Занятие 1.
Переменные, операторы, выражения.

Теория — это когда все известно, но ничего не работает.
Практика — это когда все работает, но никто не знает почему.
Мы же объединяем теорию и практику: ничего не работает…
и никто не знает почему

Альберт Эйнштейн

Приведем вариант классической первой программы. Она выводит на экран сообщение Hello, World!

```
// Эта программа выводит на экран сообщение “Hello,
world”!
#include <iostream>

using namespace std;

int main() {
    cout << “Hello, world!” << endl; // Вывод на экран
    return 0;
}
```

Рассмотрим что делает каждая строка программы.

```
cout << “Hello, world!” << endl;
```

Именно эта строка отвечает за вывод сообщения на экран.

Имя cout (произносится си-аут, character output stream) относится к стандартному потоку вывода. Поток вывода, например экран монитора (может быть и файл). Символы, “помещаются в поток cout” с помощью оператора <<.

Все, что написано после символа // (слэши) считается комментариями. Они полностью игнорируются компилятором и служат вспомогательными записями для программистов.

Комментарии могут нести полезную инфо для программистов, пояснять какие-то неочевидные места в коде, рекомендации и тд.

```
#include <iostream>
```

Представляет собой директиву препроцессора. Она заставляет “включить” функциональность которая реализована в другом файле (iostream.h). iostream.h

предоставляет доступ к стандартным возможностям ввода-вывода (ввод с клавиатуры, вывод на экран).
Расширение .h имеют заголовочные файлы (от слова header - заголовок).
По мере продвижения мы узнаем какие полезные функциональные возможности мы можем подключать и пользоваться.

Как компьютер узнает откуда начать выполнять программу? Он ищет функцию main и начинает выполнят инструкции содержащиеся в ней.

```
int main() {
    cout << “Hello, world!” << endl; // Вывод на экран
    return 0;
}
```

Каждая программа должна иметь начальную точку выполнение, то место откуда начнет выполняться программа. Именно такой начальной точкой исполнения и является функция main(). Все программы на языке C++ должны содержать функцию main() для того чтобы знать откуда начинать свое выполнение. По сути функция представляет собой последовательность действия, что за чем должна выполнять программа. Функция состоит из четырех частей:

1. *Тип возвращаемого значения.* Определяет тип значения которые вернет функция. В этой функции - тип int (от англ. integer - целое число). Когда функция закончит свою работу, она вернет значение целочисленного типа. int является зарезервированным словом, его нельзя использовать как имя функций или переменных.
2. Имя функции. В данном случае main.
3. Список параметров. Параметры перечисляются через запятую внутри круглых скобок. Пока мы не будем передавать никаких параметров в функцию и оставим только круглые скобки ().
4. Тело функции. Тело заключается в фигурные скобки {}. Внутри скобок перечисляются действия (инструкции), которые функция должна выполнить последовательно, друг за другом, сверху вниз.

Таким образом минимальная программа имеет вид:

```
void main() { }
```

Пользы от такой программы нет, так как она ничего не делает.

В программе Hello World тело функции состоит из двух инструкций:

```
cout << “Hello, world!” << endl; // Вывод на экран
return 0;
```

Сначала на экран выводится строка “Hello, world!”, а затем функция main() возвращает нулевое значение в точку вызова. Т.к программа вызывается операционной системой, то она ей вернет нулевое значение. В некоторых ОС можно проверить возвращаемое программой значение и судить от правильном

завершении программы. Как правило 0 возвращаемый программой означает о правильно завершении.

Арифметические операции

Сейчас мы научимся читать, писать и считать, как первоклашки, только на компьютере с использованием C++.

Сначала научимся считать. Мы уже знаем для того чтобы вывести на экран строчку существует команда cout. Эта команда может выводить не только строки, но и числа. Например, давайте попробуем вывести на экран результат арифметического выражения (2 + 3) * 5. Вот так будет выглядеть программа:

```
#include <iostream>

using namespace std;

int main() {
    cout << (2 + 3) * 5;
    return 0;
}
```

На экран выводится 25. Программа сначала посчитает результат выражения, а потом выведет его на экран.

В C++ существуют следующие арифметические выражения:

+ сложение
- вычитание
* умножение
/ деление
% вычисление остатка от деления

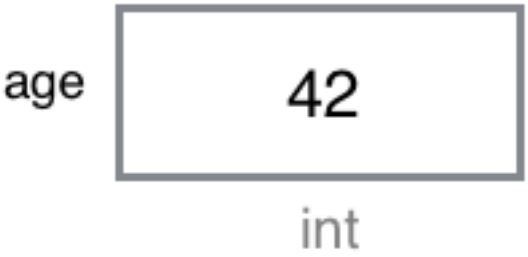
Приоритет у операций такой же как в математике, умножение и деление выполняются раньше сложения и вычитания. Влиять на порядок вычислений можно с помощью скобок.

