**Глава 3. Структура программы на C++. Операции**

**3.1 Точка входа в программу**

Основная часть программы – это функция main(). При запуске консольного приложения, написанного на языке программирования C++, операционная система компьютера передает управление функции с именем main. Функция main() не может быть вызвана из других программных функций, она является точкой входа в программу.

Функция main объявляется следующим образом: "int main()" или "void main()".

Круглые скобки после названия функции используются для передачи параметров. В этом случае операционная система не передает никаких параметров функции main(), поэтому в круглых скобках ничего нет.

Тип возвращаемого значения указывается перед именем функции. При обращении к функции main(), операционной системе возвращается целочисленное значение. В зависимости от возвращаемого значения операционная система определяет, правильно ли завершилась программа.

Если это значение не возвращается, операционная система понимает, что программа завершила работу в аварийном режиме. Чтобы вернуть целочисленное значение, перед завершением функции добавляется строка: "return 0;". Если программа завершила работу и вернула 0, это означает, что ошибок не произошло.

После объявления функции пишутся фигурные скобки, в которых находится тело программы. Пример программы, которая выводит текст на экран, приведён на рисунке 3.1.

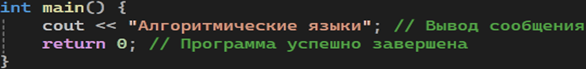


Рисунок 3.1 – Пример кода программы для вывода текста на языке C++

**3.2 Директивы препроцессора и пространства имён**

Помимо функции main(), любая программа содержит и другие важные компоненты. Программа на C++ начинается с директив препроцессора, таких как "#include",которые сообщают компилятору, какие библиотеки или файлы ему следует включить в программу [1]. Директивы помогают разработчикам использовать готовые функции и возможности языка. Общий вид программы на алгоритмическом языке C++ представлен на рисунке 3.2.

Строка "using namespace std;" указывает на то, что используется стандартное пространство имен "std". С помощью пространств имён программа следит за тем, чтобы имена переменных не повторялись.

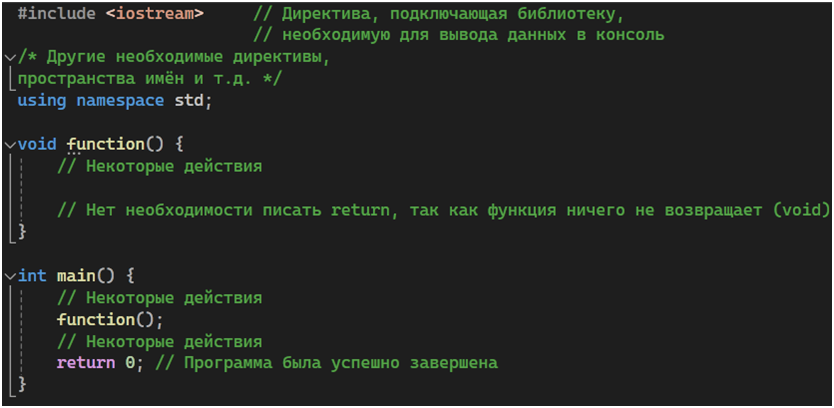


Рисунок 3.2 – Структура общего вида программы на C++

Например, с подключённой библиотекой "<iostream>", но без "using namespace std;"для вывода сообщений в консоль необходимо писать "std::cout<<"some data";", а с "using namespace std;"используется оператор вывода cout [14].

Пространства имен важны, потому что они позволяют изолировать фрагменты кода друг от друга. Без объявления пространства имен могут возникнуть конфликты имен, особенно если используются сторонние библиотеки или имеются большие проекты с множеством файлов. Пространства имен помогают предотвратить такие конфликты, разделяя код на логические группы и предоставляя уникальное пространство для каждой из них.

Далее будут рассмотрены функции из примера (рисунок 3.2). Наиболее важной из них является функция main(), с которой начинается выполнение программы. Остальные функции могут быть вызваны из main(), на рисунке 3.2 такой функцией является function(), которая ничего не возвращает.

**3.3 Бинарные и унарные арифметические операции**

В языке C++ существует несколько арифметических операций, которые можно использовать. Это сложение, вычитание, умножение, деление и остаток отделения – бинарные операции. Эти операции позволяют выполнять простые математические вычисления в программе: "+" – это сложение, "-" – вычитание, "\*" – умножение, "/" – деление, "%" – остаток от целочисленного деления.

