Лабораторная работа 2
Встроенные типы данных. Решение простейших вычислительных задач.

Задание:
Изучить теоретический материал по темам:

1. Операторы ветвления.
2. Операторы цикла.
3. Нестандартная запись операторов ветвления и операторов цикла.
4. Операторы Break и Continue
5. Функции

Написать программу с меню, которая по запросу пользователя выполняет одно из следующих действий

1. Числовой треугольник: по введенному положительному целому числу n вывести на печать:
   0
   01
   012
   0123
   *****
   0123…….n
2. Нахождение биномиальных коэффициентов: по заданному целому положительному n (n<100), вывести на печать последовательность биномиальных коэффициентов. Биномиальные коэффициенты считаются по формуле:
$$C_n^k = \frac{n!}{(n-k)! \cdot k!}$$
3. Вычисление среднего арифметического введенных точек на прямой, заданных своими координатами. Пока не введен символ конца ввода, считываем число, добавляем его к сумме уже введенных и счетчик чисел увеличиваем на 1. Если введен символ конца ввода, то сумму делим на количество чисел.
4. выход

Критерии оценивания: сделано меню и действия 1 и 4 – удовлетворительно, сделано меню и 3 действия – хорошо, все выполнено – отлично.

Форма представления выполненных заданий: файл *Ivanov_Ivan_111_111_lab2.cpp* содержащий решение поставленных задач.

Теоретические сведения:
Операторы ветвления.

if(<condition>){
  <stastement>;
}
else{
  <stastement>;
}
	int a,b,max
	if(a>b){
	  max=a;
	}
	else{
	  max=b;
	}

switch (<expression>)
  case <constant-expression> :
  <statement>
	int grade;
	switch(grade){
	  case 5:
	  std::cout<<"Otlichnick\n"; //отличник
	  break;

[default:
<statement>]
	case 4:
	std::cout<<"Choroshyst\n"; //хорошист
	break;
	case 3:
	std::cout<<"Troechnick\n"; //троечник
	break;
	case 1:
	case 2:
	std::cout<<"Dvoechnick\n"; //двоечник
	break;
	default:
	std::cout<<"Error\n"; //ошибка
	}

Условие в операторе ветвления может быть составным. Несколько условий соединяются логическими операциями:

AND	&&	a > b && b <10
OR	||	a > b || a < 3
NOT	!	! (a == 0)

Циклы.
Цикл - конструкция, позволяющая повторить некоторую инструкцию несколько раз.

for (<init-expression> ;
<cond-expression> ;
<loop-expression>)
<statement>;
	for (int i = 0; i < 2; i++){
	std::cout << i;
	}
	// Output: 01
		1. init-expression выполняется только один раз. Затем управление передается cond-expression.
		2. statement выполняется, только если cond-expression имеет значение true.
		3. В конце каждой итерации statement выполняется loop-expression.
		4. Statement может состоять из нескольких операторов, заключенных в фигурные скобки.

while (expression)
statement
	int i = 0, sum = 0;
	while (i < 1000)
	{
	i++;
	sum += i;
	}
	std::cout << sum;
	// Output:499500
		1. Значение параметра *выражение* проверяется перед каждым выполнением цикла, поэтому цикл while выполняется ноль или несколько раз.
		2. Выполнение цикла while может прекращаться, когда в его теле будет выполнен оператор break, goto или return.
		3. Чтобы прервать текущую итерацию без выхода из цикла while, используйте оператор continue. Оператор **continue**

		передает управление в следующую итерацию цикла while.

Пример создания меню:

```
#include <iostream>

//функция для решения задачи 1
void task1()
{}

//функция для решения задачи 2
void task2()
{}

int main()
{
   int choice =0; // в этой переменной будет храниться выбор пользователя
   while (true) // непрерывный цикл
   {
      std::cout << "Что вы хотите выполнить? \n"
      << 1. задание 1\n"
      << 2. задание 2\n"
      << 3. выход\n";
      std::cin >> choice;
      switch (choice)
      {
         case 1:
         {
            /*вызов функции для решения задания 1*/;
            task1();
            break; // если его не ставить, то после вызова task1(), начнет выполнятся task2()
         }
         case 2:
         {
            /*вызов функции для решения задания 1*/;
            task2();
            break;
         }
         default:
         {
            return 0;
         }
      }
   }
}
```