Лекция 28. Введение в современный язык Си.

Ревун Артем Леонидович
ст. преп. кафедры вычислительных систем

Курс «Программирование», весенний семестр, 2024
Сибирский государственный университет телекоммуникаций и информатики (г. Новосибирск)

Содержание

- Краткая история развития языка C
- Основные стандарты: от K&R до C23
- Новые функции стандарта C23

История языка Си (1)

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)

История языка Си (2)

1969 году AT&T Bell Labs начинает разработку UNIX и Си.

1971 году компилятор языка Си уже включён в UNIX редакции 2.

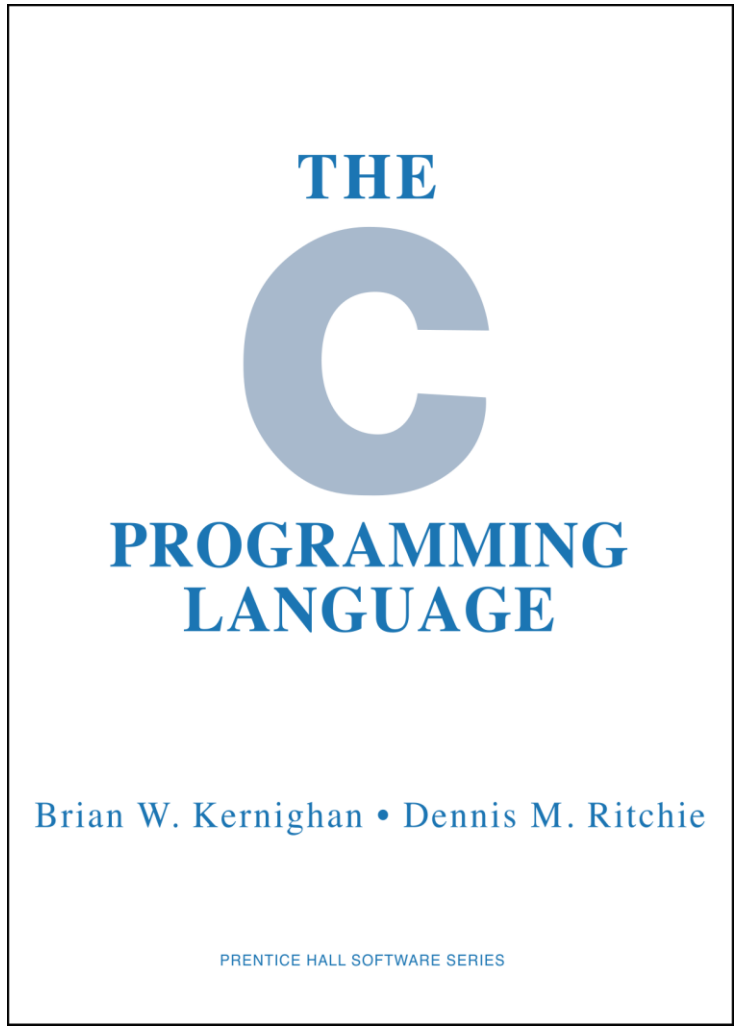
1973 году ядро UNIX переписано на Си.

Задача нового языка, разработанного Деннисом Ритчи и Брайан Уилсон Керниган — упрощение переноса ОС UNIX на другие платформы.

История языка Си (3)

Язык получился довольно простым в понимании, компактным и его компиляторы стали появляться для разных платформ.

В 1978 году вышла книга от авторов языка — **The C Programming Language**. Это по сути и есть первый стандарт де-факто языка Си, который иногда называют K&R C.



История языка Си (4)

- Язык Си продолжает набирать популярность писать программы на нём проще *чем на ассемблере*.
- *Чаще требуется написать только один раз*, а потом программу достаточно только скомпилировать на любой платформе, для которой есть компилятор языка Си.
- Язык продолжает развиваться и потихоньку получает новые возможности, появляются компиляторы, которые часто не полностью совместимы друг с другом. *Это проблема от которой уходили авторы*.

История языка Си (5)

- В1983 году, к разработке подключается Американский национальный институт стандартов (нам известный как ANSI) и решает, что пора возглавить разработку стандартов. И создаёт комитет для разработки спецификации Си. Шесть лет комитет работал над стандартом ANSI X3.159-1989, который увидел свет в 1989 году. Этот стандарт известен как ANSI C или C89.
- Керниган и Ритчи всё ещё очень активны в то время и в 1989 году выпускают вторую редакцию свой замечательной книги «The C Programming Language».

