Практическое занятие №2

Объявление и инициализация переменной. Базовые типы данных. Константы и литералы. Арифметические операторы.

Объявление и инициализация переменной

- В общем случае объявление переменной в C++ осуществляется путем указания типа переменной и ее имени. Синтаксис объявления переменной имеет вид тип имя переменной. В предыдущем примере целочисленная переменная объявлялась как int age.
- Идентификатором типа является инструкция int, а именем переменной age.

Объявление переменной

- Всего в C++ есть семь базовых типов, и о них речь пойдет несколько позже.
- Следует понимать: если переменная оглашена как такая, что принадлежит к определенному типу, в дальнейшем изменить ее тип невозможно.
- Объявляются переменные фактически в любом месте программного кода, однако использовать в программе переменную можно только после того, как ей присвоено значение. В этом случае говорят об инициализации переменной.

Инициализация переменной

- Инициализация переменной также может выполняться в любом месте программы, но не ранее объявления этой переменной и до места первого ее использования.
- Объявление и инициализацию переменной часто совмещают. Например, инструкцией `int n=10` объявлена целочисленная переменная `n` со значением 10. Если объявляется несколько переменных одного типа, достаточно перечислить эти переменные, разделенные запятой, после идентификатора типа.
- То же самое касается инициализации переменных при объявлении.

Пример

- `int n, m, k; // Объявление трех целочисленных переменных n, m и k`

- `int one = 1, two = 2, three, four = 4, five;/* Объявление с одновременной`
- `инициализацией нескольких переменных*/`

Объявление и инициализация переменной

- Однако, несмотря на такой демократизм, объявлять переменные лучше в начале соответствующего программного блока, причем желательно указывать персонально для каждой переменной идентификатор типа.
- Программный код в таком случае пользователем воспринимается намного лучше.
- В C++ существует возможность при инициализации переменных использовать не только литералы (значения, которые известны во время компиляции, что позволяет компилятору помещать их в отдельное адресное пространство только для чтения в полученных двоичных файлах), но и выражения, в которые входят другие переменные. Единственное условие состоит в том, чтобы выражение, на основании которого инициализируется такая переменная, было легитимным на момент инициализации.

Пример

```
using namespace std; // Инструкция для компилятора использовать стандартную область имен

int main() // Главная функция программы
{
    setlocale(LC_ALL, "Russian"); // Изменение кодировки консоли


        // Скорость тела – объявление переменной
        double v;
        // Время полета
        double t = 1.2; // литерал
        // Ускорение свободного падения
        double g = 9.8; // литерал
        // Скорость – инициализация переменной
        v = 12.3; // литерал
        // Высота – динамическая инициализация переменной
        double s = v * t - g * t * t / 2; //
        cout << "Height level is " << s << endl;

    system("pause"); // Задержка консольного окна
    return 0; // Значение, возвращаемое главной функцией
}
```

Пояснения

- Идентификатором действительного типа является инструкция double.
- В начале программы разными способами инициализируются переменные v, g и t (соответственно начальная скорость, ускорение свободного падения
- и время), после чего с помощью команды double s=v*t-g*t*t/2 выполняется динамическая инициализация переменной s, которая определяет высоту, на которой пребывает тело.
- Результат расчетов выводится на экран.
- В выражении инициализации были использованы операторы умножения (*), вычитания (-) и деления (/). В качестве оператора присваивания использован знак равенства (=). Про операторы, используемые в C++, речь еще будет идти, а сейчас остановимся на том, какого типа переменные могут использоваться в C++ и в чем их особенность.

Задание

- 1. Напишите программу, запрашивающую у Вас отдельно градусы, минуты и секунды, а затем возвращающую их десятичное значение (deg) .
- 2. Пользуясь интернетом, найдите конструкцию, позволяющую указывать точность данных числовых типов до указанного количества знаков после запятой.
- 3. Напишите программу осуществляющую обратный переход, то есть от десятичных градусов (deg) в градусы, минуты и секунды.

