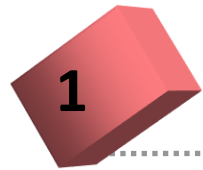




СОЗДАНИЕ ПРОГРАММ НА СИ++

**Дальневосточный государственный университет путей сообщения,
кафедра «Вычислительная техника и компьютерная графика»,
к.т.н., доцент Белозеров Олег Иванович.**

Вопросы лекции:



Базовые принципы программирования



Оценка сложности алгоритма



Решение типовых задач

1. Базовые принципы программирования.

Стиль написания.

Как лучше писать код?

Как писать хороший код?

Почему это важно?

Рассмотрим пример

```
int a[100];
```

Что плохо в приведенном примере?

- Имя массива.
- Использование чисел.
 - Тип массива.

Как улучшить?

Говорящие имена.

Имя объясняет назначение объекта. Сокращения не объясняют. Малая длина имени. Меньше транслитерации. Лучше использовать национальный алфавит.

Использовать константы.

Возможно использовать глобальные константы. Использовать константы с типами (типизированные).

Грамотно проектировать архитектуру.

Пример

```
const int size = 100;
```

```
int numbers[size];
```

```
for(int i = 0; i < size; ++i){
```

```
...
```

```
}
```

Проблема начального значения:
использование переменной до
присваивания ей значения.

- Хороший способ получить
неявную ошибку, исправить
которую сложно.
- Надо знать, как язык и среда
действуют.

Вопросы:

- Разве нужно оформлять код?
- Какое оформление кода следует считать правильным?

Ответы:

- Да, в оформлении есть свой смысл.
- Существуют разные стили оформления кода. Разница во вкусе.

Для имен распространены два способа: верблюжий и змеиный.

- При верблюжьем способе **имяПеременной** каждое следующее слово пишут с заглавной.
- При змеином способе слова в **имени_переменной** разделяют символом подчеркивания.

Как ставить скобки?

- Принято ставить открывающую скобку на линии оператора, под оператором на одном уровне или с отступом.
- Закрывающую скобку ставят на уровне оператора (в первом случае) или открывающей скобки.

Когда их ставить?

- Ставить лучше всегда.

```
if(a == b) {
```

```
...
```

```
}
```

```
if(a == b)
```

```
{
```

```
...
```

```
}
```

```
if(a == b)
{
    ...
}
```

```
if(a == b)
{
    ...
}
```

Для чего используем отступы?

- Для выделения программных структур и вложенности кода.
- Сколько пробелов ставить? 2, 3, 4, 6, 8.
- Раньше широко применялось 8 пробелов.
- Сейчас чаще используют 4.
- Удобно отделять логические блоки пустой строкой.

Комментарии

- Пояснения к исходному тексту программы, находящиеся непосредственно внутри комментируемого кода.
- Синтаксис комментариев определяется языком программирования.
- С точки зрения компилятора или интерпретатора, комментарии — часть текста программы, не влияющая на её семантику.
- Комментарии не оказывают никакого влияния на результат компиляции программы или её интерпретацию.

Виды комментариев:

- Однострочные и многострочные комментарии
- Аннотации
- Автоматическая генерация документации

Однострочные и многострочные комментарии

- Многострочный комментарий может иметь любую длину, он отмечается специальными символами в начале и конце (например, `/* */`). Некоторые языки позволяют вложение многострочных комментариев, другие — нет.
- Однострочный комментарий отмечается специальным символом в начале (например, `//`) и продолжается до конца строки. Обычно допускается вложение однострочных комментариев в другие, как одно- так и многострочные комментарии. Способы записи можно чередовать, с точки зрения семантики они одинаковы.

Аннотации

- Применяется в набросках доказательств правильности программ. Такие комментарии описывают состояние компьютера, когда программа в процессе выполнения достигнет точки, где расположен комментарий.
- Программа, снабжённая комментариями-аннотациями, называется аннотированной программой.

Автоматическая генерация документации

- Специальным образом оформленные комментарии (т.н. документирующие комментарии) используются для автоматического создания документации, в первую очередь, к библиотекам функций или классов. Для этого используются генераторы документации, например, такие как doxygen для Си и Си++ и др.
- Документирующие комментарии как правило оформляются как многострочные комментарии в стиле языка Си. В каждом случае комментарий должен находиться перед документируемым элементом. Первым символом в комментарии (и в начале строк комментария) должен быть *. Блоки разделяются пустыми строками.

