C++ [1]

Платформа для тренировок

- Олимпиадные тренировки будут проходить в отдельной группе на платформе codeforces: `https://mai2020.contest.codeforces.com/`
- Для того, чтобы получить логин-пароль, нужно заполнить форму по ссылке: `https://forms.gle/wHddRcNaExd7eL7F6`

План на первые тренировки

- Тренировки 1-4 будут посвящены языку C++
- Затем будет серия контестов по основным темам олимпиадного программирования
- В конце тренировок первого семестра — осенняя олимпиада первого курса

Язык для олимпиадного программирования

В тренировках мы будем использовать **C++**, потому что он в рамках олимпиад в общем не сложный и достаточно мощный. Это не Pascal и не C, в которых отсутствует множество полезных инструментов. В C++ есть нормальная стандартная библиотека, при этом он не страдает проблемами со скоростью, как Python. Последний работает в среднем раз в 10 медленнее C++, а в олимпиадном программировании нужно писать быстрые программы.

На контестах вы можете сдавать задачи на любом языке, который поддерживает кодфорсес, но нет гарантий, что они будут проходить по времени на Java, Python или других "медленных"языках. Во втором семестре будет мало задач, которые будут проходить на Python. К концу первого семестра тоже будет сложновато.

Про платформу программирования

Я буду пользоваться текстовым редактором Emacs в Линуксе в связке с компилятором g++. Если у вас Виндовс, вы можете поставить CodeBlocks, CLion или Visual Studio. Если вы в Линуксе и при этом вам нравятся IDE, то CodeBlocks и Clion там также работают.

Для тех, кто использует Linux и хочет выбрать текстовый редактор – можете использовать Visual Studio Code. В нём легко открывается терминал, в целом удобно. Если кто-то хочет тонкой настройки под себя – выбирайте Vim или Emacs, но разбирайтесь с ними не во время контестов, а когда-нибудь потом, например, во время дорешки.

План на сегодня

Сегодня мы пройдёмся по базовым аспектам плюсов. Вам надо уметь определить в языке переменные, уметь считать эти переменные, выполнить какие-то простейшие операции с ними и вывести результат.

Как писать программы на C++

Программы на C++ можно называть любым образом, нужно указывать расширение **.cpp**, которое обычно используется для кода на нём.

Создание файла программы в консоли, используя emacs (сохранить C-x C-s, выйти C-x C-c).

```
pc@pc ~/P/T/T/Train> emacs program.cpp
```

Программы на C++ начинаются со специальной функции, в отличие от питона, в котором исполнение начинается прямо с верху файла, в C++ исполнение программы начинается со специальной функции **int main()**.

```
int main() {

}
```

Переменные и типы переменных

Что можно делать в рамках своей программы в элементарных случаях? Во-первых, можно объявить переменные, с которыми вы будете работать. Для объявления переменных вы пишете тип переменной и имя переменной. Полезно писать понятные названия переменных, но в простых программах можно обходиться короткими вроде x, y, a, b.

Немного про типы переменных. Тогда как в Python можно написать просто "$a = 4$", или что-то подобное, и получить, чему равно "a" , в C++ нужно обязательно указывать, какого типа переменную мы создаём.

Какие числовые типы полезны в рамках олимпиад?

- int – целое число от -2^{31} до $2^{31} - 1$, занимает 4 байта
- long long – целое число от -2^{63} до $2^{63} - 1$, занимает 8 байтов
- double – числа с плавающей запятой, не очень точно представляют то, что вы хотели бы написать. От $\pm\ 1.7 * 10^{-308}$ до $\pm\ 1.7 * 10^{308}$. Занимает 8 байтов

Стоит упомянуть, что при объявлении в C++ переменные могут перечисляться через запятую.

```
1 int a;
2 long long b, c, d;
3 double f;
```

Ввод-вывод в C++

Давайте напишем простую программу, которая складывает два числа. Для этого нужно объявить 2 переменные. Затем их нужно считать.

В C++ есть заголовочный файл для работы со стандартным вводом-выводом, в которой лежат нужные нам сейчас функции. Называется он iostream (input-output stream). Чтобы подключить его, нужно написать в начале программы #include <iostream>.

В этом заголовочном файле определены потоки стандартного ввода и вывода – **cin** (console input) и **cout** (console output). Чтобы считать число в переменную x из потока ввода, нужно писать "std::cin >> x;". std – это пространство имён, префикс, который добавляется ко всем функциям и классам стандартной библиотеки. Чтобы вывести число, пишете "std::cout << x;".

Точка с запятой указывает на завершение некоторого выражения или конструкции. В Python её место занимает перенос строки.

Функция main в C++ может возвращать вызвавшей её оболочке разные значения, принятно, что в случае успешного завершения она возвращает 0. Это осуществляется путём написания в конце (или в каком-то другом месте) этой функции оператора возврата с нулевым значением "return 0;".