Рассмотрим гораздо более интересную операцию взятия остатка от деления. С помощью операции деления можно получить целую часть от деления двух чисел. Например 7 / 3 равно 2. А с помощью взятия остатка от деления можно получить остаток от деления первого числа на второе. 7 % 3 будет равен 1. Если число делится нацело то остаток от деления будет равен нулю.

Программа Hello World просто выводит на экран строку, вторая наша программа вычисляет заранее заданное арифметическое выражение. Ничего более. Никаких данных от пользователя она не получает, никакого взаимодействия с пользователем тоже не проводит. Это скучно и не очень полезно.

Переменные
Программы в реальной жизни тесно взаимодействуют с пользователями, получают от них данные и выводят результат в зависимости от данных которые ввел пользователь.

Для того чтобы считать данные, необходимо место где эти данные хранить. Какое то место в памяти компьютера чтобы размерить туда то что мы прочитали. Назовем такое место объектом. Объект - это место в памяти которое имеет определенный тип, т.е какого вида информация может храниться в этом месте. Именованный объект называется переменной. Например, строку мы можем поместить в объект типа string, а целое число в объект типа int.
Переменную можно интерпретировать как коробку, которая имеет тип (тип определяет количество памяти, размер коробки), имя коробки и значение которое хранится в ней.



На рисунке изображен объект с именем age (англ. возраст) типа int (целочисленный) со значение 42. Для того чтобы считать строку с клавиатуры надо использовать переменную строкового типа string.

Давайте напишем программу, которая запрашивает имя пользователя и приветствует его:

```
#include <iostream>

int main() {
    cout << “Введите ваше имя (и нажмите enter): \n”;
    string name; // name – переменная типа string
    cin >> name; // Считываем символы с клавиатуры в
переменную name
    cout << “Привет, ” << name << “!\n”;
}
```

If control reaches the end of main without encountering a return statement, the effect is that of executing return 0;

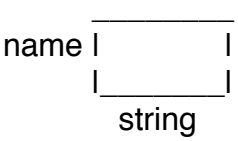
Давайте разберем что делает эта программа.
Директива #include подключает библиотеку iostream (input / output, ввод / вывод) в которой хранятся средства для ввода и вывода cin и cout.
Функция main() является начальной точкой исполнения программы.

```
cout << “Введите ваше имя (и нажмите enter): \n”;
```

Первая строчка функции main() просто выводит сообщение на экран предлагающее пользователю ввести свое имя. Такое сообщение называется приглашение, поскольку говорил что должен сделать пользователь.

```
string name; // name – переменная типа string
```

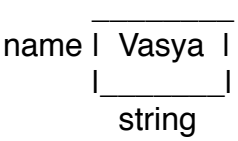
Вторая строчка выделяет участок памяти для хранения строки символов и присваивает этому участку памяти имя name.



Следующая строка считывает символы с клавиатуры и помещает в переменную.

```
cin >> name; // Считываем символы с клавиатуры в
переменную name
```

Команда cin обращается к стандартному потоку ввод (читается как “си-ин” от англ. character input), который определено в стандартной библиотеке iostream. Второй оператор >> (“получить из”) определяет участок памяти в который производится ввод. Т.е возьми символы которые вводятся с клавиатуры и положи их в участок памяти который выделился с именем name. Итак, если мы вреде имя, например Vasya и нашем клавишу <Enter>, то в переменную с именем name положится строка “Vasya”. Переменная name примет значение “Vasya”



Почему нажимаем именно клавишу <Enter?>. Потому что мы должны подать компьютеру знак о том что закончили ввод, таким знаком служит переход на новую строчку. При нажатии <Enter> выполняется переход на новую строчку, который служит знаком что пора закончить ввод.

```
cout << “Привет, ” << name << “!\n”;
```

Эта строка выводит на экран слово Привет, за ним следует переменная name, на ее место подставляется значение которое в ней хранится, а именно Vasya и завершает вывод восклицательный знак и символ перехода на новую строку.

Обратите внимание что что слово “Привет” заключается в двойные кавычки, но переменную name мы оставили без кавычек. В кавычки заключаются литеральные строки (литеральный, означает статический, неизменяемый). Если кавычки не указаны, то мы ссылаемся на имя (имя переменной). Рассмотрим пример:

```
cout << “name” << “ – ” << name;
```

“name” - это литеральная строка, просто неизменяемая последовательность символов, если вывести на экран name без кавычек, по на экран выводится значение которое хранится в переменной с именем name, в данном случае Vasya.
В итоге на экран выводится

name - Vasya

Переменные и типы

Тип переменной определяет какую информацию может хранить переменная.