Бинарные арифметические операции можно использовать совместно с операцией присваивания. Следующие операторы являются аналогами операторов, приведённых в комментариях (//):

1. object += выражение; // object = объект + выражение

2. object -= выражение; // object = объект – выражение

3. object \*= выражение; // object = объект \* выражение

4. object /= выражение; // object = объект / выражение

5. object %= выражение; // object = объект % выражение

Унарные операции выполняются над одним операндом. В языке C++ есть возможность использовать унарные операции такие как инкремент, декремент, отрицание и логической отрицание. Эти операции позволяют измениться значениям переменных на 1 или выполнять логические операции над ними:

++ (увеличение на 1)

-- (уменьшение на 1)

- (изменение знака)

Результат вычисления выражения, содержащего операции увеличения или уменьшения, зависит от того, где расположен знак операции – до или после переменной.

Если операция расположена перед переменной или выражением, то сначала значение переменной изменяется на 1, а затем это значение используется для выполнения следующих операций. Если одна из этих операций расположена после переменной, то сначала выполняется операция, а затем значение переменной изменяется на 1 [23].

int one = 5;

int two = 6;

int three = 0;

three = one\*++two; //three = 35, так как в операции умножения two = 7

three = one \* two++; //three = 30, так как в операции умножения two = 6

**3.4 Процесс построения программы**

Весь процесс состоит из нескольких этапов: препроцессорное преобразование, компиляция, линковка и выполнение.

*Препроцессорное преобразование*

*Директивы препроцессора:* Заголовочные файлы включаются в текст программы с помощью директивы препроцессора #include. Все директивы препроцессора начинаются со знака #, который должен быть самым первым символом строки.

*Препроцессор:* На этом этапе выполняется обработка директив препроцессора. Препроцессор выполняет такие задачи, как включение заголовочных файлов, макроопределения и условная компиляция. В результате получается единый исходный текст, который готовится для последующей компиляции.

*Компиляция:*

*Компилятор:* На этом этапе компилятор преобразует обработанный препроцессором исходный текст программы в машинный код. Этот процесс включает синтаксический анализ, семантический анализ и оптимизацию. Результатом компиляции является объектный файл (или несколько объектных файлов), который содержит машинный код, но не является еще исполняемым файлом.

*Линковка (компоновка):*

*Компоновщик (линкер):* На этом этапе из нескольких объектных файлов создается один исполняемый файл. Компоновщик связывает объектные файлы, разрешает внешние ссылки (например, вызовы функций из других файлов или библиотек) и добавляет необходимые системные библиотеки. В результате получается исполняемый файл, готовый к запуску.

*Выполнение:*

*Запуск программы:* Исполняемый файл загружается операционной системой в память и начинается его выполнение. Процесс выполнения управляется функцией main(), которая является точкой входа в программу.

Схема построения программы на алгоритмическом языке C++ представлен на рисунке 3.3.

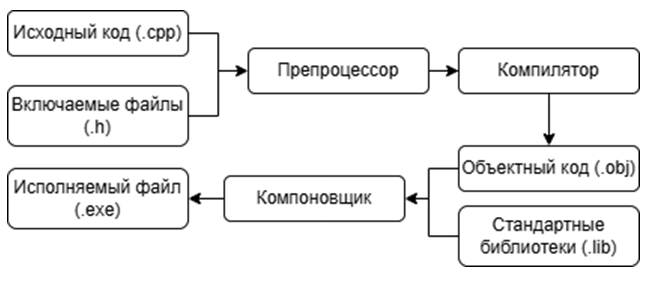


Рисунок 3.3 – Схема построения программы на языке C++

**3.5 Создание проекта в Visual Studio и отладка программы**

В среде разработки процесс построения программы упрощается. Среда разработки автоматически выполнит все необходимые шаги, включая компиляцию и линковку, и покажет результат.

Все начинается с открытия проекта. На главной странице необходимо выбрать "Создать проект". Из предложенных вариантов следует выбрать "Пустой проект".

Название проекта вводится в столбец "название".

После создания проекта нужно создать файл для хранения программного кода. Для этого нужно щелкнуть правой кнопкой мыши по названию проекта, выбрать "Добавить"/"Создать элемент". В появившемся окне необходимо выбрать "Файл C++" и присвоить ему имя. В этом открытом окне записывается программный код.

