Функции, операторы, массивы

Задача 1.2: Перевод расстояния в милях в км с использованием функций.

Функция – именованный блок программного кода (вызывается по имени). При описании функции указывается тип возвращаемого результата, имя и список аргументов, в фигурных скобках приводится тело функции (программный код, кторый выполняется при вызове функции). Функция пересчета милей в км

```
double getDistKm(double dist) {
    //В одной миле километров
    double KmInMile=1.609344;
    //Результат функции
    return dist*KmInMile;
}
```

Задача 1.2: Перевод расстояния в милях в км с использованием функций.

Функция для считывания расстояния в милях:

```
double getDistMile() {
    // Переменная для записи результата функции
    double dist;
    // Запрос на ввод расстояния в милях
    cout<<"Укажите расстояние в милях  ->  ";
    // Считываение значения для расстояния в милях:
    cin>>dist;
    // Результат функции
    return dist;
}
```

Область доступности переменной ограничена блоком, в котором она объявлена. Такие переменные называются локальными. Код функции выполняется при ее вызове.

Задача 1.2: Перевод расстояния в милях в км с использованием функций.

Основная программа с вызовом обеих функций:

```
int main()
{
    //Изменение кодировки консоли
    setlocale(LC_ALL, "Russian");
        //Переменная для записи расстояния в милях
    double distMile=getDistMile();
    // Выполнение вычислений
    cout<<"Расстояние в км :  "<<getDistKm(distMile)<<endl;
    //Задержка консольного окна
    system("pause>nul");
    return 0;
}
```

Результат выполнения программы:

```
Укажите расстояние в милях  ->  28
Расстояние в км :  45.0616
```

Задача 2.0: Сумма квадратов чисел, оператор цикла.

Код программы:

```
int main()
{
    //Изменение кодировки консоли
    setlocale(LC_ALL, "Russian");
    // Верхняя граница суммы, сумма и индексная переменная
    int n=10, s=0, k=1;
    // Оператор цикла для вычисления суммы
    while (k<=n) {
        // Добавление нового слагаемого к сумме
    s=s+k*k;
    // Увеличение на 1 значения индексной переменной
    k++;
    }
    // Отображение результата вычислений
    cout<<"   Сумма квадратов от 1 до  "<< n <<" : "<<s<<endl;
    //Задержка консольного окна
    system("pause>nul");
    return 0;
}
```

Задача 2.0: Сумма квадратов чисел, оператор цикла.

Оператор цикла **while** выполняется до тех пор, пока логическое выражение истинно, то есть пока **k** не превышает **n**.

k++ увеличивает значение **k** на единицу.

Такие операторы называются *унарными*:

k++ эквивалентно **k=k+1**; - оператор инкремента.

k-- эквивалентно **k=k-1**; - оператор декремента.

```
Сумма квадратов от 1 до  10 : 385
```

Задача 2.1: Сумма квадратов чисел, оператор цикла, альтернатива

Код программы:

```
int main()
{
    //Изменение кодировки консоли
    setlocale(LC_ALL, "Russian");
    // Верхняя граница суммирования
    // Инициализация суммы
    int n, N, s = 0;
    // Ввод значения для верхней границы суммирования
    cout << "   Верхняя граница суммы  ->    ";
    cin >> n;
    N = n; // сохранение верхней границы суммиров. для вывода
    // Оператор цикла для вычисления суммы
    while (n) {
        // Добавление нового слагаемого к сумме
        s += n * n;
        // Уменьшение на 1 значения переменной
        n--;
    }
    // Отображение результата вычислений
    cout << "   Сумма квадратов  " << N << " чисел : " << s << endl;
    //Задержка консольного окна
    system("pause>nul");
    return 0;
}
```

Задача 2.1: Сумма квадратов чисел, оператор цикла, альтернатива

В этом варианте логическое выражение заменено числовым, при этом число, отличное от 0, интерпретируется как **true**, а нулевое, как **false**.