История языка Си (6)

Отличия между K&R C и ANSI C:

- прототипы функций
- добавлен квалификатор типа `volatile`, `const`, `void`, `signed`
- добавлен новый тип `long double`
- упразднён `long float`
- добавлен суффикс *U* (`unsigned int a = 123U;`)
- упразднены восьмеричные 8 и 9 (`int x = 09;`)
- добавлено многоточие ... для вариативных функций
- инкремент/декремент `+=` и `-=` вместо `=+` и `=-`

История языка Си (7)

- 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 году — ISO/IEC 9899:1999/Cor 1:2001(E), в 2004 году — ISO/IEC 9899:1999/Cor 2:2004(E), а в 2007 году — ISO/IEC 9899:1999/Cor 3:2007(E), которая примечательна тем, что в ней функция `gets()` объявляется устаревшей (и вовсе удаляется уже в следующем стандарте C11).
- С 2011 года стандарт C99 больше не поддерживается ни ANSI, ни ISO, т.к. выпущен новый стандарт C11.

История языка Си (8)

Отличия C99 от ANSI C

- новые типы — `long long`, `_Bool`, `_Complex`
- встраиваемые функции `inline`
- объявление переменных не ограничено началом блока
- массивы переменной длины (VLA)
- квалификатор `restrict`
- поддержка однострочных комментариев //
- составные константы: `calculate((struct point){ 4, 2 })`
- инициализаторы массивов и структур:вариативные макросы:
  `#define eprintf(...) fprintf (stderr, __VA_ARGS__)`

История языка Си (9)

- В том же самом 2007 году, в котором выходит Cor 3 для C99, начинается работа над новым стандартом, который получил условное обозначение C1x. Это значит, что публикация стандарта ожидается в 2010-х годах. И ожидания не подвели, в 2011 году выходит стандарт ISO/IEC 9899:2011 или C11. Для этого стандарта была выпущена одна корректировка — ISO/IEC 9899:2011/Cor 1:2012.
- Следующий стандарт языка Си выпущен в 2018 году — ISO/IEC 9899:2018. 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. Среди прочего в C17 однозначно разрешается вызывать функцию realloc() с нулевым размером памяти, *но в C23 это будет изменено.*

История языка Си (10)

Отличия C11 и C17 от C99

- удалена функция `gets()`
- выравнивание данных: `_Alignas`, `_Alignof`, `aligned_alloc()`, `<stdalign.h>`
- спецификатор `_Noreturn`
- выражения, не зависящие от типа: `_Generic`
- анонимные структуры и объединения:

```
struct T { int tag; union { float x;
int n; }; };
```

История языка Си (11)

Отличия C11 и C17 от C99

- статические утверждения (`_Static_assert`)
- привилегированный режим `"x"` для `fopen()`
- функция `quick_exit(), at_quick_exit();`
- выборочные возможности, вводятся макросы:

Анализирование	__STDC_ANALYZABLE__	Недоступно
Действия с дробными числами по стандарту IEC 60559	__STDC_IEC_559__	Выборочно
Арифметика комплексных чисел, совместимая со стандартом IEC 60559	__STDC_IEC_559_COMPLEX__	Выборочно
Интерфейсы проверки границ массива	__STDC_LIB_EXT1__	Недоступно
Типы комплексных чисел (<complex.h>)	__STDC_NO_COMPLEX__	Обязательно
Многопоточное программирование (<threads.h>)	__STDC_NO_THREADS__	Недоступно
Атомарные операции (<stdatomic.h> и квалификатор типа _Atomic)	__STDC_NO_ATOMICS__	Недоступно
Массивы переменной длины	__STDC_NO_VLA__	Обязательно

История языка Си (12)

Отличия C11 и C17 от C99

- Улучшенная поддержка Unicode, основанная на техническом отчете C Unicode Technical Report ISO/IEC TR 19769:2004 (типы char16_t и char32_t для хранения данных в кодировках UTF-16/UTF-32, функции преобразования, находящиеся в заголовочном файле <uchar.h> и соответствующие префиксы u и U перед строковыми литералами, как и префикс u8 для строк в кодировке UTF-8);

История языка Си (13)

Отличия C11 и C17 от C99

- Интерфейсы для проверки границ массива `strcpy_s, memcpy_s, scanf_s` **(не нашли реализации в GCC)**
- Возможности анализирования (пример, для IDLE)
- Добавлено больше макросов для получения характеристик чисел с плавающей точкой, касающихся денормализованных чисел и максимального числа десятичных цифр, которые можно хранить без потери точности;(пример, <limits.h>)
- Макросы для создания комплексных чисел (были добавлены потому, что код real + imaginary*I мог не привести к ожидаемому значению, если мнимая часть была бесконечной или «не числом» (NaN).