Пример

```
/**
```

```
**  Имя или краткое описание объекта
```

```
**
```

```
**  Развернутое описание
```

```
**
```

```
**  @имя_дескриптора значение
```

```
**  @return тип_данных
```

```
**/
```

Пример

```
/**
```

```
**  Имя или краткое описание объекта
```

```
**
```

```
**  Развернутое описание
```

```
**
```

```
**  @имя_дескриптора значение
```

```
**  @return тип_данных
```

```
**/
```

Назначение

- Комментарии должны объяснять намерения программиста, а не код
- То, что можно выразить на языке программирования, не должно выноситься в комментарии
- Надо использовать говорящие названия переменных, функций, классов, методов и пр.
- Разбивать программу на лёгкие для понимания части

Назначение

- Есть мнение (его придерживаются в экстремальном программировании и некоторых других гибких методологиях программирования), что если для понимания программы требуются комментарии — значит, она плохо написана.
- Концепция грамотного программирования настаивает на включение в текст программы настолько подробных и продуманных комментариев, чтобы она стала исходным текстом не только для исполняемого кода, но и для сопроводительной документации.

Советы

- Приоритетная задача каждого программиста — это актуальные комментарии.
- Пишите код так, как будто сопровождать его будет человек с низкими интеллектуальными способностями.
- Комментируем код именно для себя (грамотно, чтобы и другие поняли).
- Не нужно комментировать очевидные вещи.

- Комментарий должен дополнять код, а не перефразировать его.
- Если код использует стандартные функции, конструкции или классы описанные в азах языка, то не тратьте свое время, в данном случае поможет только правильно оформленный код.
- Комментарий к фрагменту кода, нужно писать с тем же отступом, что и у комментируемого кода.
- Вредная привычка многих — разговаривать с кем-то в комментарии, комментарий должен быть коротким и четким.

- В комментарии можно оставить на потом задачу, которую по каким-то причинам не выполнили в настоящий момент. Такие заметки хранить в отдельном файле не советую, потому как, кроме Вас над кодом может работать еще кто-то и проще, если он будет на месте видеть, где и что не доделано, но будет реализовано. Возможно будут найдены более актуальные решения.
- Хорошая практика — в начале файла кратко описать его назначение.
- Новые классы и функции — желательно описать, какие и откуда попадают входящие данные и какая цель.
- Чтобы избавиться от лишних комментариев кода, можно выбирать название классов, функций, объектов и т.п. по назначению.

2. Оценка сложности алгоритма

Наверняка вы не раз сталкивались с обозначениями вроде **$O(\log n)$** или слышали фразы типа «логарифмическая вычислительная сложность» в адрес каких-либо алгоритмов.

Есть задача: сравнить два
алгоритма и оценить
эффективность.

Сложность алгоритмов обычно оценивают по времени выполнения или по используемой памяти.

Функции

$T(N)$ – время работы;

$M(N)$ – объем памяти.

В обоих случаях сложность зависит от размеров входных данных: массив из 100 элементов будет обработан быстрее, чем аналогичный из 1000. При этом точное время мало кого интересует: оно зависит от процессора, типа данных, языка программирования и множества других параметров. Важна лишь асимптотическая сложность, т. е. сложность при стремлении размера входных данных к бесконечности.

Допустим, некоторому алгоритму нужно выполнить $4n^3 + 7n$ условных операций, чтобы обработать n элементов входных данных. При увеличении n на итоговое время работы будет значительно больше влиять возведение n в куб, чем умножение его на 4 или же прибавление $7n$. Тогда говорят, что временная сложность этого алгоритма равна $O(n^3)$, т.е. зависит от размера входных данных кубически.

Использование заглавной буквы O (или так называемая O-нотация) пришло из математики, где её применяют для сравнения асимптотического поведения функций.

Формально $O(f(n))$ означает, что время работы алгоритма (или объём занимаемой памяти) растёт в зависимости от объёма входных данных не быстрее, чем некоторая константа, умноженная на $f(n)$.

Примеры

$O(n)$ — линейная сложность

Такой сложностью обладает, например, алгоритм поиска наибольшего элемента в не отсортированном массиве. Нам придётся пройти по всем n элементам массива, чтобы понять, какой из них максимальный.

$O(\log n)$ — логарифмическая сложность

Простейший пример — бинарный поиск. Если массив отсортирован, мы можем проверить, есть ли в нём какое-то конкретное значение, методом деления пополам. Проверим средний элемент, если он больше искомого, то отбросим вторую половину массива — там его точно нет. Если же меньше, то наоборот — отбросим начальную половину. И так будем продолжать делить пополам, в итоге проверим $\log n$ элементов.

$O(n^2)$ — квадратичная сложность

Такую сложность имеет, например, алгоритм сортировки вставками. В канонической реализации он представляет из себя два вложенных цикла: один, чтобы проходить по всему массиву, а второй, чтобы находить место очередному элементу в уже отсортированной части. Таким образом, количество операций будет зависеть от размера массива как $n * n$, т.е. n^2 .

Бывают и другие оценки по сложности, но все они основаны на том же принципе.

Также случается, что время работы алгоритма вообще не зависит от размера входных данных. Тогда сложность обозначают как $O(1)$. Например, для определения значения третьего элемента массива не нужно ни запоминать элементы, ни проходить по ним сколько-то раз. Всегда нужно просто дождаться в потоке входных данных третий элемент и это будет результатом, на вычисление которого для любого количества данных нужно одно и то же время.

