**ЛАБОРАТОРНАЯ РАБОТА № 8**

**С++: ДАННЫЕ. УКАЗАТЕЛИ И МАССИВЫ**

## ТЕОРИЯ И ПОЯСНЕНИЯ

### *Этапы подготовки переменных и функций к использованию в программе*

Основными элементами программы на языке C являются переменные и функции. Любой из этих элементов должен быть подготовлен к использованию. В целом эта подготовка выполняется в 3 этапа: объявление, определение и инициализация.

***Объявление*** – задание имени и типа переменной или функции.

***Определение*** – создание переменной (выделение памяти) или задание исходного кода функции.

***Инициализация*** – присвоение начального значения переменной.

Для переменных объявление и определение обычно выполняются одновременно. Например,



Глобальная переменная, т.е. переменная, объявленная вне функций, получает по умолчанию нулевое начальное значение; для локальных переменных начальное значение не определено. Если заранее известно, каким должно быть начальное значение, то инициализация проводится вместе с определением переменной, например,

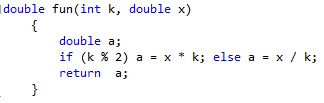


Объявление функции – это задание ее прототипа, т.е. имени функции, типов параметров и типа возвращаемого значения (имена параметров можно не задавать).

Например,



Определение функции включает полный заголовок с именами параметров и исходный код (тело функции). Например,



Если функция не была объявлена ранее в файле программы, то определение включает и объявление функции.

Присваивать переменной новое значение можно только после ее объявления и определения – с помощью оператора присваивания или оператора ввода. Использовать переменную в сравнениях и вычислениях можно только после присвоения ей значения (в том числе, после инициализации).

Вызывать функцию, например, можно только после ее объявления или определения. Это связано с тем, что код программы просматривается от начала до конца только один раз, поэтому объявление и/или определение любого объекта должно проводиться до его первого использования.

### *Область видимости данных*

Объявленным именем можно пользоваться только в определенной части программы. Областью видимости переменной является блок, в котором она объявлена, от места объявления до конца блока. Глобальные переменные объявляются вне функций. Если имена глобальной и локальной переменных совпадают, то видимой в своем блоке остается локальная переменная. Областью видимости формальных параметров функции является тело этой функции.

### *Указатели* *в С++*

Указатель – это переменная, значением которой является адрес области памяти, выделенной для переменной или значения определенного типа. Объявление указателя содержит тип данных, с которым этот указатель будет связан. Например, объявление и определение указателей на разные типы данных:



**Операторы работы с памятью**

**new** – динамическое выделение памяти, инициализация переменной-указателя. Например,

****

**delete** – освобождение памяти. Например,



**Операции с указателями**

**Взятие адреса** – получение адреса переменной (с целью присваивания его указателю), унарный префиксный **&**. Например,



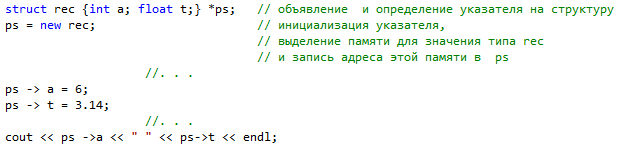
**Разыменование** – ссылка на объект, на который указывает указатель, унарная префиксная **\***.

Например,



**Адресация полей структуры с помощью указателя**

Если структура адресуется с помощью указателя, то при обращении к полям структуры используется операция **->**. Например,



**Арифметические операции с указателями**

К указателям можно применять только целочисленное сложение или вычитание. При изменении значения указателя учитывается размер памяти, выделяемый для типа, с которым связан указатель. Например,

**** после выполнения этого оператора адрес в **h** будет указывать не на следующий байт, а на **следующее целое число** (значение **h** увеличивается не на 1, а на 4, поскольку переменной типа **int** выделяется 4 байта).

### *Массивы в С++*

Имя массива – это указатель, который хранит адрес начала массива. Элементы массива нумеруются от 0.

Для задания числа элементов при объявлении массивов удобно использовать именованные константы, которые определяются с помощью директивы препроцессора **#define**. Директива должна появиться в исходном тексте до первого использования определяемого идентификатора. В конце директивы точка с запятой НЕ ставится.