  Подсказка:

```
int a, b, x, y;
cin >> x;
cin >> y;
a = (int)(x / y); //целая часть от деления (div)
b = x % y; // остаток от целения (mod)
cout << "Целая часть от деления " << a << endl;
cout << "Остаток от деления " << b << endl;
```

Базовые типы данных

- Как уже отмечалось, в C++ существует семь базовых, или основных, типов данных (точнее, семь базовых идентификаторов типа).

Таблица 1.1. Основные типы данных

Идентификатор типа	Тип данных
`bool`	Логический тип
`char`	Символьный тип
`wchar_t`	Символьный двухбайтовый тип
`double`	Действительные числа двойной точности
`float`	Действительные числа
`int`	Целые числа
`void`	Значение не возвращается

Ключевое слово void

- Ключевое слово void используется при определении функций, которые не возвращают результата. Это функции – аналог процедур в таких языках программирования, как, например, Pascal. Ключевое слово void также используется при определении обобщенных указателей.

Модификаторы типа

- Вместе с идентификаторами типов могут использоваться так называемые модификаторы типа.
- Модификаторы типа – это специальные ключевые слова, которые указываются перед идентификатором типа и позволяют изменять базовый тип.
- В C++ используются модификаторы signed (значения со знаком), unsigned (значения без знака), short (укороченный тип) и long (расширенный тип). Все четыре модификатора могут использоваться для типа int.
- Модификаторы signed и unsigned, кроме этого, используются с типом char, а с типом double используют модификатор long.

Диапазон значений для разных типов

- Что касается диапазона значений для данных разных типов, то в разных компиляторах диапазоны значений данных различны. В языке C++ вводятся стандарты только для минимально необходимого диапазона, который должен поддерживать компилятор.
- В таблице 1.2 перечислены минимальные диапазоны в 32-разрядной среде для данных разных типов с учетом наличия модификаторов типов.

Минимально необходимые диапазоны для данных разных типов

Таблица 1.2. Минимально необходимые диапазоны для данных разных типов

Тип	Количество бит	Диапазон значений
bool	1	Значения true или false
char	8	от -128 до 127
wchar_t	16	от 0 до 65535
double	64	от 2.2E-308 до 1.8E+308
float	32	от 1.8E-38 до 1.8E+38
int	32	от -2147483648 до 2147483647
unsigned char	8	от 0 до 255
signed char	8	от -128 до 127
unsigned int	32	от 0 до 4294967295
signed int	32	от -2147483648 до 2147483647

Минимально необходимые диапазоны для данных разных типов

Тип	Количество бит	Диапазон значений
short int	16	от -32768 до 32767
unsigned short int	16	от 0 до 65535
signed short int	16	от -32768 до 32767
long int	32	от -2147483648 до 2147483647
unsigned long int	32	от 0 до 4294967295
signed long int	32	от -2147483648 до 2147483647
double	64	от 2.2E-308 до 1.8E+308
float	32	от 1.8E-38 до 1.8E+38
long double	64	от 2.2E-308 до 1.8E+308

Константы и литералы

- В C++ под константами и литералами, как правило, подразумевают одно и то же: это такие значения, предназначенные для восприятия пользователем, которые не могут быть изменены в ходе выполнения программы. Приведенное определение относится большей частью к литералу.
- Под константами здесь и далее будем подразумевать именованные ячейки памяти, значения которых фиксируются на начальном этапе выполнения программы и затем в процессе выполнения программы не могут быть изменены. В этом смысле константы – это те же переменные, но только с фиксированным, определенным единожды, значением

Константа

- Для превращения переменной в постоянное запоминающее устройство, то есть в константу, используют идентификатор const. Идентификатор указывается перед типом переменной и гарантирует неизменность значения переменной. Само значение переменной присваивается либо при объявлении, либо позже в программе, но только один раз и до первого использования переменной в программе.
- Например, инструкцией const int m=5 инициализируются целочисленная переменная m со значением 5. После этого переменную m можно использовать в выражениях, однако изменить значение переменной не удастся.