Аналогично проводят оценку и по памяти, когда это важно. Однако алгоритмы могут использовать значительно больше памяти при увеличении размера входных данных, чем другие, но зато работать быстрее. И наоборот. Это помогает выбирать оптимальные пути решения задач исходя из текущих условий и требований.

Наглядно

Время выполнения алгоритма с определённой сложностью в зависимости от размера входных данных при скорости 10^6 операций в секунду.

Можно оценить сложность основных алгоритмов сортировки и работы с данными (смотрим таблицу).

размер сложность	10	20	30	40	50	60
n	0,00001 сек.	0,00002 сек.	0,00003 сек.	0,00004 сек.	0,00005 сек.	0,00005 сек.
n^2	0,0001 сек.	0,0004 сек.	0,0009 сек.	0,0016 сек.	0,0025 сек.	0,0036 сек.
n^3	0,001 сек.	0,008 сек.	0,027 сек.	0,064 сек.	0,125 сек.	0,216 сек.
n^5	0,1 сек.	3,2 сек.	24,3 сек.	1,7 минут	5,2 минут	13 минут
2^n	0,0001 сек.	1 сек.	17,9 минут	12,7 дней	35,7 веков	366 веков
3^n	0,059 сек.	58 минут	6,5 лет	3855 веков	2×10^8 веков	$1,3 \times 10^{13}$ веков

3. Решение типовых задач

```
#include <iostream>
#include <conio.h> // подключаем getch
using namespace std;
int main()
{
    setlocale(LC_ALL, "Russian");
    int var1; // описание переменной var1
    int var2; // описание переменной var2
    var1 = 20; // присвоение значения переменной var1
    var2 = var1 + 10; // присвоение значения переменной var2
    cout << "var1+10 равно "; // вывод строки
    cout << var2 << endl; // вывод значения var2
    getch();
    return 0;
}
```

 C:\Users\lenovo\Desktop\zzz.exe

var1+10 равно 30

—

```
#include <iostream>

#include <conio.h> // подключаем getch

using namespace std;

int main(int argc, char* argv[])
{
    setlocale(LC_ALL, "Russian");

    float a = 0, b = 0;

    cout << "Введите a \n" ;
    cin >> a;

    cout << "Введите b \n" ;
    cin >> b;

    if (a > b)
        cout << a << "-Max " << b << "-Min\n" ;
    else
        cout << b << "-Max " << a << "-Min\n" ;

    getch(); //пауза

    return 0;
}
```

Введите a

3

Введите b

5

5-Max 3-Min

—

```
#include <iostream>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
using namespace std;
int main(int argc, char* argv[])
{
    setlocale(LC_ALL, "Russian");
    float a = 0, b = 0;
    cout << «Введите a \n" ;
    cin >> a;
    cout << «Введите b \n" ;
    cin >> b;
    cout << «Максимальное число - " << max(a,b) ;
    getch();
    return 0;
}
```



C:\Users\lenovo\Desktop\111.exe

Введите a

10

Введите b

2

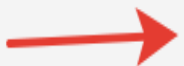
Максимальное число - 10

Функция **getch** нужна для чтения одного символа с клавиатуры, этот символ на экран не выводится. Чаще всего данная функция используется чтобы не дать консоли преждевременно закрыться. Эта функция ожидает пока пользователь введёт символ и лишь затем закрывает консоль. Чтобы воспользоваться функцией getch, необходимо подключить заголовочный файл: `#include <conio.h>.`

Функция **getch** является альтернативной формой команды **getchar**. Главной отличительной чертой является то, что вызывая **getchar**, консоль ожидает подтверждения ввода нажатием Enter. В случае с **getch** выход происходит сразу же, как только нажата какая-то клавиша. Возвращается код этой клавиши.

```
#include <iostream>
#include <conio.h>
using namespace std;
int main(int argc, char* argv[])
{
    setlocale(LC_ALL, "Russian");
    int x = getch();
    printf("%d", x); // %d – десятичные целые
    getch();
}
```

ASCII (American standard code for information interchange — название таблицы (кодировки, набора), в которой некоторым распространённым печатным и непечатным символам сопоставлены числовые коды. Таблица была разработана и стандартизирована в США, в 1963 году.

