История развития языка Си. Нововведения стандарта C23.

Этапы развития:

- **K&R C** – неформальный стандарт (1978)
- **ANSI C / ISO C (C89/C90)** – первый официальный стандарт (1989)
- **C99** – существенные улучшения (1999)
- **C11** – модернизация и безопасность (2011)
- **C17 (C18)** – исправления и уточнения (2017)
- **C23 (в разработке)** – дальнейшие улучшения (2023)

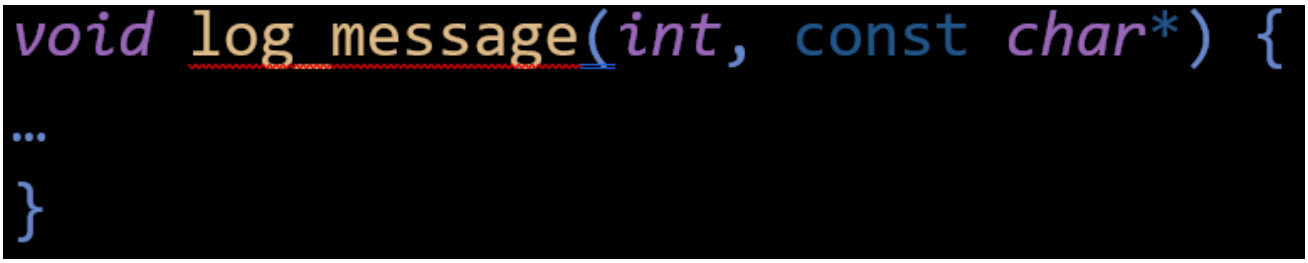
Краткая история

В 1969 году AT&T Bell Labs начинает разработку UNIX и Си.
В 1971 году компилятор языка Си уже включён в UNIX редакции 2.
В 1973 году ядро UNIX переписано на Си.
В 1978 году вышла книга от авторов языка — **The C Programming Language**. Это по сути и есть первый стандарт языка Си, который иногда называют K&R C (Керниган и Ричи)
Язык продолжает развиваться и потихоньку получает новые возможности, появляются компиляторы, которые часто не полностью совместимы друг с другом.
В 1983 году, к разработке подключается Американский национальный институт стандартов (нам известный как ANSI). Шесть лет комитет работал над стандартом ANSI C или C89.
Керниган и Ритчи В 1989 году выпускают вторую редакцию своей книги «The C Programming Language».
ISO активно работали над C90, его корректировках и C95, в 1999 году выпускает стандарт ISO/IEC 9899:1999 известный как C99. К тому времени ANSI ещё пытается сохранить лидерство и выпускает аналогичный стандарт в 2000 году.
Для C99 были выпущены три корректировки - в 2001, 2004 и 2007 годах.
С 2011 года стандарт C99 не поддерживается, т.к. выпущен стандарт C11.
Следующий стандарт языка Си выпущен в 2018 году. Тем не менее стандарт называется C17, хотя иногда ошибочно его называют C18. C17 принципиально новых возможностей не добавляет и содержит в основном исправление формулировок и неточностей C11.
Иногда оба стандарта объединяют под именем C11/C17.
Уже в 2016 году появилось неофициальное название следующего стандарта — C2x. Предположительно стандарт будет опубликован в 2020-х. В 2019 году состоялась первая встреча рабочей группы по языку Си посвящённая будущему стандарту. И спустя пять лет

и одну эпидемию COVID-19 C23 был наконец-то утверждён. Следующий стандарт языка Си ожидается относительно скоро, т.к. имеет неофициальное название C2y.

Нововведения C23

1. Метки
   Метка — это идентификатор, за которым следует двоеточие ':'
   Раньше в Си можно было ставить метки только перед операторами , например:
   

2. Неименованные параметры
   Это параметры функции, которые указаны в её объявлении или определении, но не имеют имени. Они нужны, когда важно указать тип параметра, но его значение игнорируется внутри тела функции.
   

3. Бинарные литералы
   Это способ записи целых чисел в двоичной системе счисления прямо в коде.
   Бинарный литерал начинается с 0b или 0B. Далее следуют цифры 0 и 1.
   `int mask = 0b101010;`
4. Разделители цифр
   Это символ подчёркивания "_" , который можно использовать внутри числовых литералов , чтобы улучшить читаемость больших чисел Разделитель игнорируется компилятором , то есть он не влияет на значение числа — это только для удобства

разработчика.

```
int     a = 1_234_567;
long    b = 0xFFFF_CCCC;
int     c = 0b1010_0000_1111;
float   d = 3.141_592_653_589_793f;
```

5. **typeof()**
   typeof() — это оператор времени компиляции , который возвращает тип выражения или имени типа. Это мощное и удобное расширение, которое позволяет получать тип переменной или выражения на этапе компиляции.