Литерал

- Представление литералов в программном коде существенно зависит от типа, к которому относится литерал. Так, отдельные символы – литералы типа char указываются в одинарных кавычках, например 'a' или 'd'. Для литералов типа wchar_t (напомним, это расширенный 16-битовый символьный тип) перед непосредственно значением указывается префикс L. Примеры литералов типа wchar_t: L'a', L'A', L'd' и L'D’.
- Несмотря на то, что в C++ нет отдельного типа для строчных (текстовых) переменных, в C++ существуют литералы типа текстовой строки. Соответствующие значения указываются в двойных кавычках. Примером текстового литерала, например, может быть фраза "Hello, World!".

Литерал

- Числовые литералы, как и положено, задаются в стандартном виде с по- мощью арабских цифр. Однако здесь есть один важный момент. Связан он с тем, что в C++ есть несколько числовых типов, поэтому не всегда очевидно, к какому именно числовому типу относится литерал. Здесь действует общее правило: литерал интерпретируется в пределах минимально необходимого типа.
- Это правило, кстати, относится не только к числовым литералам. Исключение составляет тип double: литералы в формате числа с плавающей точкой интерпретируются как значения этого типа (а не типа float, как можно было бы ожидать).

Арифметические операторы

- Выражения в C++ содержат, помимо переменных, еще и операторы. Операторы условно можно поделить на арифметические, логические, поразрядные и операторы отношения. Далее описываются особенности работы с каждой из означенных групп операторов. В первую очередь остановимся на арифметических операторах.
- Арифметические операторы используются для сложения, вычитания, умножения и деления чисел. Список основных арифметических операторов, которые используются в C++, приведен в таблице 1.3.

Арифметические операторы C++

Таблица 1.3. Арифметические операторы C++

Оператор	Назначение
+	Сложение
–	Вычитание
*	Умножение
/	Деление. Если операндами являются целые числа, выполняется целочисленное деление
%	Остаток от деления (деление по модулю)
++	Инкремент
––	Декремент

Арифметические операторы C++

- Первые пять операторов являются бинарными, то есть используются с двумя операндами. За исключением операторов деления по модулю (остаток от целочисленного деления), инкремента и декремента, операторы совпадают с соответствующими математическими операторами. Если оператор деления по модулю особых комментариев не требует, то операторы инкремента и декремента в определенном смысле являются визитной карточкой языка C++. Оператор инкремента даже присутствует в названии языка.