81	Q прописная буква q	82	R прописная буква r	83	S прописная буква s
84	T прописная буква t	85	U прописная буква u	86	V прописная буква v
87	W прописная буква w	88	X прописная буква x	89	Y прописная буква y
90	Z прописная буква z	91	[левая квадратная скобка	92	\ обратная косая черта
93] правая квадратная скобка	94	^ знак вставки (диакритический знак циркумфлекс)	95	_ символ подчеркивания
96	` гравис (диакритический знак)	97 	a строчная буква a	98	b строчная буква b
99	c строчная буква c	100	d строчная буква d	101	e строчная буква e
102	f строчная буква f	103	g строчная буква g	104	h строчная буква h
105	i строчная буква i	106	j строчная буква j	107	k строчная буква k
108	l строчная буква l	109	m строчная буква m	110	n строчная буква n
111	o строчная буква o	112	r строчная буква r	113	q строчная буква q

При необходимости вывода управляющих символов (% \ и т.п.) их нужно указать 2 раза, например:

```
#include <iostream>
```

```
#include <conio.h>
```

```
using namespace std;
```

```
int main(int argc, char* argv[])
```

```
{
```

```
    setlocale(LC_ALL, "Russian");
```

```
    printf("Только %d%% предприятий не работало. \n",5);
```

```
    getch();
```

```
}
```

получим: *Только 5% предприятий не работало.*

 C:\Users\lenovo\Desktop\444.exe

Только 5% предприятий не работало.

■

Управляющие символы: $\backslash n$ – переход на новую строку;

$\backslash t$ – горизонтальная табуляция; $\backslash r$ – возврат в начало

строки; $\backslash a$ – звуковой сигнал; $\backslash ?$ – знак вопроса.

Приведем основные форматы печати:

- $\%d$ – десятичные целые (*int*);
- $\%s$ – строка символов (*string*);
- $\%ld$ – длинное целое;
- $\%x$ – шестнадцатеричные данные;
- $\%c$ – один символ (*char*);
- $\%f$ – данные типа *float*;
- $\%lf$ – данные типа *double*;
- $\%o$ – восьмеричные данные.

Лабораторная работа 1

Вариант №20

1. Пассажирский поезд, двигаясь равномерно, за 10 мин прошел путь 15 км. Вычислить скорость поезда (в м/с и км/ч)
2. Сторона ромба равна a , меньшая диагональ L . Найти большую диагональ.

$$a=5, L=6, m=8.$$

3. Проверить истинность высказывания: "Данное число является трехзначным четным числом".

```
#include <cstdlib> // # - указывает на наличие директивы
#include <iostream>
#include <locale>
#include <cmath>
using namespace std; // std – стандартное пространство имен
int main(int argc, char *argv[]) // main – функция, argc – содержит в себе
количество аргументов командной строки, argv – указатель на массив строк
{
    setlocale(LC_ALL, "Russian");
    float V=0, t=0, S=0;
    cout<< "Введите t(мин)"<<endl;
    cin>>t;
    cout<< "Введите S(км)"<<endl;
    cin>>S;
    t/=60;
    V=S/t;
    cout<<"V="<<V<<"км/ч"<<"="<<V/3.6<<"м/с"<<endl; // для перевода в м/с
    нужно умножить на 1000 метров и разделить на 3600 секунд
    return 0; // инструкция обеспечивает механизм завершения работы. Говорит
    об успешном выполнении функции main
}
```

```
int main(int argc, char *argv[])
{
    setlocale(LC_ALL, "Russian");

    float a=0, L=0, m=0;

    cout<< "Введите сторону ромба a"<<endl;
    cin>>a;

    cout<< "Введите меньшую диагональ L"<<endl;
    cin>>L;

    m=sqrt(4*pow(a,2)-pow(L,2));

    cout<<"Большая диагональ m="<<m<<endl;

    return 0;
}
```

```
int main(int argc, char *argv[])
{
    setlocale(LC_ALL, "Russian");
    int n=0;
    bool t=true;
    cout<< "Введите проверяемое число"<<endl;
    cin>>n;
    if (n%2==0 && n>99 && n<1000) // n%2 - остаток от деления на 2, == логическая операция сравнения,
    && - и, || - или
    {
        t=true;
    }
    else
    {
        t=false;
    }
    if (t) // if(t)=true
    {
        cout<<"Это трехзначное четное число"<<endl;
    }
    else
    {
        cout<<"Это число не подходит под условия"<<endl;
    }
    return 0;
}
```

Лабораторная работа 2

Задание №1.

Запрограммировать диалог пользователя с машиной на тему по варианту. Предусмотреть контроль ввода и выдачу сообщения при отказе.

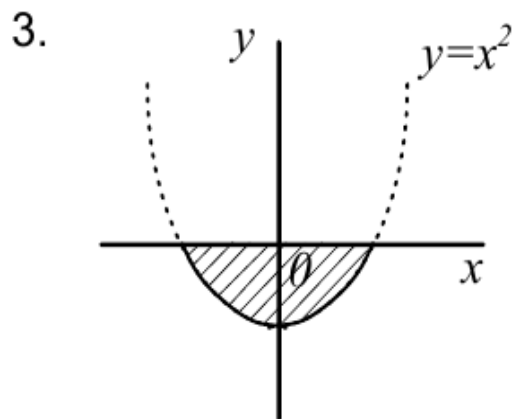
Задание №2.

Составить программу решения задачи.