**#define имя\_константы значение**

***Объявление и описание статического массива*** может выполняться разными способами. Например: целочисленный массив **s** из известного числа элементов:

**int s[константа];**

или

**int s[имя\_константы];**

Здесь значение константы – число элементов в массиве.

Имена статических массивов – это константы-указатели. Их значение менять нельзя. (Память под массив выделяется при компоновке исполняемого кода и не может быть изменена).

***Объявление динамического массива*** выполняется как объявление переменной-указателя на тип элементов массива. Например, целочисленный динамический массив **d**:



Переменная-указатель **d** – это имя динамического массива. Ее значение можно менять.

***Определение динамического массива*** должно быть выполнено перед первым обращением к массиву. Например, выделение памяти для динамического целочисленного массива из ХХХ элементов:



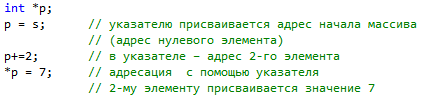
Память, выделенную для динамического массива, нужно освобождать, когда работа с массивом завершена:



Для обращения к элементу статического или динамического массива можно использовать индекс (индексная адресация) или адрес, хранящийся в указателе (адресация с помощью указателя, косвенная адресация). Например, обращение к элементу массива

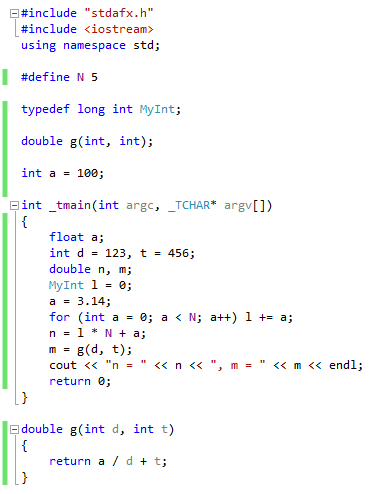


или

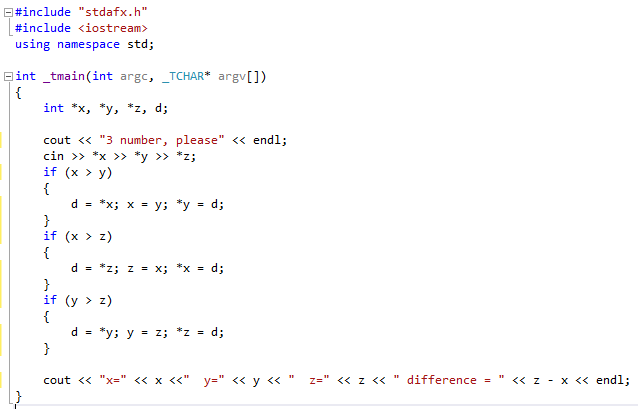


## ЗАДАНИЯ

### УПРАЖНЕНИЕ 1. Объясните использование типов и переменных в приведенной ниже программе. Определите (без компиляции и выполнения на компьютере), какие значения получат переменные n и m в результате выполнения этой программы.



### УПРАЖНЕНИЕ 2. Определите (без компиляции и выполнения на компьютере), какие ошибки имеются в приведенной ниже программе. Какую задачу решает эта программа?



### УПРАЖНЕНИЕ 3. Напишите 4 варианта алгоритма заполнения каждого из 10-и элементов целочисленного массива значением квадрата индекса этого элемента. Оформите вывод результатов из заполненного массива, используя тот же способ адресации, что и при заполнении массива.

***Требования к вариантам***:

1. статический массив, индексная адресация;
2. статический массив, адресация с помощью указателя (косвенная адресация);
3. динамический массив, индексная адресация;
4. динамический массив, адресация с помощью указателя (косвенная адресация).

### УПРАЖНЕНИЕ 4. Напишите программу создания линейного самоадресуемого списка из 10-и элементов. В поле данных каждого элемента списка – порядковый номер этого элемента. Вывод значений элементов списка выполните после построения списка.