С помощью этой функции найдите произведение элементов, расположенных от максимального элемента до конца массива в целочисленном массиве b размером nb .

20. Опишите функцию вещественного типа $\text{sum_elem}(a, n)$, которая находит среднее арифметическое чисел, расположенных от максимального элемента до конца массива в целочисленном одномерном массиве a размером n . С помощью этой функции найдите среднее арифметическое элементов, расположенных от максимального элемента до конца массива в целочисленном массиве b размером nb .

21. Опишите функцию $\text{sort_elem}(a, n)$, которая выполняет сортировку по возрастанию значений элементов в одномерном массиве вещественных чисел a размером n . С помощью этой функции выполните сортировку по возрастанию значений элементов в массиве вещественных чисел b размером nb .

22. Опишите функцию `sort_elem(a, n)`, которая выполняет сортировку по убыванию значений элементов в одномерном массиве вещественных чисел a размером n . С помощью этой функции выполните сортировку по убыванию значений элементов в массиве вещественных чисел b размером nb .

23. Опишите функцию `sort_elem(a, n)`, которая выполняет сортировку по возрастанию значений элементов от начала массива до минимального элемента в одномерном массиве вещественных чисел a размером n . С помощью этой функции выполните сортировку по возрастанию значений элементов от начала массива до минимального элемента в массиве вещественных чисел b размером nb .

24. Опишите функцию `sort_elem(a, n)`, которая выполняет сортировку по убыванию значений элементов от максимального элемента до конца массива в одномерном массиве вещественных чисел a размером n . С помощью этой функции выполните сортировку по убыванию значений элементов от максимального элемента до конца массива в массиве вещественных чисел b размером nb .

25. Опишите функцию `sort_elem(a, n)`, которая выполняет сортировку по возрастанию значений элементов, расположенных между максимальным и минимальным элементами (включить и оба эти числа) в одномерном массиве вещественных чисел a размером n . С помощью этой функции выполните сортировку по возрастанию значений элементов, расположенных между максимальным и минимальным элементами в массиве вещественных чисел b размером nb .

26. Опишите функцию `sort_elem(a, n)`, которая выполняет сортировку по убыванию значений элементов, расположенных между максимальным и минимальным элементами (включить и оба эти числа) в массиве вещественных чисел a размером n . С помощью этой функции выполните сортировку по убыванию значений элементов, расположенных между максимальным и минимальным элементами в массиве вещественных чисел b размером nb .

27. Опишите логическую функцию `prov_elem(a, n)`, которая выполняет проверку отсортирован ли по возрастанию значений элементов одномерный массив вещественных чисел a размером n . Если массив отсортирован, то возвращается значение *true*, иначе *false*. С помощью этой функции выполните проверку массива вещественных чисел b размером nb .

28. Опишите логическую функцию `prov_elem(a, n)`, которая выполняет проверку отсортирован ли по убыванию значений элементов одномерный массив вещественных чисел a размером n . Если массив отсортирован, то

возвращается значение *true*, иначе – *false*. С помощью этой функции выполните проверку одномерного массива вещественных чисел *b* размером *nb*.

29. Опишите логическую функцию `prov_elem(a, n)`, которая выполняет проверку наличия одинаковых элементов в одномерном массиве целых чисел *a* размером *n*. Если одинаковые элементы встречаются, то возвращается значение *true*, иначе – *false*. С помощью этой функции выполните проверку массива целых чисел *b* размером *nb*.

30. Опишите функцию целого типа `pros_elem(a, n)`, которая выполняет подсчет количества простых чисел в одномерном массиве целых чисел *a* размером *n*. С помощью этой функции подсчитайте количество простых чисел в массиве целых чисел *b* размером *nb*.

Задачи второго уровня сложности

1. Составьте программу вычисления функции *y* и суммы *S*, представляющей собой формулу разложения заданной функции *y* в ряд.

Варианты заданий приведены ниже (табл. 2).

В первой графе содержится порядковый номер задания. Рекомендуется выбирать номер задания, соответствующий порядковому номеру фамилии студента в списке группы.

Во второй графе приводится формула функции *y*.

В третьей графе помещается формула разложения функции *y* в ряд.

В четвертой графе показан диапазон значения аргумента *x*, для которого следует выполнить вычисления. Рекомендуется вычислить *y* и *S* для 10 точек заданного диапазона изменения *x*.

Шаг изменения аргумента *x* определить по формуле

$$H = \frac{x_k - x_n}{10},$$

где x_n – начальное значение *x*;

x_k – конечное значение *x*.

В пятой графе указано значение *n* – количество членов суммы *S*.

При составлении программы необходимо использовать функции, основанные на рекурсивном методе вычислений.