Также здесь использована сокращенная форма оператора присваивания:

x+=y эквивалентно **x=x+y**; точно так же

x-=y эквивалентно **x=x-y**; **x*=y** эквивалентно **x=x*y**;

и так далее

```
Верхняя граница суммы  ->   12
Сумма квадратов  12 чисел : 650
```

Задача 2.2: Условный оператор, проверка правильности ввода

Условный оператор строится так: после ключевого слова **if** в круглых скобках указывается условие (*логическое выражение*), далее следует блок команд в фигурных скобках, которые выполяются при истинности условия. Если условие ложно, то выполняется блок команд после ключевого слова **else**

```
if (n>0) {
    cout<<"  Введено правильное значение  ";
    }

    else {
        cout<<"  Введено неправильное значение  ";
    }
```

Задача 2.2: Условный оператор, проверка правильности ввода

```
int main()
{
    //Изменение кодировки консоли
    setlocale(LC_ALL, "Russian");
    // Верхняя граница суммы и инициализация s
    int n, N, s=0;
    // Ввод значения для верхней границы ряда
    cout<<"   Верхняя граница суммы  ->   ";
    cin>>n;
    N = n;
    // Если введено положительное число
    if (n>0) {
                // Оператор цикла для вычисления суммы
        while (n) {
            // Добавление нового слагаемого к сумме
            s+=n*n;
            // Уменьшение на 1 значения переменной
            n--;
        }
    // Отображение результата вычислений
    cout<<"   Сумма квадратов  "<< N <<" чисел : "<<s<<endl;
    // Если введено отрицательное число или 0
    }
    else {
        cout<<"  Введено некорректное значение  ";
    }
    //Задержка консольного окна
    system("pause>nul");
    return 0;
}
```

Условный оператор

В условном операторе между блоками **if ()** и **else** можно разместить любое количество блоков **else if ()** с соответствующими логическими выражениями в круглых скобках.

Пример:

```
if (n == 1) {
    cout<<"  Введено число 1  ";
    }

    else if (n == 2) {
    cout<<"  Введено число 2  ";
    }

    else if (n == 3) {
    cout<<"  Введено число 3  ";
    }

    else {
        cout<<"  Введено другое число  ";
    }
```

Задача 3.0: Создание одномерного массива

Одномерный массив – это упорядоченная линейная конструкция элементов, имеющих общее имя, но различающихся своими индексами.

Объявление массива: сначала указывается идентификатор типа (все элементы массива должны быть одного типа), затем указывается имя массива и в квадратных скобках его размер – количество элементов.

При обращении к элементу массива указывается имя и индекс в квадратных скобках. Индекс первого элемента 0, а индекс последнего – на единицу меньше размера массива. Размер массива задается константой (обычная переменная

Задача 3.0: Создание одномерного массива

Создадим массив, элементы которого представляют арифметическую прогрессию с начальным значением для первого элемента массива **Num[0] = 1**. Затем каждый последующий элемент массива образуется из предыдущего добавлением переменной d = 3.

Для создания массива воспользуемся оператором цикла **while** с соответствующим логическим выражением.

Вычисленные элементы массива выведем в одну строчку, и предусмотрим символ | в качестве разделител

Задача 3.0: Создание одномерного массива

```
int main()
{
    //Изменение кодировки консоли
    setlocale(LC_ALL, "Russian");
    // Константа для определения размера массива
    const int n = 10;
    // Создание целочисленного массива
    int Num[n];
    int k = 0; // Индексная переменная
    int d = 3; // параметр изменения элементов массива
    // Первый элемент массива
    Num[0] = 1;
    // Отображение первого элемента массива
    cout <<endl<< " | " << Num[0];
    // Оператор цикла для заполнения массива
    while (k < (n - 1)) {
        // Значение элемента массива
        Num[k + 1] = Num[k] + d;
        // Отображение элемента массива
        cout << " | " << Num[k + 1];
        // Изменение значения индексной переменной
        k++;
    }
    cout<<" |" << endl;
    //Задержка консольного окна
    system("pause>nul");
    return 0;
}
```

Спасибо за внимание!