Задание №3.

Дана ограниченная область и точка $A(x_0, y_0)$. Написать программу, которая проверяет, попадает ли точка с координатами пользователя в заданную область.

1. Составить программу для определения подходящего возраста для вступления в брак, используя следующее правило: возраст девушки равен половине возраста мужчины плюс 7, возраст мужчины определяется как удвоенный возраст девушки минус 14.
2. Голодная зима. Суточный рацион коровы составляет u кг сена, v кг силоса и w кг комбикорма. В хозяйстве, содержащем стадо из k голов, осталось s центнеров сена, t тонн силоса и f мешков комбикорма по 50 кг. Сколько еще дней хозяйство сможет кормить коров по полному рациону? Какой из кормов кончится раньше других?



```
    for ( ;pol!='m' && pol!='w'; )
    {
        cout << "Нужно ввести только одну букву (m или w): " << endl;
        cin >> pol;
    }
    cout << "Введите Ваш возраст (полных лет) ";
    cin >> a;

    if (pol=='m')
    {
        cout << "Подходящий возраст для вступления в брак для мужчины равен:
" << a*2-14 << endl;
    }
    else
    {
        cout << " Подходящий возраст для вступления в брак для женщины
равен: " << a/2+7 << endl;
    }
    return 0;
}
```

```

int t1=0,t2=0,t3=0;
int u=0,v=0,w=0;
int s=0,t=0,f=0;
int min=0;
cout << "Введите суточный рацион сена на 1
корову, кг ";
cin >> u;
cout << "Введите суточный рацион силоса на
1 корову, кг ";
cin >> v;
cout << "Введите суточный рацион
комбикорма на 1 корову, кг ";
cin >> w;
cout << "Сколько осталось сена, центнеров ";
cin >> s;
cout << "Сколько осталось силоса, тонн ";
cin >> t;
cout << "Сколько осталось комбикорма,
мешков ";
cin >> f;

```

```

s=s*100;
t=t*1000;
f=f*50;
t1=s/u;
t2=t/v;
t3=f/w;
cout << "Сена хватит на " << t1 << " дней"<< endl;
cout << "Силоса хватит на " << t2 << " дней"<< endl;
cout << "Комбикорма хватит на " << t3 << " дней"<<
endl;
if (t1<t2){
    min=t1;
}
else {
    min=t2;
}
if (min<t3) {
    min=min;
}
else
{
    min=t3;
}
cout << "Корма хватит на " << min << "
полных дней"<< endl;

return 0;
}

```



```
6 using namespace std;
7
8 int main(int argc, char *argv[])
9 {
10     setlocale (LC_ALL, "Russian");
11     int t1=0,t2=0,t3=0;
12     float u=0,v=0,w=0;
13     float s=0,t=0,f=0;
14     int k=0, min=0;
15
16     cout << "Введите количество коров, шт. ";
17     cin >> k;
18
19     cout << "Введите суточный рацион сена на 1 корову, кг ";
20     cin >> u;
21     u*=k;
22     cout << "Введите суточный рацион силоса на 1 корову, кг ";
23     cin >> v;
24     v*=k;
25     cout << "Введите суточный рацион комбикорма на 1 корову, кг ";
26     cin >> w;
27     w*=k;
28
29     cout << "Сколько осталось сена, центнеров ";
30     cin >> s;
31     cout << "Сколько осталось силоса, тонн ";
32     cin >> t;
33     cout << "Сколько осталось комбикорма, мешков ";
34     cin >> f;
35
36     s=s*100;
37     t=t*1000;
38     f=f*50;
```

```
63 }  
64 cout << "Корма хватит на " << min << " полных дней"<< endl;  
65  
66 if (t1==min && t1!=t2 && t1!=t3)  
67 {  
68     cout << "Первым закончится сено"<< endl;  
69 }  
70  
71     else if (t2==min && t2!=t3 && t2!=t1)  
72     {  
73         cout << "Первым закончится силос"<< endl;  
74     }  
75     else if (t3==min && t3!=t1 && t3!=t2)  
76     {  
77         cout << "Первым закончится комбикорм"<< endl;  
78     }  
79     else if (t1==min && t1==t2 && t1!=t3)  
80     {  
81         cout << "Первыми закончатся сено и силос"<< endl;  
82     }  
83     else if (t1==min && t1==t3 && t1!=t2)  
84     {  
85         cout << "Первыми закончатся сено и комбикорм"<< endl;  
86     }  
87     else if (t2==min && t2==t3 && t2!=t1)  
88     {  
89         cout << "Первыми закончатся комбикорм и силос"<< endl;  
90     }  
91     else if (t1==min && t1==t2 && t1==t3)  
92     {  
93         cout << "Сено, силос и комбикорм закончатся одновременно"<< endl;
```

C:\Users\lenovo\Desktop\444.exe

Введите количество коров, шт. 1

Введите суточный рацион сена на 1 корову, кг 1

Введите суточный рацион силоса на 1 корову, кг 1

Введите суточный рацион комбикорма на 1 корову, кг 1

Сколько осталось сена, центнеров 1

Сколько осталось силоса, тонн 0.1

Сколько осталось комбикорма, мешков 2

Сена хватит на 100 дней

Силоса хватит на 100 дней

Комбикорма хватит на 100 дней

Корма хватит на 100 полных дней

Сено, силос и комбикорм закончатся одновременно

Process exited after 33.4 seconds with return value 0

Для продолжения нажмите любую клавишу . . .