История языка Си (14)

```
void f1(void) {puts("pushed first");   quick_exit
fflush(stdout);}
void f2(void){ puts("pushed second"); }
void f3(void){ puts("won't be called"); }
int main(void) {
    at_quick_exit(f1);
    at_quick_exit(f2);
    atexit(f3);
    quick_exit(0); }
```

```
pushed second
pushed first
```

История языка Си (15)

```
#define cbrt(X) _Generic((X), \                    _Generic
                        long double: cbrtl, \
                        default: cbrt, \
                        float: cbrtf)(X)

float       a = 27.0f;
double      b = 27.0;
long double c = 27.0L;
printf("cbrt(float): %f\n",       cbrt(a));
printf("cbrt(double): %f\n",      cbrt(b));
printf("cbrt(long double): %Lf\n", cbrt(c));
```

История языка Си (16)

Unicode

```
const char *utf8_str=u8"Привет";        // UTF-8
const char16_t *utf16_str=u"Привет";    // UTF-16
const char32_t *utf32_str=U"Привет";    // UTF-32
//UTF-8 Многобайтовая кодировка переменной длины, совместима с ASCII
//UTF-16 Кодировка с использованием 16-битных кодовых единиц
//UTF-32 Фиксированная 32-битная кодировка каждого символа Unicode
```

История языка Си (17)

Static_assert

```
_Static_assert(sizeof(int) == 4, \
               "int должен быть 4 байта");
```

История языка Си (18)

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 стандарт ISO/IEC 9899:2024 или C23 был наконец-то утверждён. Следующий стандарт языка Си ожидается относительно скоро, т.к. имеет неофициальное название C2y.

Отличия C23 от C11 и C17 (1)

Удалено из стандарта:

- обязательная поддержка чисел с дополнительным кодом
- определение функций в стиле K&R
- вызов *realloc()* с 0 теперь неопределённое поведение ~~(ха-ха)~~

Нововведения языка:

- директивы препроцессора #embed, #elifdef, #elifndef, #warning
- типы _Decimal32, _Decimal64 и _Decimal128
- атрибуты в стиле C++11:

Отличия C23 от C11 и C17 (2)

Нововведения языка:

- [[nodiscard]], [[maybe_unused]], [[deprecated]], [[fallthrough]], [[noreturn]], [[reproducible]], [[unsequenced]]
- метки могут появляться до объявлений и в конце выражений
- неименованные параметры в объявлении функций:
- int f(int, int) { return 7; }
- бинарные литералы: 0b10101010
- разделители цифр: 0xFF’FF’FF’FF
- оператор typeof()

Отличия C23 от C11 и C17 (3)

Нововведения языка:

- пустая инициализация с помощью {} (включая VLA)
- ключевое слово auto для объявления переменных
- спецификатор constexpr
- ключевое слово static_assert вместо _Static_assert
- ключевое слово thread_local вместо _Thread_local
- ключевые слова alignas, alignof, bool, true, false

Отличия C23 от C11 и C17 (4)

Изменения в стандартной библиотеке:

- новые заголовочные файлы <stdbit.h> и <stdckdint.h>
- некоторые POSIX функции становятся стандартными:
- memccpy(), strdup() (наконец-то) и strndup()
- gmtime_r(), localtime_r()
- расширения fscanf() и fprintf():
- спецификатор %b для вывода бинарных чисел
- H, D, DD для _Decimal32, _Decimal64 и _Decimal128

Нововведения стандарта (1)

`#embed` — встраивание содержимого файла

Позволяет встроить содержимое внешнего файла (например, бинарного или текстового) как массив байтов (unsigned char[]) прямо в программу.

```
unsigned char logo_data[] = {
    #embed "logo.png"
};
```

Нововведения стандарта (2)

`#elifdef` — условная компиляция по наличию макроса. Сокращает запись конструкции `#else #if defined(...)`.