Итак, наш код выглядит таким образом. Вы берёте, объявляете 2 переменных, говорите, считай мне в консольку первую, вторую переменную, а потом говорите, выведи мне в консольку сумму двух переменных.

```
#include <iostream>

int main() {
    int x, y;
    std::cin >> x >> y;
    std::cout << x + y;
    return 0;
}
```

Компиляция программы в Linux

Для компиляции написанной программы, у вас должен быть установлен один из компиляторов кода на C++, например g++ или clang++. Для компиляции нужно написать в консоли g++ и имя файла с кодом. После этого у вас в этой же папке создаётся исполняемый файл со стандартным именем a.out. Чтобы его запустить, пишете ./a.out.

```
pc@pc ~/P/T/T/Train> g++ program.cpp
pc@pc ~/P/T/T/Train> ./a.out
1 2
3pc@pc ~/P/T/T/Train>
```

Как видно в приведённом фрагменте, строка приглашения (prompt, в данном случае "pc@pc ~/P/T/T/Train>") выводится сразу после выведенного числа. Чтобы это исправить, можно добавить после вывода перенос строки. Одним из вариантов сделать это является вывод std::endl. В итоге строка с std::cout будет выглядеть так:

```
std::cout << x + y << std::endl;
```

Разделители в консольном вводе

Что нужно знать про консольный ввод. Он никак не смотрит на пробелы. Можно вводить переменные через несколько пробелов, через табуляции, переводить строки сколько угодно. Консольный ввод все эти разделители просто игнорирует, потому что они при чтении не нужны. Вы можете попросить считать число типа int, но при вводе задать ему букву, например "a". Язык не может распарсить букву как число и произойдёт что угодно. В рамках C++ основная философия заключается в следующем: что вы написали, то и будет использовано в процессе выполнения программы

Неинициализированные переменные

В нашей программе объявлены две переменные. Чему они изначально равны? Неизвестно, они не инициализируются при объявлении, как это делается в Java или C#. При объявлении переменным выдаётся определённая область в памяти и их значения становятся равными тому набору битов, который лежал в этой области раньше. Можно попытаться вывести эти переменные до чтения с консоли через пробел. Все символы при выводе окружаются одинарными кавычками, строки – двойными.

Как теперь выглядит код:

```
#include <iostream>

int main() {
    int x, y;
    std::cout << x << ' ' << y << std::endl;
    std::cin >> x >> y;
    std::cout << x + y << std::endl;
    return 0;
}
```

Можете его скомпилировать и запустить у себя. Если вам повезёт, мусор, который изначально лежит в переменных, может оказаться равным нулю.

Немного про стандартную библиотеку

Как уже говорилось, для использования функций из стандартной библиотеки, перед ними нужно добавлять перфикс "std::". Это нужно для того, чтобы избегать конфликта имён, когда у вас есть длинный код и много функций и классов, среди которых могут оказаться несколько с одинаковыми названиями.

В олимпиадном программировании такие ситуации бывают не так часто, поэтому код можно сократить и не писать эти префиксы, добавив в начале программы "using namespace std;". Таким образом, вы говорите, что далее в коде функции из пространства имён std можно записывать без префикса.

Арифметические операции в C++

Стандартные арифметические операции:

- Сложение (x + y), вычитание (x - y)
- Умножение (x * y)
- Деление (x / y)
  - если оба числа целые, ответом будет также целое значение, округлённое вниз (13 / 5 = 2)
  - если хотя бы одно число double, ответом будет double без каких-либо округлений до целого (5 / 3.14 = 1.59236)

Про выполнение операций с типом double

Из-за особенности представления чисел с плавающей запятой, результат выполнения операций над типом double может быть не таким, какой вы ожидаете.

Если кому-то интересно, как именно они представлены в компьютере, можете почитать статью с хабра: `https://habr.com/ru/post/112953/`, или найти другой источник. В целом тема полезная.

Как видно из результата выполнения кода ниже, double представляет числа не очень точно. По этой причине старайтесь заменять этот тип на целочисленный в большинстве случаев, когда это только возможно.

```cpp
#include <iostream>
using namespace std;

int main() {
    double x = 0.05 + 0.01;
    double y = 0.6 / 10;
    cout.precision(60);
    cout << x << ' ' << y << endl;
    return 0;
}
```

```
pc@pc ~/P/T/m/Lections> ./a.out
0.060000000000000004718447854656915296800434589385986328125
↪  0.059999999999999997779553950749686919152736663818359375
```

Операция взятия остатка

Не совсем стандартная операция – взятие остатка от деления (x % y). Она работает только для чисел целых типов. Примеры взятия остатка: 13 % 5 = 3, 5 % 2 = 1, 12 % 3 = 0. К сожалению, взятие остатка в C++ реализовано не по математическим правилам.

