# Когда может быть удобно использовать язык C

- Написание операционных систем, драйверов.
- Написание гипервизоров и другого ПО для виртуализации (например, VirtualBox).
- Для робототехники и встраиваемых систем.
- Для алгоритмического трейдинга и любых систем реального времени.
- Для высокопроизводительных вычислений (например, математическое моделирование).
- Для компьютерной графики.
- Для высокопроизводительных систем управления базами данных.
- Для языковых виртуальных машин, интерпретаторов, Just In Time компиляторов, таких как JVM.
- Для программ, которые должны работать с очень небольшим количеством ресурсов: мало памяти, слабый процессор, требуется низкое энергопотребление.

Прежде чем мы начнём, несколько замечаний.

1. Язык C чувствителен к регистру. Например, имена `fun` и `Fun` различны.
2. Комментарии можно писать двумя способами:

```
/* Можно так, на несколько строчек

и пока не встретим звёздочку и косую черту */



x = 4 ; // а можно так -- от двух косых чёрт до конца строчки. Перед ним может быть код.
```

3. Пробелы и переносы строк можно ставить где хотим.
   Можно написать так:

```
int f() { return 42; }
```

   А можно так:

```
int f   (



)

 {

 return     42 ;
```

```
    }
```

4. Есть разные версии языка; основная линейка стандартов это C89, C99, C11, C18. Другие версии, обычно, являются расширениями стандартов, например, ядро Linux написано на GNU C, расширении стандарта C99. Мы будем пользоваться актуальной версией языка, C18; она, однако, практически не отличается от C11.
5. `<угловые скобки>` мы используем для описания мест в тексте программы, куда можно что-то вписать; в угловых скобках мы пишем пояснение того, что именно туда можно вписать. Конечно, при подстановке угловые скобки убираются. Например, мы можем написать `<число> + <число>`, и это значит, что возможно написать два числа и знак "+" между ними, скажем, `4 + 22`.
6. В первых модулях мы будем намеренно упрощать сложные понятия, чтобы не перегружать учеников и студентов. Наша цель – ясно передать вам идеи о самом важном (последовательное выполнение программы, функции, предложения и выражения, изменяемая память, экземпляры функций, рекурсия и т.д.). При этом мы будем во многом недоговаривать, или даже писать не совсем красиво и профессионально в угоду педагогичности. Обещаем, что как только мы изложим главные идеи, мы исправимся, будем писать красиво и правильно (как говорят ещё: *идиоматично*), и погрузимся в самые тонкие детали языка.
7. Одна из наших целей – чтобы вы писали красивый, легко читаемый и корректный код. Мы постараемся дать такой набор упражнений, который будет способствовать выработке у вас хороших привычек, насколько это возможно в рамках онлайн-курса.

# Распространённые ошибки

- Если в задании вас не просят выводить на экран ничего, ничего выводить нельзя.
- Если в задании просят написать функцию, напишите функцию, не нужно писать программу целиком.
- Не пишите `#include` и не определяйте функцию `main`, кроме как в заданиях, где эту функцию явно требуется определить.
- Компилятор можно настроить так, чтобы он выдавал больше или меньше ошибок или предупреждений. Предупреждения выдаются не просто так: они свидетельствуют о каких-то проблемах в вашем коде.
  В курсе компилятор настроен так, чтобы давать максимальное количество предупреждений и не пропускать программы с предупреждениями!
  Поэтому часто возникает ситуация: "у меня всё компилируется, а на степике нет". Это сделано намеренно! Внимательно прочитайте сообщение об ошибке и исправьте программу.
- Проходите задания по порядку, они связаны между собой!