Если нажать "Локальный отладчик" или комбинацию клавиш после ввода CTRL+F5, то программа автоматически пройдет все три этапа отладки и запустится, если ошибок не будет. Существует также пошаговая отладка, которая позволяет шаг за шагом проходить программу, находя и исправляя ошибки и неточности. Разработчик может установить точки останова на нужных строках кода, а затем следить за изменениями в значениях переменных, выполнять код по шагам и анализировать его состояние. Это мощный инструмент для отладки и исправления ошибок.

Чтобы начать пошаговую отладку, необходимо нажать кнопку F10. Откроется консоль, в которой будут отображаться данные из программы. Отладка начинается с точки входа – main().

Желтый курсор слева указывает на строку, которая должна быть выполнена. Теперь, при нажатии кнопки F10 – шаг с вводом вызываемых функций – или F11 – шаг без ввода вызываемых функций, курсор будет опускаться вниз, тем самым выполняя программу постепенно, строка за строкой.

В нижней части окна есть три вкладки для отслеживания переменных: видимые переменные, локальные и контрольные значения.

Пример отслеживания значений переменных в программе: Вычисление суммы двух чисел и разницы между ними (рисунок 3.4).

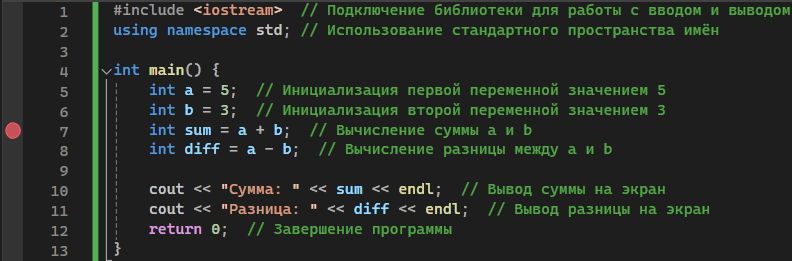


Рисунок 3.4 – Код программы отслеживания значений переменных

На строке номер 7 установлена точка останова, после запуска программы в режиме отладки программа будет остановлена на ней. На рисунке 3.5 приведены вкладки для отслеживания переменных, переменные a и b инициализированы, а переменные sum и diff – нет, так что имеют случайные значения.