Используется для проверки, определён ли макрос в рамках одной директивы.

```
#ifdef DEBUG
    // debug mode
#elifdef RELEASE
    // release mode
#else
    // default mode
#endif
```

```
#ifdef DEBUG

#else #ifdef RELEASE

#else

#endif   #endif
```

Нововведения стандарта (3)

`#elifndef` – условная компиляция по отсутствию макроса

Проверяет, не определён ли макрос , и выполняет блок кода, если это так.

```
#ifdef LINUX
    // Linux-specific code
#elifndef WINDOWS
    // fallback for non-Windows systems
#else
    // Windows-specific code
#endif
```

Нововведения стандарта (4)

`#warning` — вывод предупреждения на этапе компиляции.

Выводит пользовательское сообщение предупреждения при компиляции программы.

```
#if defined(USE_OLD_API)
    #warning "Вы используете устаревший API. Обновите код."
#endif
```

Нововведения стандарта (5)

Эти типы реализуют десятичную арифметику , которая особенно важна в таких областях, как финансы, бухгалтерия, научные вычисления , где двоичная арифметика с плавающей точкой (например, float, double) может приводить к округлению и потере точности .

_Decimal32	32 бита	~7	$\pm 10^{1999999}$	IEEE 754-2008
_Decimal64	64 бита	~16	$\pm 10^{1999999}$	IEEE 754-2008
_Decimal128	128 бит	~34	$\pm 10^{19999999999999999}$	IEEE 754-2008

Нововведения стандарта (6)

```
_Decimal32 price = 19.99df;   // суффикс df – для _Decimal32
_Decimal64 amount = 12345.67d;
// суффикс d тоже подходит
_Decimal128 big_number = 12345678901234567890123456789012 34D;
```

Нововведения стандарта (7)

[[noreturn]]	Функция никогда не возвращает управление
[[deprecated]]	Объект или функция устарела
[[deprecated("reason")]]	То же, но с указанием причины
[[carries_dependency]]	Используется для передачи зависимости между потоками
[[fallthrough]]	Явно указывает, что break отсутствует в case
[[maybe_unused]]	Переменная или функция может быть не использована
[[nodiscard]]	Результат функции не должен игнорироваться
[[unlikely]] / [[likely]]	Подсказки компилятору о вероятности выполнения ветки

Нововведения стандарта (8)

[[noreturn]]	Функция никогда не возвращает управление
[[deprecated]]	Объект или функция устарела
[[deprecated("reason")]]	То же, но с указанием причины
[[carries_dependency]]	Используется для передачи зависимости между потоками
[[fallthrough]]	Явно указывает, что break отсутствует в case
[[maybe_unused]]	Переменная или функция может быть не использована
[[nodiscard]]	Результат функции не должен игнорироваться
[[unlikely]] / [[likely]]	Подсказки компилятору о вероятности выполнения ветки

Нововведения стандарта (8)

[[noreturn]] — функция не возвращает управление

```
#include <stdlib.h>
[[noreturn]] void fatal_error(
const  char* msg) {
    fprintf(stderr, "%s\n", msg);
    exit(EXIT_FAILURE);
}
```

Нововведения стандарта (9)

[[deprecated]] — устаревший код

```
[[deprecated("Используйте new_api() вместо этой функции")]]
void old_api(void) {
    // ...
}
```

Нововведения стандарта (10)

[[nodiscard]] — результат функции нельзя игнорировать

```
[[nodiscard]] int compute_result(void);
int main(void) {
    compute_result(); // Компилятор может выдать предупреждение
    return 0;}
```

Нововведения стандарта (11)

[[maybe_unused]] — переменная может не использоваться

```
void debug_info([[maybe_unused]] const char* info) {
    // В режиме релиза info может не использоваться
    #ifdef DEBUG
        printf("Debug: %s\n", info);
    #endif }
```

Нововведения стандарта (11)

[[fallthrough]] — намеренный переход в switch-case

```
switch (value) {
    case 1:
        do_something();
        [[fallthrough]];  // явно указываем, что это не ошибка
    case 2:
        do_something_else();
        break;
}
```

Нововведения стандарта (11)

[[likely]] / [[unlikely]] — подсказки компилятору

```
if (likely(condition)) {
    // Ожидается, что условие истинно
} else [[unlikely]] {
    // Редкий случай
}
```

Спасибо за внимание
Продолжение следует…

Ревун Артем Леонидович
ст. преп. кафедры вычислительных систем

Курс «Программирование», весенний семестр, 2024
Сибирский государственный университет телекоммуникаций и информатики (г. Новосибирск)