Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Реферат

По дисциплине «Основы алгоритмизации и программирования»

На тему «Порядок выполнения операций»

Выполнил:

Студент ПИ 1 курса 10 группы 1 подгруппы

Сегренёв Кирилл Сергеевич

Преподаватель: Белодед Н.И

2023, Минск

Содержание

[**ВВЕДЕНИЕ** 3](#_Toc149679701)

[**ПОСТОНОВКА ЗАДАЧИ** 4](#_Toc149679702)

[**АЛГОРИТМ** 6](#_Toc149679703)

[**ВЫВОД** 10](#_Toc149679704)

**ВВЕДЕНИЕ**

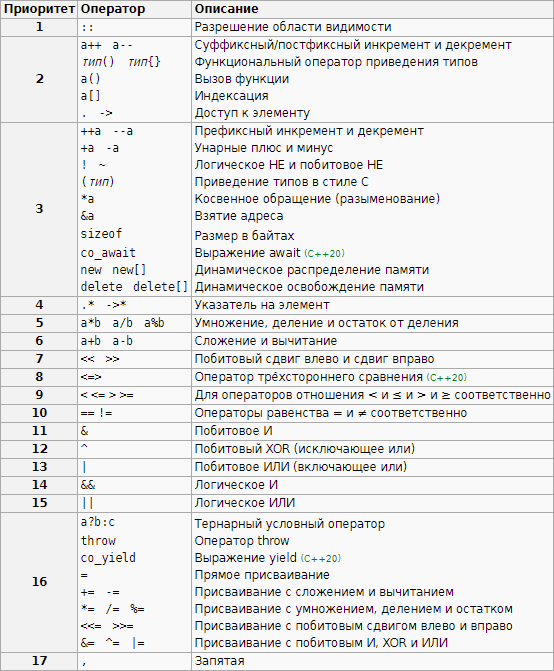
На лекции 31 октября преподавателем была поставлена задача объяснить код, который был показан на лекции.

Объяснить, почему выводятся такие значения переменных.

**ПОСТОНОВКА ЗАДАЧИ**

В данной задаче требуются знания о приоритетности операция в языках С

Для того, чтобы понимать, какая операция приоритетней, в google можно найти таблицу с приоритетностью операций



В данных нам примерах представлены действия над единственной переменной, но в каждом новом примере мы присваиваем ей новое значение и выполняем другие операции

В каждом задании i присваивается новому значению и после над i проводят разные операции. Нужно разобраться в какой последовательности происходят вычисления и почему выводятся именно такие значения переменных n и i.

**АЛГОРИТМ**

Данный код я писал в одном .cpp файле, поэтому у меня не будет объявление переменных int и функций main

**1 задание.**

i = 1;

cout << "At me beggining: i = " << i << '\n';

cout << "After n = i++ : \n";

n = i++;

// В данном случае выполняется присвоение значению n значения i, а после значение i увеличивается на 1

// Если выведем значение переменной i до увеличения операнда, то у нас будет выводится 1

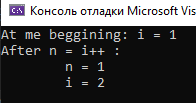
cout << "\tn = " << n << "\n";

cout << "\ti = " << i << "\n";

cout << "\n";

// Вывод: n = 1, i = 2

**Output:**

****

**2 задание.**

i = 2;

cout << "At me beggining: i = " << i << '\n';

cout << "After n = ++i :\n";

n = ++i; // В данном случае мы сначала увеличиваем нашу переменную i на 1, а уже после присваиваем увеличенной значение переменной n

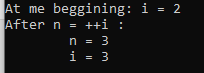
cout << "\tn = " << n << "\n";

cout << "\ti = " << i << "\n";

cout << "\n";

//Вывод: n = 3, i = 3

**Output:**



**Задание 3.**

i = 3;

cout << "At me beggining: i = " << i << '\n';

cout << "After n = (--i) \* (i--) :\n";

n = (--i) \* (i--);

// 1 действие: Вычисляется префиксный декремент --i = 2

// 2 действие: Умножаются измененные значения переменной: 2 \* 2 = 4

// 3 действие: Переменной n присваивается значение перемноженных чисел n = 4

// 4 действие: После выполнения всех операций вычисляется постфиксальный декремент, который нахо дится во вторых скобках: i-- = 1

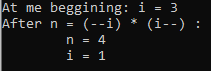
cout << "\tn = " << n << "\n";

cout << "\ti = " << i << "\n";

cout << "\n";