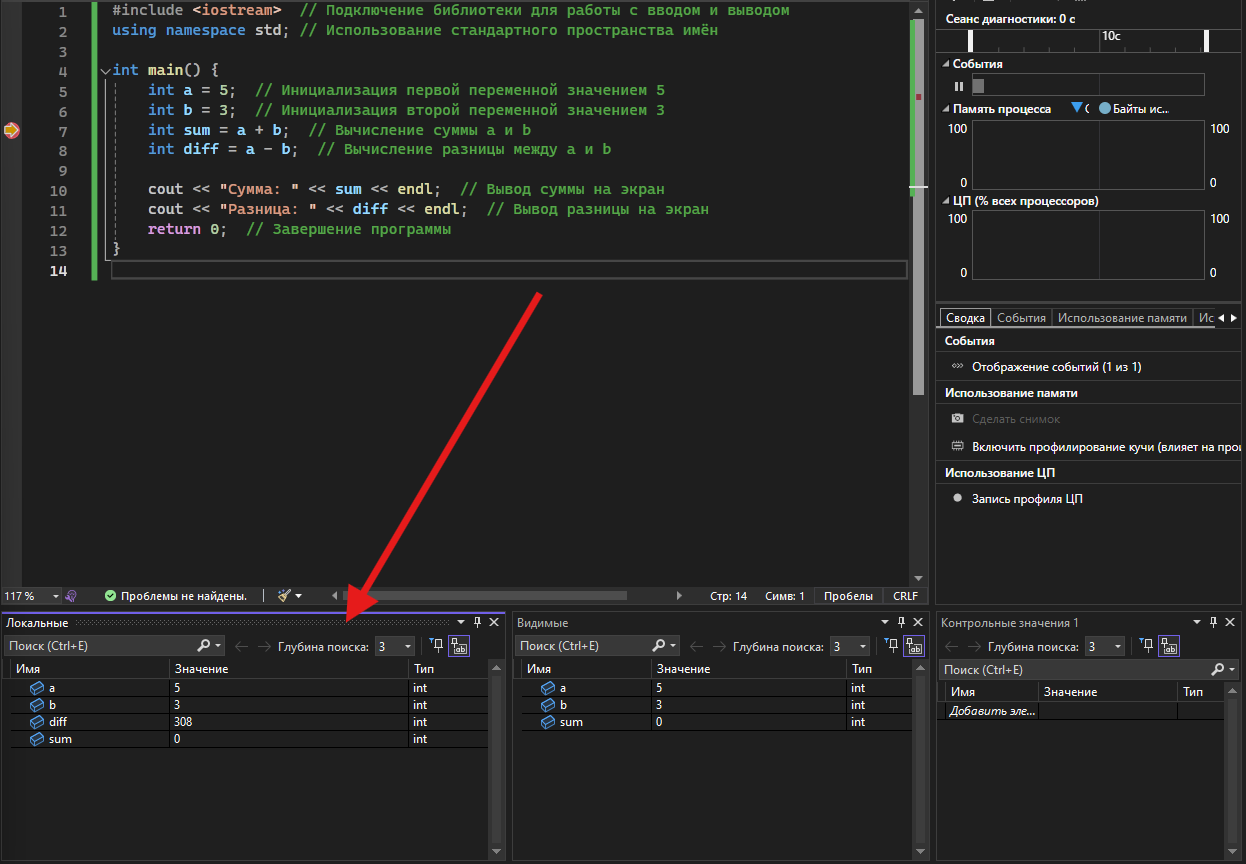


Рисунок 3.5 – Вкладки отслеживания переменных

После нажатия кнопки F10 или F11 переменная sum будет инициализирована и получит соответствующее значение (Рисунок 3.6).