C:\Users\lenovo\Desktop\444.exe

Введите количество коров, шт. 1

Введите суточный рацион сена на 1 корову, кг 1.1

Введите суточный рацион силоса на 1 корову, кг 2.1

Введите суточный рацион комбикорма на 1 корову, кг 3.5

Сколько осталось сена, центнеров 2

Сколько осталось силоса, тонн 2

Сколько осталось комбикорма, мешков 3

Сена хватит на 181 дней

Силоса хватит на 952 дней

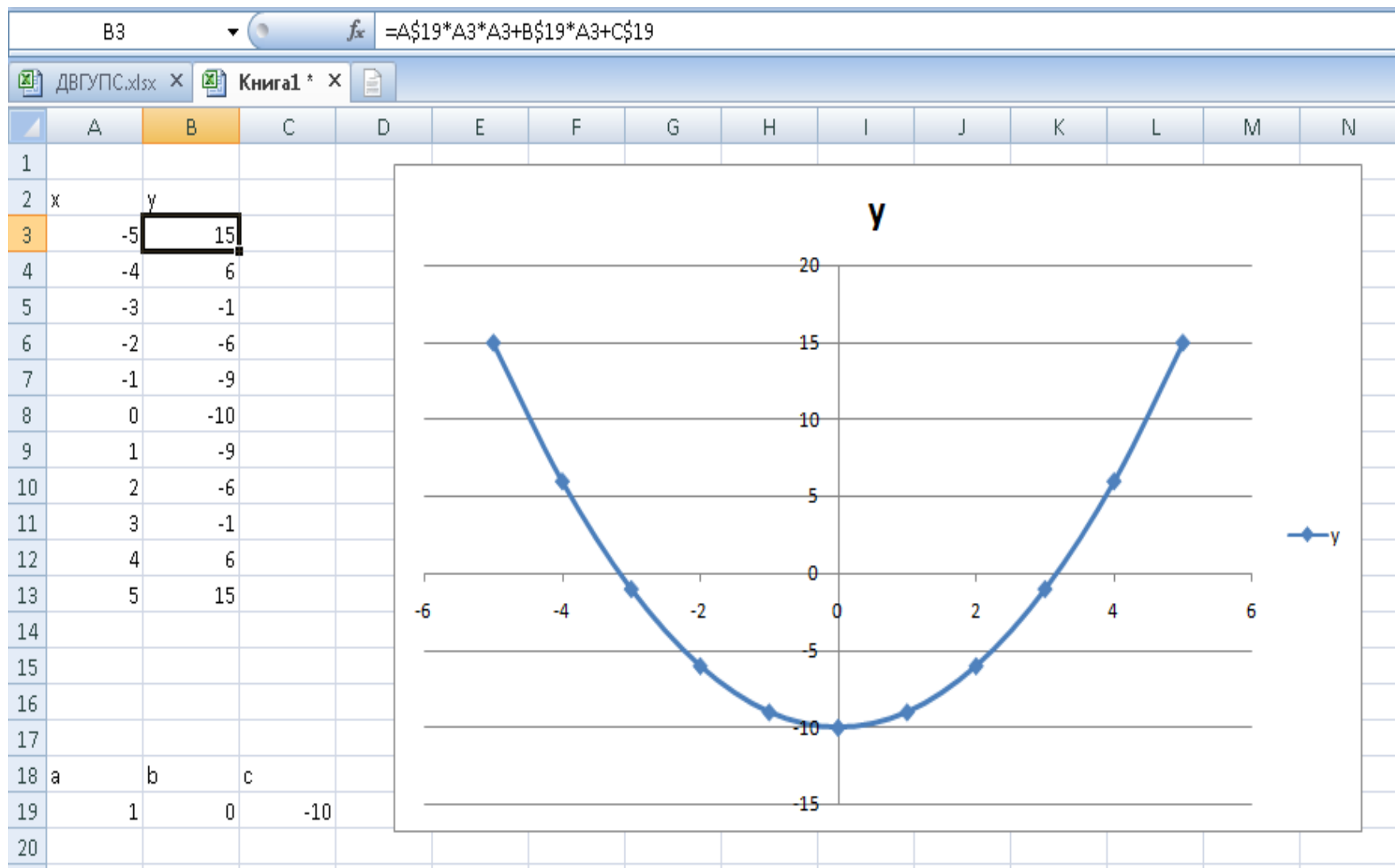
Комбикорма хватит на 42 дней

Корма хватит на 42 полных дней

Первым закончится комбикорм

Process exited after 21.78 seconds with return value 0

Для продолжения нажмите любую клавишу . . . █



float x=0, y=0, a=1, c=-10, b=0; // a,c задаем коэффициенты
уравнения параболы

```
cout << "Введите x: " << endl;
```

```
cin >> x ;
```

```
cout << "Введите y: " << endl;
```

```
cin >> y ;
```

```
b = a*x*x+c;
```

```
if (y>=b && y<=0)
```

```
{
```

```
cout << "Точка входит в заданный диапазон " << endl;
```

```
}
```

```
else
```

```
{
```

```
cout << "Точка не входит в заданный диапазон " << endl;
```

```
}
```

Задание 1

Вычислить значения функции F на интервале от начального значения $x = X_{\text{нач}}$ до конечного значения $x = X_{\text{кон}}$ с шагом dX . $X_{\text{нач}}$, $X_{\text{кон}}$, dX , a , b и c задаются пользователем и являются действительными числами.

Задание 2

Вычислить для заданного n указанное выражение.

Задание 3

Составить программу решения задачи.

Задание 4

Составить алгоритм и программу решения задачи.

Задание 5

Составить алгоритм и программу решения задачи.

1.

$$F = \left\{ \begin{array}{l} -ax^2 - b \\ \frac{2x - a}{cx} \\ \frac{-x}{b} \end{array} \right\}$$

при $c < 0$ и $a < 0$,

при $b \leq 0$ и $a \geq 0$,

в остальных случаях.

2. $\sin x + \sin x^2 + \dots + \sin x^n$

3. Дано натуральное число n . Вычислить $\sum_{k=1}^n (-1)^k (2k^2 + 1)!$

4. Найти все простые несократимые дроби, заключенные между 0 и 1, знаменатели которых не превышают 7. (Дробь задается двумя натуральными числами – числителем и знаменателем).

5. Численно проверить второй замечательный предел: $\lim_{x \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n = e$. При каком n исследуемое выражение отличается от e менее чем на заданную погрешность ε ?