Варианты заданий

№	Функция Y	Сумма S	Диапазон изменения аргумента	n
1	$Y = 3^x$	$S = 1 + \frac{\ln 3}{1!}x + \frac{\ln^2 3}{2!}x^2 + \dots + \frac{\ln^n 3}{n!}x^n$	$0.1 \leq x \leq 1$	10
2	$Y = -\ln \left 2 \sin \frac{x}{2} \right $	$S = \cos x + \frac{\cos 2x}{2} + \dots + \frac{\cos nx}{n}$	$\frac{\pi}{5} \leq x \leq \frac{9\pi}{5}$	40
3	$Y = \sin x$	$S = x - \frac{x^3}{3!} + \dots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$	$0.1 \leq x \leq 1$	10
4	$Y = e^x$	$S = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots + \frac{x^n}{n!}$	$1 \leq x \leq 2$	15
5	$Y = e^{x \cos \frac{\pi}{4}} \times \cos(x \sin \frac{\pi}{4})$	$S = 1 + \frac{\cos \pi \cos}{1!}x + \dots + \frac{\cos^n \pi \cos}{n!}x^n$	$0.1 \leq x \leq 1$	25
6	$Y = \cos x$	$S = 1 - \frac{x^2}{2!} + \dots + (-1)^n \frac{x^{2n}}{(2n)!}$	$0.1 \leq x \leq 1$	10
7	$Y = \frac{1}{4} \ln \frac{1+x}{1-x} + \frac{1}{2} \arctan g(x)$	$S = x + \frac{x^5}{5} + \dots + \frac{x^{4n+1}}{4n+1}$	$0.1 \leq x \leq 0.8$	40
8	$Y = e^{\cos x} \cos(\sin x)$	$S = 1 + \frac{\cos x}{1!} + \dots + \frac{\cos^n x}{n!}$	$0.1 \leq x \leq 1$	20
9	$Y = (1 + 2x^2)e^{x^2}$	$S = 1 + 3x^2 + \dots + \frac{2n+1}{n!}x^{2n}$	$0.1 \leq x \leq 1$	10
10	$Y = \frac{1}{2} \ln x$	$S = \frac{x-1}{x+1} + \frac{1}{3} \left(\frac{x-1}{x+1} \right)^3 + \dots + \frac{1}{2n+1} \left(\frac{x-1}{x+1} \right)^{2n+1}$	$0.2 \leq x \leq 1$	10
11	$Y = \frac{1}{4} \left(x^2 - \frac{\pi^2}{3} \right)$	$S = -\cos x + \frac{\cos 2x}{2^2} + \dots + (-1)^n \frac{\cos nx}{n^2}$	$\pi/5 \leq x \leq \pi$	20
12	$Y = \frac{e^x + e^{-x}}{2}$	$S = 1 + \frac{x^2}{2!} + \dots + \frac{x^{2n}}{(2n)!}$	$0.1 \leq x \leq 1$	10

№	Функция Y	Сумма S	Диапазон изменения аргумента	n
13	$Y = e^{2x}$	$S = 1 + \frac{2x}{1!} + \dots + \frac{(2x)^n}{n!}$	$0.1 \leq x \leq 1$	20
14	$Y = \left(\frac{x^2}{4} + \frac{x}{2} + 1\right)e^{\frac{x}{2}}$	$S = 1 + 2\frac{x}{2} + \dots + \frac{n^2 + 1}{n!} \cdot \frac{x^n}{2^n}$	$0.1 \leq x \leq 1$	30
15	$Y = \operatorname{arctg}(x)$	$S = x - \frac{x^3}{3} + \dots + (-1)^n \frac{x^{2n+1}}{2n+1}$	$0.1 \leq x \leq 0.5$	40
16	$Y = 7^x$	$S = 1 + \frac{\ln 7}{1!} x + \frac{\ln^2 7}{2!} x^2 + \dots + \frac{\ln^n 7}{n!} x^n$	$0.1 \leq x \leq 1$	15
17	$Y = \left(1 - \frac{x^2}{2}\right) \cos x - \frac{x}{2} \sin x$	$S = 1 - \frac{3}{2} x^2 + \dots + (-1)^n \frac{2n^2 + 1}{(2n)!} x^{2n}$	$0.1 \leq x \leq 1$	35
18	$Y = 2(\cos^2 x - 1)$	$S = -\frac{(2x)^2}{2} + \frac{(2x)^4}{24} + \dots + (-1)^n \frac{(2x)^{2n}}{(2n)!}$	$0.1 \leq x \leq 1$	15
19	$Y = \ln \frac{1}{2 + 2x + x^2}$	$S = -(1+x)^2 + \frac{(1+x)^4}{2} + \dots + (-1)^n \frac{(1+x)^{2n}}{n}$	$-2 \leq x \leq -0.1$	40
20	$Y = \frac{e^x - e^{-x}}{2}$	$S = x + \frac{x^3}{3!} + \dots + \frac{x^{2n+1}}{(2n+1)!}$	$0.1 \leq x \leq 1$	20
21	$Y = \frac{x(3-x)}{(1-x)^3}$	$S = 3x + 8x^2 + \dots + n(n+2)x^n$	$0.1 \leq x \leq 0.8$	40
22	$Y = 5^x$	$S = 1 + \frac{\ln 5}{1!} x + \frac{\ln^2 5}{2!} x^2 + \dots + \frac{\ln^n 5}{n!} x^n$	$0.1 \leq x \leq 1$	15
23	$Y = \frac{1+x^2}{2} \operatorname{arctg} x - \frac{x}{2}$	$S = \frac{x^3}{3} - \frac{x^5}{15} + \dots + (-1)^{n+1} \frac{x^{2n+1}}{4n^2 - 1}$	$0.1 \leq x \leq 1$	30
24	$Y = \frac{x}{2}$	$S = \sin x - \frac{\sin 2x}{2} + \dots + (-1)^{n+1} \frac{\sin nx}{n}$	$\frac{\pi}{5} \leq x \leq \frac{4\pi}{5}$	40