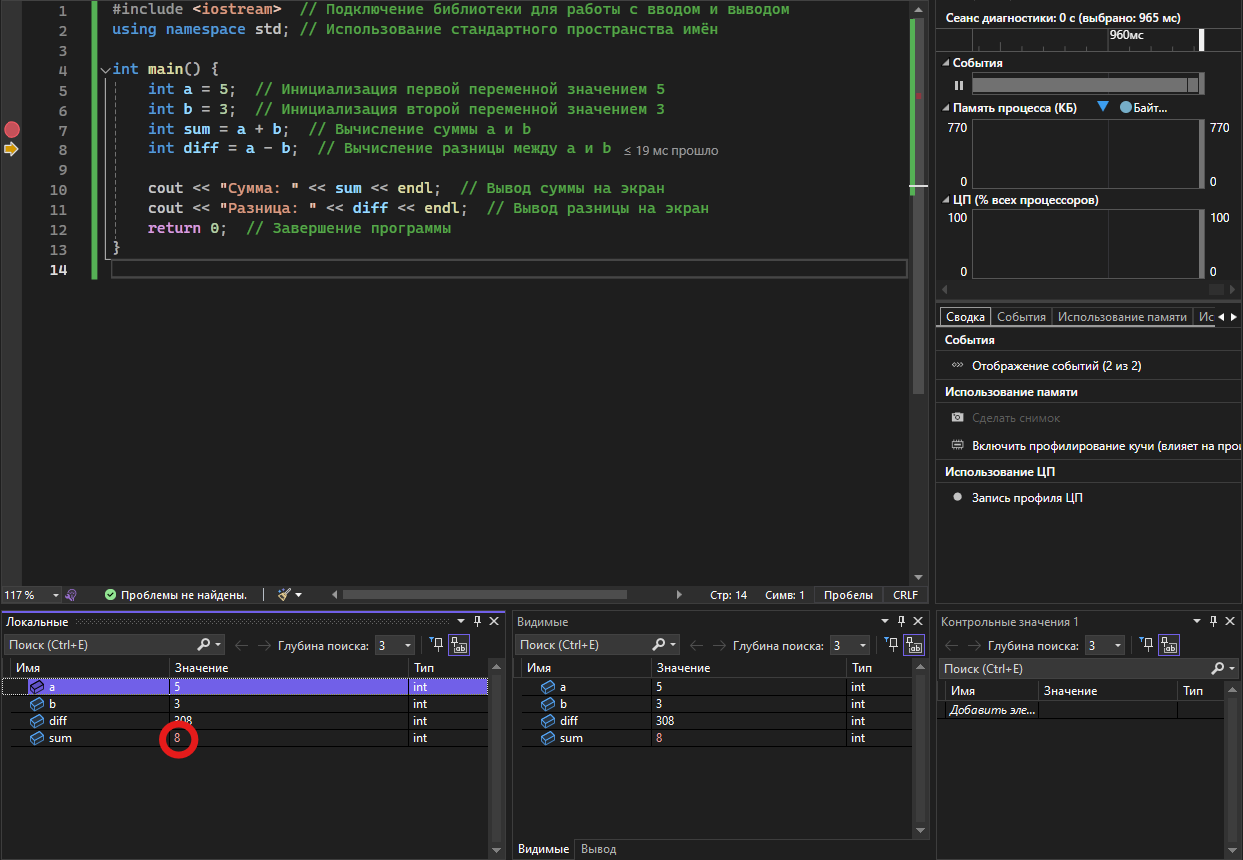


Рисунок 3.6 – Скриншот программы после выполнения одного шага

После выполнения программы (но перед выполнением строки return 0) все переменные инициализированы. Значение любой переменной может быть изменено, и это отобразится на вкладках для отслеживания переменных (рисунок 3.7).

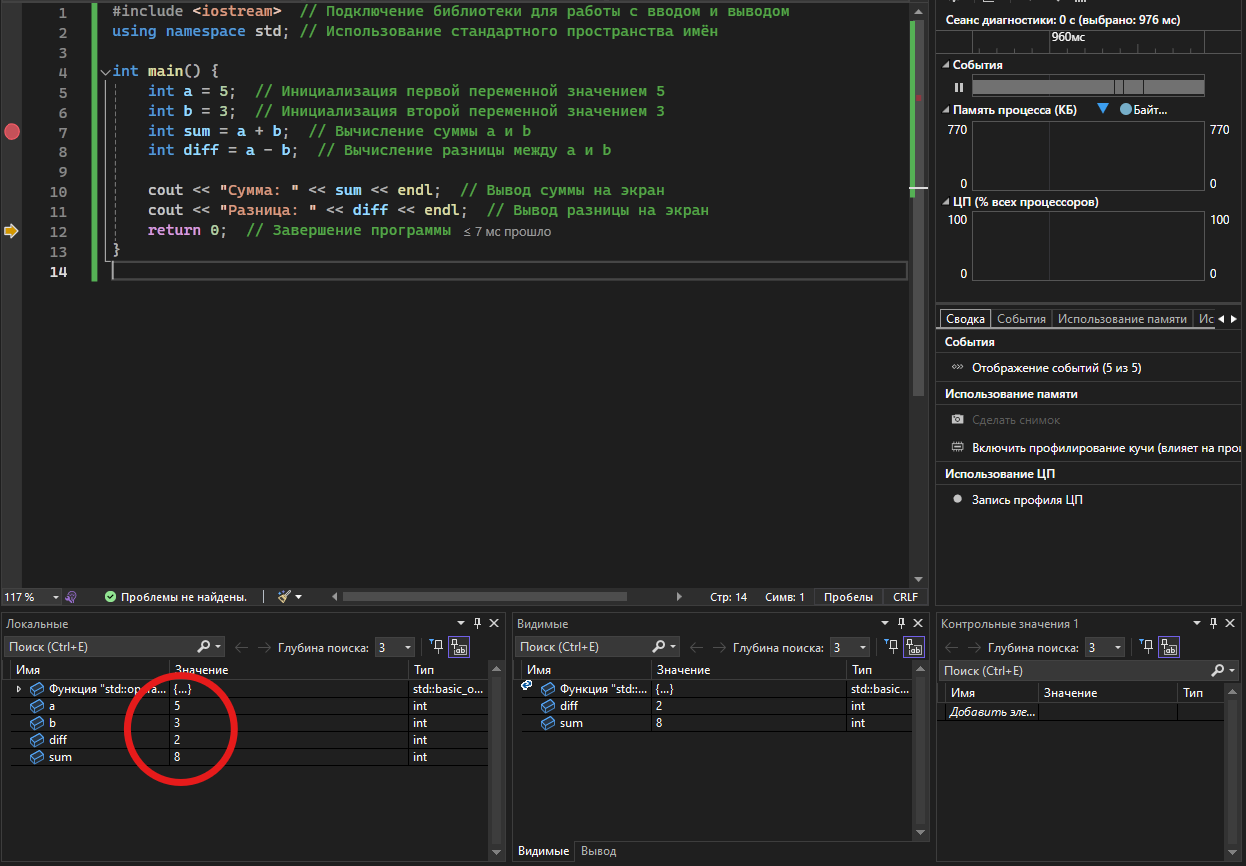


Рисунок 3.7 – Скриншот программы после её выполнения

Для закрепления материала необходимо решить практические задачи:

1. Разработать программу на C++, которая выводит сообщение

"Алгоритмические языки" в консоль.