```
cout << "Привет!";
float F=0,a=0,b=0,c=0,x=0;
cout << "Введите значение переменных a,b,c " << endl;
cin >> a >> b >> c;
float x1=0,x2=0,step=0;
cout << "Введите начальное и конечное значение x, а также шаг: " << endl;
cin >> x1 >> x2 >> step;
for (x=x1;x<=x2;x=x+step){

    if (c<0 && a<0){
        F=-a*x*x-b;
    } else if (b<=0 && a>=0){
        F=(2*x-a)/c*x;}
    else {
        F=-x/b;
    }
    cout << "Ответ: функция F равна " << F << endl;
}
```

```
int main(int argc, char *argv[])
{
    setlocale(LC_ALL, "Russian");

    cout << "Привет!";
    int n=0;
    float y=0, x=0;
    cout << "Введите x и n... ";
    cin >> x >> n ;

    for (int a = 1;a<=n; a++)
    {
        y=y+sin(pow(x,a));
    }
    cout << "Ответ: " << y << endl;
    return 0;
}
```

```
long double fact(int F)
{
    if(F < 0) // если у нас отрицательное число,
        return 0; // возвращаем ноль
    if (F == 0) // если у нас ноль,
        return 1; // возвращаем факториал нуля - 1
    else // во всех остальных случаях
        return F * fact(F - 1); // делаем рекурсию.
}

int main(int argc, char *argv[])
{
    setlocale(LC_ALL, "Russian");
    cout << "Привет!";
    int n=0,k=0,F=0;
    cout << "Введите натуральное число n "<< endl;
    cin >> n ;
    for (k=1;k<=n;k++){
        F+=pow(-1,k)*fact(2*pow(k,2)+1);
    }
    cout << " F = " << F << endl;
    cout << " Факториал = " << F << endl;
```

```
const int N = 7;
int main(int argc, char** argv)
{
    int found = 0, chisl=0, znam=0;

    for (znam = 2; znam <= N; znam++)
    {
        for (chisl = 1; chisl < znam; chisl++)
        {
            if (znam%chisl != 0) // условие проверки на
несократимость
            {
                printf("%4u.   %4d / %d\n", ++found, chisl,
znam); } // %u – десятичное число без знака, %d –
десятичное число целого типа со знаком.
        }
    }
```

```
1.      2 / 3
2.      3 / 4
3.      2 / 5
4.      3 / 5
5.      4 / 5
6.      4 / 6
7.      5 / 6
8.      2 / 7
9.      3 / 7
10.     4 / 7
11.     5 / 7
12.     6 / 7
```

Process exited after 0.166 seconds with return value 0

Для продолжения нажмите любую клавишу . . .