Объявление (создание) переменной (выделение памяти, создание пустой коробочки) имеет следующий вид:

Тип переменной Имя переменной;

```
int x, y; // Объявление целочисленных переменных x и y
```

Слово int обозначает что переменные x и y будут иметь целочисленный тип, т.е смогут хранить только целые значения.
Для того чтобы присвоить значение переменной надо слева от равно написать имя переменной, а справа значение которые надо положить в переменную.

```
x = 2 + 3;
y = x * 5;
```

Присваивать значение можно одновременно с объявлением

Тип переменной Имя переменной = Значение;

```
int z = 10;
```

int - для целых чисел
Занимает 4 байта

```
int age = 21; // Объявление переменной целочисленного типа с именем age и присвоение значения 21
```

double - для чисел с плавающей точкой
(от англ. double precision floating point - число с плавающей точкой двойной точности)
Занимает 8 байт

```
double price = 29.81;
```

```
float weight = 55.7; // число с плавающей точкой с одинарной точностью
```

char - для символов

Занимает 1 байт

```
char first_letter = ‘a’;
char point = ‘.’;
```

string - для строк

```
string city = “Moscow”
```

bool - для логических переменных
Занимает 1 байт

```
bool turn_on = true;
bool even = false;
```

Обратите внимание, что каждый из типов имеет характерный способ записи

39 // int: целое число
3.5 // double: число с плавающей точкой
‘!’ // char: отдельный символ в одинарных кавычках
“Saint-Petersburg” // string: последовательность символов в двойных кавычках
true // book: либо истина, либо ложь

Рассмотрим еще пример. Считаем 2 целочисленные переменные с клавиатуры и выводим на экран сумму.

```
#include <iostream>

using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    cout << a + b;
    return 0;
}
```

Обратите внимание что при считывании и выводе на экран стрелки направлены в разные стороны >> <<.

Команда cout может выводить несколько значений за раз, давайте посмотрим пример

```
#include <iostream>

using namespace std;

int main() {
      int a, b;
      cin >> a >> b;
      cout << a << " + " << b << " = " << a + b << endl;
      cout << a << " - " << b << " = " << a - b;
      return 0;
}
```

Если ввести числа 1 и 2, то на экран будет выведено:
1 + 2 = 3
1 – 2 = -1

Попробуем решить более жизненную задачу:

Предположим, пассажир самолёта перепутал дверь туалета с выходом и случайно вышел на высоте. Приземлился он через t секунд, и нужно определить, на какой высоте летел самолёт.

Эту задачу можно решать так, как обычно решаются задачи по физике. Ускорение свободного падения нам известно (поскольку наши числа целые, то мы возьмём его равным 10). Мы посчитаем скорость (v), на которой пассажир достиг земли, затем среднюю скорость (vm, это конечная скорость, поделенная на 2) и, зная среднюю скорость и время, легко рассчитаем расстояние.

```
#include <iostream>

using namespace std;

int main() {
    int t, v, g = 10;
    cin >> t;
    v = g * t;
    int vm = v / 2;
    int s = vm * t;
    cout << s;
    return 0;
}
```

Рассмотрим, как решить совсем простую на первый взгляд задачу, которая превращается в достаточно сложную из-за того, что мы мало что умеем и знаем.

Задача формулируется так. Даны два числа a и b, причём b > 0. Надо посчитать целую часть от деления a на b, округлённую вверх. Напомню, что при делении C++ округляет результат вниз, не так, как нам нужно.

Первая идея — разделить с округлением вниз и прибавить к результату единицу. Эта идея неправильная: она не работает, если одно число делится на другое нацело. Так 8 / 2 + 1 будет равно 5, хотя правильный ответ 4.

Следующая идея, правильная, — прибавить к числу что-нибудь и затем разделить его с округлением вниз. Осталось понять, что же нужно прибавлять к числу. Если число a делится на b нацело, то результат не должен изменяться, значит, нельзя прибавлять к числу a что-либо большее b – 1 (если прибавить больше, то результат деления получится уже больше правильного). Можно ли прибавить что-нибудь меньшее b – 1? Рассмотрим «худший» случай, когда остаток от деления a на b равен единице, например, a = 11, b = 5. Тогда мы сложим a и b – 1 (получим 15) и разделим на 5 — получится правильный ответ 3.

```
#include <iostream>

using namespace std;

int main() {
	int a, b;
	cin >> a >> b;
	cout << (a + b - 1) / b;
	return 0;
}
```

Правила оформление кода

Программы нужно писать красиво, иначе их будет неудобно читать. На реальной работе программы много раз читаются и переписываются другими людьми, поэтому соблюдать правила оформления кода очень важно.

1. После открывающейся фигурной скобки добавляется отступ в начале строки, на строке с закрывающейся фигурной скобкой отступ убирается.
2. Все бинарные операции (+,–, *, /, %, =, <<, >>) окружаются пробелами.
3. После унарного минуса пробел не ставится (–5 нужно писать слитно).
4. Перед знаками препинания (запятая и точка с запятой) пробел не ставится, после — ставится.
5. После открывающейся и перед закрывающейся круглой скобкой пробел не ставится.
6. Если в условии задачи сказано «на вход даются два числа A и B», то переменные, в которые считываются эти числа, должны называться так же, но маленькими буквами (a и b соответственно).