№	Функция Y	Сумма S	Диапазон изменения аргумента	n
25	$Y = \frac{x \sin \frac{\pi}{4}}{1 - 2x \cos \frac{\pi}{4} + x^2}$	$S = x \sin \frac{\pi}{4} + x^2 \sin 2 \frac{\pi}{4} + \dots + x^n \sin n \frac{\pi}{4}$	$0.1 \leq x \leq 0.8$	40
26	$Y = -\frac{1}{2} \ln (1 - 2x \cos \frac{\pi}{3} + x^2)$	$S = x \cos \frac{\pi}{3} + \frac{x^2 \cos 2 \frac{\pi}{3}}{2} + \dots + \frac{x^n \cos n \frac{\pi}{3}}{n}$	$0.1 \leq x \leq 0.8$	35
27	$Y = \frac{1}{2} - \frac{\pi}{4} \sin x $	$S = \frac{\cos 2x}{3} + \frac{\cos 4x}{15} + \dots + \frac{\cos 2nx}{4n^2 - 1}$	$0.1 \leq x \leq 0.8$	50
28	$Y = \frac{\pi}{4}$	$S = \sin x + \frac{\sin 3x}{3} + \dots + \frac{\sin (2n - 1) x}{2n - 1}$	$\frac{\pi}{10} \leq x \leq \frac{9\pi}{10}$	40
29	$Y = \frac{\pi^2}{8} - \frac{\pi}{4} x $	$S = \cos x + \frac{\cos 3x}{3^2} + \dots + \frac{\cos (2n - 1)x}{(2n - 1)^2}$	$\frac{\pi}{5} \leq x \leq \pi$	40
30	$Y = -\ln \sin x $	$S = \ln 2 + \cos 2x + \frac{\cos 4x}{2} + \dots + \frac{\cos 2nx}{n}$	$\frac{\pi}{6} \leq x \leq \frac{\pi}{2}$	40

Контрольные вопросы к защите лабораторной работы

1. Для чего предназначаются подпрограммы?
2. Что включает в себя заголовок функции?
3. Чем отличаются формальные и фактические параметры?
4. Для чего нужен оператор **return**?
5. Как можно передавать параметры в функции?
6. Как осуществляется передача параметров по умолчанию?
7. Для чего предназначена рекурсия?
8. Какие разновидности организации рекурсии известны?

Содержание отчета

Отчет по лабораторной работе включает:

- титульный лист;
- условие задания;
- алгоритм решения задачи;
- текст программы;
- результаты тестирования программы.

Список используемых источников

1. Белов В.Г. Основы программирования на языке C++ Builder [Текст]: учеб. пособие / В.Г. Белов, Т.М. Белова; Юго-Зап. гос. ун-т. – Курск, 2015. – 160 с.
2. Белов В.Г. Основы программирования на языке C++ Builder [Электронный ресурс]: учеб. пособие / В.Г. Белов, Т.М. Белова; Юго-Зап. гос. ун-т. – Курск, 2015. – 160 с.
3. Архангельский, А.Я. Программирование в C++ Builder [Текст] / А.Я. Архангельский. – М.: Изд-во БИНОМ, 2010. – 1304 с.
4. Дэвид Р. Мюссер. C++ и STL. Справочное руководство [Текст] / Дэвид Р. Мюссер, Жилмер Дж. Дердж, Атул Сейни. – М.: Вильямс, 2010. – 432 с.
5. Культин, Н. C++ Builder [Текст] / Н. Культин. – СПб.: БХВ-Петербург, 2012. – 464 с.
6. Лафоре, Р. Объектно-ориентированное программирование в C++ [Текст] / Р. Лафоре. – СПб.: ПИТЕР, 2013. – 924 с.
7. Прата, С. Язык программирования C++. Лекции и упражнения [Текст] / С. Прата. – М.: Вильямс, 2012. – 1244 с.
8. Липпман Стенли Б. Язык программирования C++. Базовый курс [Текст] / Стенли Б. Липпман, Жози Лажойе, Барбара Э. Му. – М.: Вильямс, 2014. – 1120 с.
9. Страуструп, Б. Программирование. Принципы и практика использования C++ [Текст] / Б. Страуструп. – М.: Вильямс, 2011. – 1206 с.