1.	2 / 3
2.	3 / 4
3.	2 / 5
4.	3 / 5
5.	4 / 5
6.	4 / 6
7.	5 / 6
8.	2 / 7
9.	3 / 7
10.	4 / 7
11.	5 / 7
12.	6 / 7

4

Process exited after 0.5002 seconds with return value 0

Для продолжения нажмите любую клавишу . . . ■

```
cout << "Введите величину погрешности для  
проверки второго замечательного предела" << endl;  
cin>>eps;  
e=exp(1.0);  
while (abs(k-e)>eps)  
{  
    n++;  
    k=pow((1+1.0/n),n);  
}  
cout<<"Минимальная погрешность при проверке  
числа e= "<<e<<" была достигнута при n = "<<n<<  
endl;
```

Рассчитать стоимость автомобильного номера (1вариант)

```
if (number < 1000 && number > 0) {
    thir = number%10;
    number /= 10;
    sec = number%10;
    number /= 10;
    fir = number;

    if (fir == sec && sec == thir) {
        cout <<"Номер автомобиля -три одинаковых цифры- стоит 100 тыс. рублей";
    } else if (fir == sec || fir == thir || sec == thir) {
        cout <<"Номер автомобиля -две одинаковых цифры- стоит 30 тыс. рублей";
    } else if (fir == sec-1 && sec == thir-1 || fir == sec+1 && sec == thir+1) {
        cout <<"Номер автомобиля -лесенка- стоит 20 тыс. рублей";
    } else {
        cout <<"Номер автомобиля обычный, с вас штраф 5 тыс. за некреативность";
    }
} else {
    cout <<"Введён некорректный номер";
}
```




Проект

Классы

[*] main.cpp

```
11 int a, b, c; // переменные для последнего числа, второго и первого числа номера
12
13 cout << "Введите автомобильный номер" << endl;
14 cin >> num;
15
16 a = num-10 * floor(num/10); // выделяем последнее число номера с помощью его деления на 10
17
18 num=num/10; // отбрасываем последнее число
19
20 b = num-10*floor(num/10); // повторяем действия, чтобы узнать второе число
21
22 num=num/10;
23
24 c=num; // присваиваем переменной первую цифру номера
25
26 cout <<a<<endl;
27 cout <<b<<endl;
28 cout <<c<<endl;
29
30 if (a==c && b!=c)
31 cout <<"Цена номера 100 тыс. руб." <<endl;
32 else
33 if (a==c && b==c && a!=1)
34 cout <<"Гони 150" <<endl;
35 else
36 if (a==1 && b==1 && c==1 )
37 cout <<"Гони 200" <<endl;
38 else
39 if (b==a+1 && c==b+1)
40 cout <<"Гони 10" <<endl;
41
42 return 0;
43 }
```

**Рассчитать стоимость
автомобильного номера
(2 вариант)**

Компилятор Ресурсы Журнал компиляции Отладка Результаты поиска Закрывать

Прервать

- Errors: 0

Line: 24 Col: 46 Sel: 0 Lines: 43 Length: 1050 Вставка Done parsing in 0,016 seconds



Задача про идеальный вес

Вес у нормального человека не должен отклоняться более чем на 10% от нормы.

Идеальное соотношение:

вес — (рост - 100) = 0. Составьте программу, которая будет выдавать человеку рекомендации по его весу и росту (все в порядке (в пределах 10%), в других случаях: нужно похудеть на... кг., нужно поправиться на... кг.).

```
cout << "Здравствуйте, введите свой вес " << endl;
cin >> a;
cout << "Здравствуйте, введите свой рост " << endl;
cin >> b;
c=abs(a-(b-100));
if (a*0.1>=(c*100/a)) {
cout << "У Вас все в норме! " << endl;
}
else {
cout << "У Вас есть проблемы: ";
}
if ((a-(b-100))>0 && (a*0.1<(c*100/a))) {
    cout << "нужно похудеть на " << c << " кг.";
}
if ((a-(b-100))<0 && (a*0.1<(c*100/a))) {
    cout << "нужно поправиться на " << c << " кг.";
}
```

Составить алгоритм и программу решения задачи. Вычислить с точностью до заданного $0 < \varepsilon < 1$:

$$\exp(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \dots + \frac{x^n}{n!} + \dots$$

Значение $0 < x < 2$ задать с клавиатуры. Считать, что нужная точность достигнута, если очередное слагаемое по модулю меньше ε .

```
cout << "Введите 0 < x < 2: ";
cin >> x;
cout << "Введите точность 0 < e < 1: ";
cin >> e;
if (x > 0 && x < 2 && e > 0 && e < 1) {
    for (; temp > e; n++) {
        fact *= n;
        temp = (pow(x, n)) / (fact);
        exp += temp;
    }
    cout << exp;
} else {
    cout << "Введены некорректные данные";
}
```

СПАСИБО ЗА ВНИМАНИЕ!