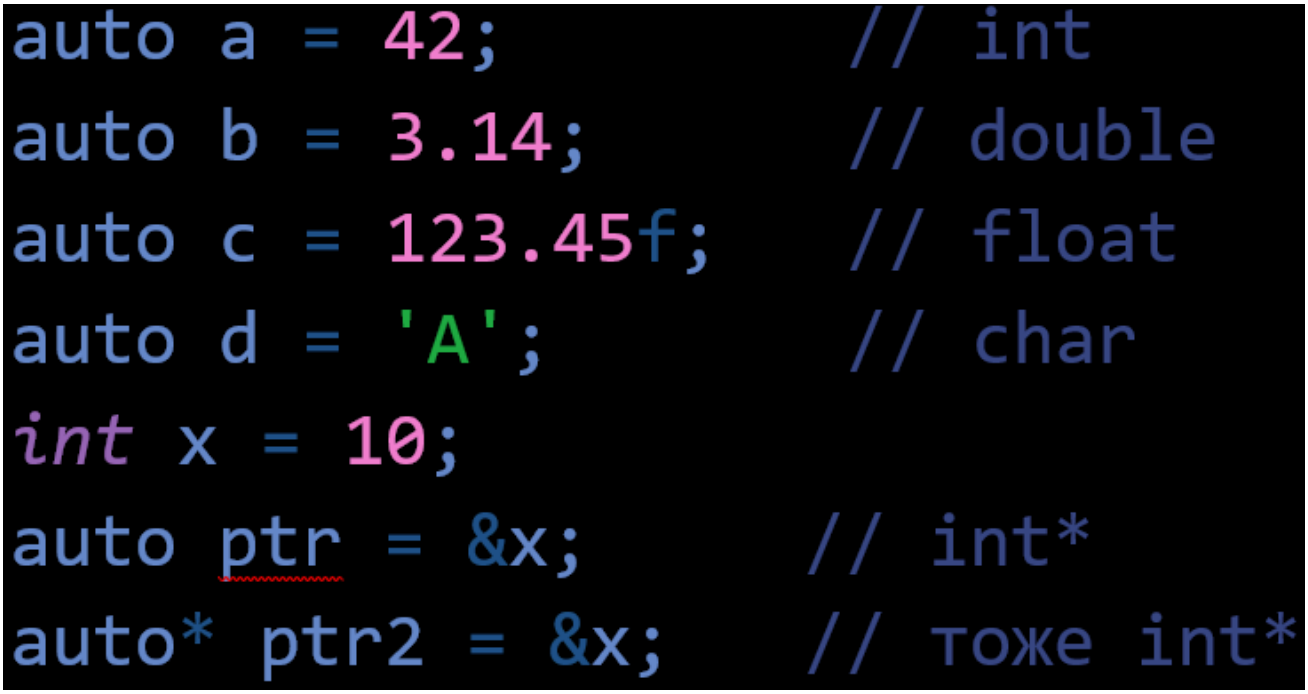
```
int a = 10;
typeof(a) b = 20; // EQ int b = 20;
typeof(int *) ptr; // EQ int *ptr;
typeof(int*) p1; // EQ: int *p1;
typeof(1 + 2.0f) result = 5.0f;
                  // EQ float result = 5.0f;
```

6. Пустая инициализация
   Добавлена возможность инициализировать переменные, массивы или структуры пустым списком инициализаторов, все поля или элементы будут проинициализированы нулевыми значениями.

```
int x = {};
int arr[10] = {};
```

7. Квалификатор **auto**
   Ключевое слово auto позволяет компилятору автоматически определять тип

переменной на основе значения, которым она инициализируется.



8. Спецификатор **constexpr**
   Это спецификатор гарантирует, что значение переменной или результат вызова функции будет вычислен на этапе компиляции , если это возможно, позволяет использовать константные выражения в местах, где требуется постоянство и предсказуемость, улучшает оптимизацию кода, обеспечивает безопасность и стабильность значений.
9. Ключевые слова **alignas, alignof, bool, true, false**
10. Типы **_Decimal32**, **_Decimal64** и **_Decimal128**