2. Определить, чему будет равно выражение

i = ++i\*i++;

при начальном значении i = 1.

3. Найти ошибку в строке

int a=2.8;

int b=12;

4. Разработать программу на C++, которая считывает два целых числа с клавиатуры, складывает их и выводит результат на экран.

5. Найти результат выполнения следующего кода на языке C++:

int x = 5;

int y = 10;

int z = x++ + ++y;

Ответы к практическим задачам с пояснениями:

Решение задачи 1: Программа начинается с подключения библиотеки <iostream>, которая необходима для работы с функцией cout, выводящей данные в консоль. Функция main() является точкой входа в программу, и именно с нее начинается выполнение кода. Сообщение выводится с помощью оператора <<, а endl используется для переноса строки (рисунок 3.8).

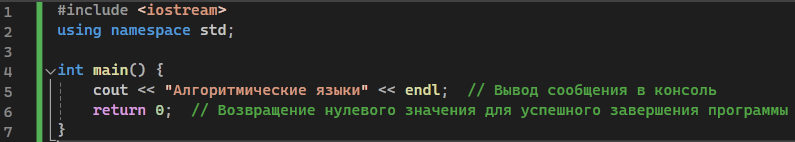


Рисунок 3.8 – Решение задачи 1

Решение задачи 2:

Для понимания результата необходимо учитывать, как работает инкремент в C++:

++i – это префиксный инкремент: сначала значение переменной увеличивается на 1, затем используется в выражении.

i++ – это постфиксный инкремент: сначала значение переменной используется в выражении, а затем увеличивается на 1. Начальное значение i=1. ++i увеличивает i до 2, затем это значение используется в выражении. После этого выполняется умножение: 2 \* 2 = 4. После использования i++ значение i увеличивается на 1 и становится равным 3. Таким образом, результат выполнения выражения будет i = 4.

Решение задачи 3:

Ошибка заключается в том, что тип int используется для переменной a, которая инициализируется значением 2.8. Тип int предназначен для целых чисел, и при присваивании дробного числа оно будет усечено до целой части, т.е. a станет равно 2, а не 2.8. Если требуется сохранить дробную часть, необходимо использовать тип float или double. В противном случае дробная часть будет потеряна при приведении типа.

float a = 2.8; // Или double a = 2.8;

intb = 12;

Решение задачи 4:

Программа начинается с объявления двух переменных num1 и num2, которые будут использоваться для хранения введенных пользователем чисел. cin используется для получения данных от пользователя. После ввода чисел программа вычисляет их сумму и выводит результат на экран с помощью cout (рисунок 3.9).

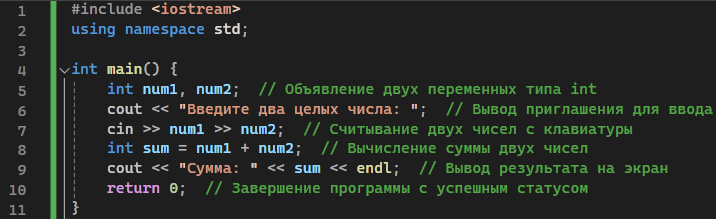


Рисунок 3.9 – Решение задачи 4

Решение задачи 5:

Изначальные значения: x = 5, y = 10.

x++ – постфиксный инкремент: сначала значение x используется в выражении, затем увеличивается на 1. В выражении участвует значение x=5, но после этого x становится равно 6.

++y – префиксный инкремент: сначала y увеличивается на 1, и затем используется в выражении. Значение y увеличивается до 11, и именно оно участвует в операции сложения.

В выражении z = x++ + ++y выполняется сложение: 5 + 11 = 16.

Таким образом, результат выполнения: x станет равен 6, y станет равен 11, z будет равно 16.

Ответ: после выполнения кода x = 6, y = 11, и z = 16.