// Вывод: n = 4, i = 1

**Output:**



**Задание 4.**

i = 5;

cout << "At me beggining: i = " << i << '\n';

cout << "After n = (--i) \* (--i) :\n";

n = (--i) \* (--i);

// 1 действие: Т.к у нас в выражении есть 2 префиксных декремента, сначала вычисляются они, и значение i = 3

// 2 действие: Умножаем новые данные переменных 3 \* 3 = 9

// 3 действие: Присваиваем результат выполненного умножения переменной n = 9

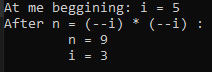
cout << "\tn = " << n << "\n";

cout << "\ti = " << i << "\n";

cout << "\n";

// Вывод: n = 9, i = 3

**Output:**

****

**Задание 5.**

i = 6;

cout << "At me beggining: i = " << i << '\n';

cout << "After n = (--i) \* (i++) :\n";

n = (--i) \* (i++);

// 1 действие: вычисляется префиксный декремент => i = 5

// 2 действие: перемножаются между собой уже измененные значения переменных: 5 \* 5 = 25

// 3 действие: присвоение значения переменной n = 25;

// 4 действие: вычисление постфиксального инкремента после всех действий: i++ = 6

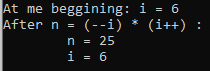
cout << "\tn = " << n << "\n";

cout << "\ti = " << i << "\n";

cout << "\n";

// Вывод: n = 25, i = 6

**Output:**

****

**Задание 6.**

cout << "At me beggining: i = " << i << '\n';

cout << "After n = (i--) \* (++i) :\n";

n = (i--) \* (++i);

// 1 действие: Вычисляется префиксный инкремент из-за большей приоритетности ++i = 8

// 2 действие: Вычисляется произведение между измененными значениями переменных 8 \* 8 = 64

// 3 действие: Присвоение значения переменной n = 64

// 4 действие: После выполнения присвоения переменной n значения, из-за постфиксального декремента, значение i уменьшается на 1: i = 7

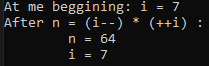
cout << "\tn = " << n << "\n";

cout << "\ti = " << i << "\n";

cout << "\n";

// Вывод: n = 64, i = 7

**Output:**

****

**Задание 7.**

i = 8;

cout << "At me beggining: i = " << i << '\n';

cout << "After n = i-- \* ++i :\n";

n = i-- \* ++i; // Отсутствие скобок может запутаться, но мы модем обратиться к таблице приоритета операций (в реферате будет прикреплен jpg-файл)

// В данной таблице мы видим, что в первую очередь будут выполняться постфиксальные декременты со значение приоритета 2( но это действие выполняется не в начале, а в конце, что я объясню в выводе)

// После будут выполняться префиксные инкременты: ++i = 9 (со значением приоритета 3)

// После будет выполняться умножение: 9 \* 9 = 81 (с приоритетом 5)

// И дальше будет идти присвоение значение переменной n = 81 ( значение приоритета 16)

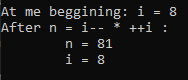
cout << "\tn = " << n << "\n";

cout << "\ti = " << i << "\n";

cout << "\n";

// Вывод: n = 81, i = 8

**Output:**



**Задание 8.**

i = 9;

cout << "At me beggining: i = " << i << '\n';

cout << "After n = (--i) \* (++i) :\n";

n = (--i) \* (++i);

// 1 действие: выполняются префиксные инкремент и декремент, в сумме они дают 0, поэтому i остается 9

// 2 действие: вычисления произведения значений i: 9 \* 9 = 81

// 3 действие: присвоение переменной n значение прошлого действия: n = 81

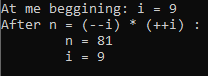
cout << "\tn = " << n << "\n";

cout << "\ti = " << i << "\n";

cout << "\n";

// Вывод: n = 81, i = 9

**Output:**



**ВЫВОД**

При рассмотрении алгоритмов подсчета в коде я заметил, что постфиксальный оператор несмотря на свою приоритетность (таблица прикреплена в постановке задачи), выполняет увеличение/уменьшение после всех операций.

Постфиксальные операторы сначала возвращают свое значение, а после увеличиваются/уменьшаются после выполнения операции(это хорошо можно увидеть применяя циклы)

Если брать наши примеры, то из таблицы приоритета операторов видно, что у постфиксных операторов приоритетность выполнения будет выше, чем у префиксных, но особенность работы постфиксных операторов я описал выше.

Но ниже по приоритетности уже идет умножение (приоритетность 5) и присвоение (приоритетность 16)