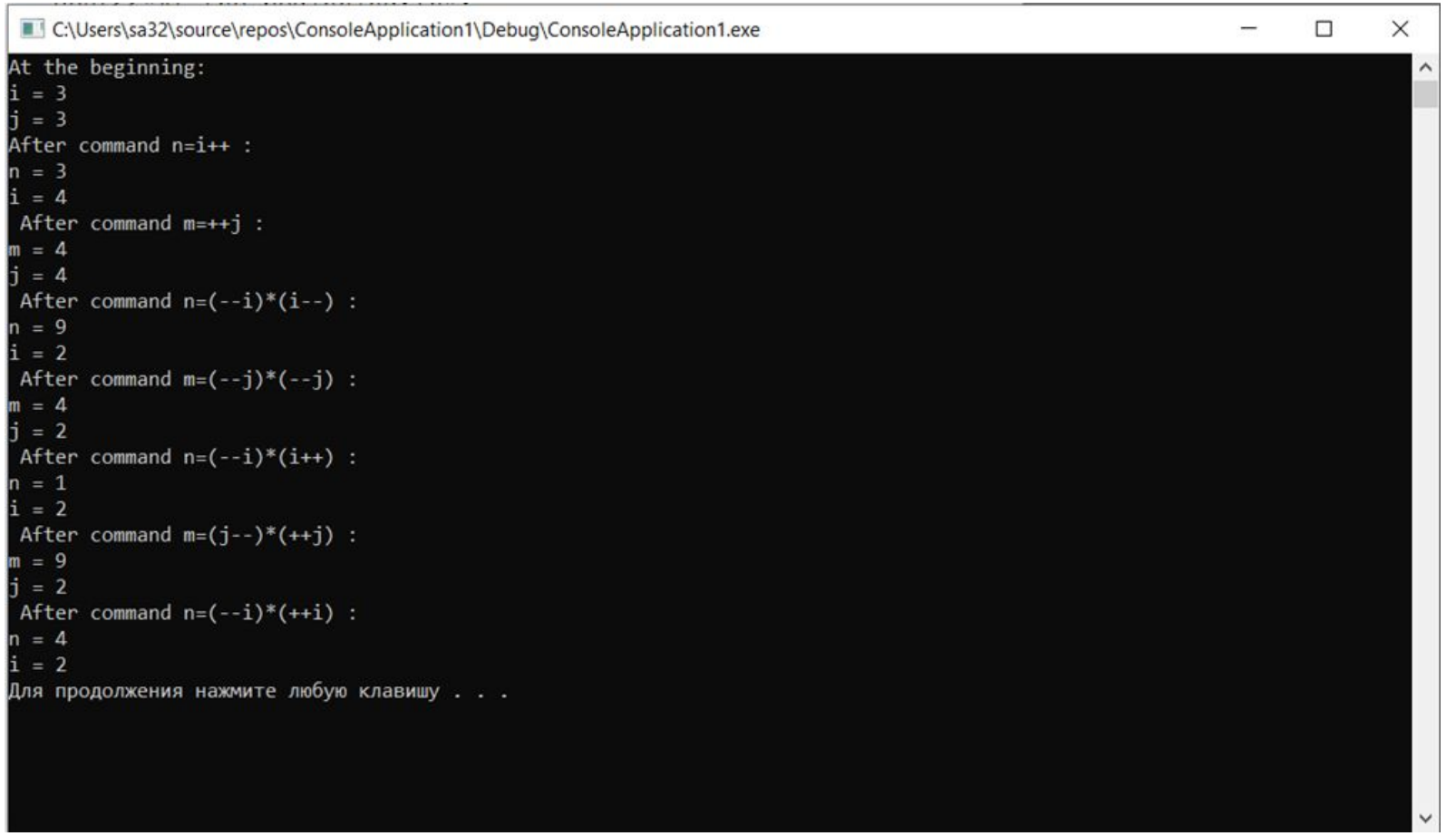
Операторы инкремента и декремента

- Операторы инкремента и декремента являются унарными (используются с од- ним операндом). Действие операторов состоит в увеличении или уменьшении значения операнда на единицу. Например, результат выполнения команды i++ есть увеличение значения переменной i на единицу. Другими словами, команды i++ и i=i+1 с точки зрения влияния на значение переменной i являются эквивалентными. Аналогично в результате команды i-- значение перемен- ной i уменьшается на единицу, как и в результате выполнения команды i=i-1.

Операторы инкремента и декремента

- Операторы инкремента и декремента могут использоваться в префиксной и постфиксной формах. В префиксной форме оператор указывается перед операндом, а в постфиксной форме – после него. Выше операторы инкремента и декремента использовались в постфиксной форме. В префиксной форме соответствующие операции выглядели бы как ++i и --i.
- Для префиксной формы сначала изменяется значение переменной, а затем рассчитывается выражение, а для постфиксной формы выражение вычисляется со старым значением переменной, а только после этого изменяется значение этой переменной. В этом и состоит разница между

Пример


C:\Users\sa32\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe
At the beginning:
i = 3
j = 3
After command n=i++ :
n = 3
i = 4
After command m=++j :
m = 4
j = 4
After command n=(--i)*(i--) :
n = 9
i = 2
After command m=(--j)*(--j) :
m = 4
j = 2
After command n=(--i)*(i++) :
n = 1
i = 2
After command m=(j--)*(++j) :
m = 9
j = 2
After command n=(--i)*(++i) :
n = 4
i = 2
Для продолжения нажмите любую клавишу . . .

Спасибо за внимание!