В математике остаток – всегда неотрицательное число, по абсолютной величине меньше делителя. У нас же при взятии, например, такого остатка: -7 % 4, ответом будет -3.

В связи с этим при взятии остатка от отрицательных чисел, необходимо использовать самописное взятие остатка.

Библиотека для работы с математическими функциями cmath

Оператора возведения в степень в C++, в отличие от Python, нет. Для того, чтобы посчитать степень, нужно подключить заголовочный файл cmath, добавив строчку #include <cmath> в начале программы. Функция, которая нам нужна, называется pow() (от англ. power). Первый аргумент данной функции – число, которое хотим возвести, второй – степень, в которую возводим. Так как все функции в cmath работают с типом double и возвращают значения этого типа, вы можете также возводить дробные числа в дробные степени.

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    int x, y;
    cin >> x >> y;
    cout << 1 / pow(x, y) << endl;
    return 0;
}
```

В библиотеке cmath также есть логарифмические, тригонометрические (кроме котангенса и арккотангенса) и обратные тригонометрические функции. Полный список при желании можно посмотреть здесь `http://www.cplusplus.com/reference/cmath/`.

Также из полезных сайтов для изучения возможностей C++ могу предложить cppreference.com. Хороший сайт.

Условный оператор if

Условный оператор выполняет или не выполняет определённые команды в зависимости от заданных условий. В C++ его синтаксис таков: слово "if обязательно круглые скобочки, в которых записывается условие и затем (без каких-либо двоеточий, как в питоне или "then как в паскале), выражение, которое будет выполняться, если условие будет истинным. Если нужно будет выполнить несколько команд, их нужно окружить фигурными скобками. Вообще это называется телом операторов.

Условиями могут быть сравнения чисел. Стандартные для C++ операторы сравнения:

- равно / не равно: ==, ! =
- строгие больше / меньше: >, <
- больше / меньше или равно: >= , <=

При сравнении на равенство **необходимо** ставить именно два равно. Если вы напишите одно, то это будет расценено компилятором как оператор присваивания.

Несколько условий можно комбинировать при помощи логических операторов:

- оператор "и": &&
- оператор "или": ||

Если мы хотим выполнять какие-то выражения также в случае невыполнения условия, после тела оператора "if"нужно написать "else"и затем написать эти выражения. Также можно предлагать несколько альтернатив, для этого нужно писать "else if (какое-то выражение)".

Примеры использования условного оператора:

```
// пример 1
if (a > b)
    cout << a << endl;
// пример 2
if (x > 0 && x + 1 / x >= 2) {
    cout << "Always\n"; // символ \n переводит на новую строку
}
else {
    cout << "Explain that\n";
}
// пример 3
if (a1 > 0 && a2 > 0 && a1 * a2 == 1 || a1 > 1 && a2 > 1) {
    int num = pow(2, 2);
    cout << "(" << a1 << " + 1) * (" << a2 << " + 1) > " num
    ↪  << endl;
}
// пример 4
if (a > 0) {
    cout << "positive\n";
}
else if (a < 0) {
    cout << "negative\n";
}
else {
    cout << "zero\n";
}
```

Сравнение чисел с плавающей запятой

Как было показано, числа типа double представляются в компьютере не точно. Кроме того, при большом количестве вычислений над ними, эта неточность увеличивается. Поэтому их сравнение не совсем тривиальная штука. В последующих тренировках это будет раскрыто. Пока же совет: при сравнении старайтесь как можно дольше обходиться целыми типами.

Функции в C++

Когда какая-то задача в коде решается несколько раз, её можно вынести в отдельную функцию. Как это можно сделать?

Для объявлении функции во-первых нужно сказать, что она возвращает, во-вторых, её название, а в-третьих указать в круглых скобках, какие у неё будут аргументы. Через запятую можно указывать несколько аргументов.

Пример функции, которая переводит градусы в радианы. Для этого нужно поделить угол на 180 и умножить на π.

Откуда взять число π? Во-первых, есть константа M_PI. Во-вторых можно посчитать его как арккосинус минус единицы.

В конце функции должен быть оператор return, в котором вы пишете, какое значение функция будет возвращать.

```cpp
#include <iostream>
using namespace std;

const double PI = acos(-1); // при попытки поменять, выведется ошибка

double to_radian(double phi) {
    return phi / 180 * PI;
}

int main() {
    double grad;
    cin >> grad;
    cout << cos(to_radian(phi)) << endl;
}
```

Итог

Надеюсь, после прочтения этой лекции вы сможете решить или дорешать все 10 задач первого контеста. Будут вопросы — пишите в чат телеги.