От автора

Автор Вики - @zhikhkirill. Обратная связь (в том числе и по ошибкам) приветствуется.

Создано в рамках проекта "Аполлон".

НЕ окончательная версия. Дата последнего обновления: 5:00 26 июня 2024.

А

- Адрес
- Алфавит
- Аппаратный стек
- Арифметическое преобразование типа
- Ассемблер
- Ассемблирование

Б

- Байт
- Библиотека
- Бинарная операция
- Бинарный файл

В

- Вещественный тип данных
- Видимость
- Возвращаемое значение
- Время выполнения
- Время жизни
- Время компоновки
- Время препроцессирования
- Время трансляции
- Вызов подпрограммы
- Выравнивание
- Выражение

Д

- Данные
- Динамическая библиотека
- Динамический массив
- Директива препроцессора

Е

- Единица трансляции

З

- Заголовочный файл
- Зарезервированное ключевое слово
- Защита заголовочного файла
- Значащий разряд

И

- Идентификатор
- Интерпретатор
- Интерпретация
- Интерфейс
- Исходный код

К

- Класс памяти
- Ключевое слово
- Компилятор
- Компиляция
- Компоновка
- Компоновщик
- Компьютер
- Компьютерная программа
- Константа
- Константный указатель

Л

- Лексема
- Лексика
- Лексикографический порядок
- Ленивая логическая схема
- Ленивые вычисления
- Литерал
- Логический тип данных
- Логическое выражение
- Логическое значение

М

- Макрос
- Массив
- Массив переменной длины
- Матрица
- Машинно зависимое поведение
- Машинное слово
- Машинный язык
- Многомерный массив
- Модуль

Н

- Неопределенное поведение
- Неспецифицированное поведение
- Неявное преобразование типа
- Нулевой адрес
- Нулевой указатель

О

- Область видимости
- Объединение
- Объектный файл
- Объявление
- Одномерное адресное пространство
- Одномерный массив
- Операнд
- Оператор
- Оператор break
- Оператор continue
- Оператор do while
- Оператор for
- Оператор goto
- Оператор return
- Оператор typedef
- Оператор while
- Оператор выбора

- Оператор выражение
- Операция
- Операция sizeof

П

- Память
- Передача параметра по значению
- Передача параметра по ссылке
- Переменная
- Переменная объединенного типа
- Переменная структурного типа
- Перечисляемый тип данных
- Подпрограмма
- Поле структуры
- Порядок байтов
- Предопределенное имя
- Преобразование типа
- Препроцессирование
- Препроцессор
- Пробельный символ
- Простой тип данных
- Процедура
- Пунктуатор
- Пустой оператор

Р

- Размер
- Ранк целочисленного типа
- Расширение файла
- Регистр процессора

С

- Связывание
- Семантика
- Сигнатура подпрограммы
- Символьный тип данных
- Синтаксис
- Система компиляции
- Слово
- Соглашение о вызове
- Составной оператор
- Составной тип данных
- Спецификатор формата
- Стандартная библиотека
- Статическая библиотека
- Статический массив
- Стековый кадр
- Строка
- Строковый литерал
- Структура

Т

- Тег объединения
- Тег структуры
- Текстовый файл
- Тело подпрограммы
- Терминальный нуль
- Тип ptrdiff t
- Тип size t
- Тип данных
- Точка входа

- Транслятор
- Трансляция
- Требование к выравниванию

У

- Указатель
- Указатель на константу
- Унарная операция
- Упаковка
- Управляющая последовательность
- Условный оператор

Ф

- Файл
- Файловая система
- Фактический параметр
- Формальный параметр
- Функция

Ц

- Целочисленный тип данных
- Целочисленный тип фиксированной длины

Я

- Явное преобразование типа
- Язык ассемблера
- Язык программирования

Адрес

Адресом называется уникальное беззнаковое целое число, которым характеризуется элемент в одномерном адресном пространстве.

Алфавит

Алфавитом называется конечное непустое множество символов языка.

Алфавит языка С включает:

- латинские буквы `[a-zA-Z]`;
- цифры `[0-9]`;
- 29 графических символов `!` `"` `#` `%` `&` `'` `(` `)` `*` `+` и др.;
- пробельные символы.

Аппаратный стек

Аппаратным стеком называется непрерывная область памяти, используемая программой для вызова подпрограмм.

Использование аппаратного стека

1. Вызов функции
   - поместить адрес подпрограммы;
   - передача управления по адресу.
2. Возврат из функции
   - извлечь из стека адрес возврата подпрограммы;
   - передать управление по адресу.
3. Передача аргументов в подпрограмму
   - Используется соглашение о вызове.

Арифметическое преобразование типа

Арифметическим преобразованием типа в языке С называется неявное преобразование типа операнда арифметической или логической операции.

Правила арифметического преобразования типа

1. Если один из операндов `long double`, то второй приводится к `long double`.

```
long double a;
double b;
a + b;  // long double
```

2. Иначе, если один из операндов `double`, то второй приводится к `double`.

```
double a;
int b;
a + b;  // double
```

3. Иначе, если один из операндов `float`, то второй приводится к `float`.

```
float a;
long long b;
a + b;  // float
```

4. Иначе, происходит преобразование целочисленных операндов:

4.1. Если операнды имеют одинакового типа, преобразование не требуется.

```
int a;
int b;
a + b;  // int
```

4.2. Иначе, если оба операнда знаковые, то операнд с меньшим ранком приводится к типу операнда с большим ранком.

```
long a;
int b;
a + b;  // long
```

4.3. Иначе, если оба операнда беззнаковые, то операнд с меньшим ранком приводится к типу операнда с большим ранком.

```
unsigned long long a;
unsigned int b;
a + b;  // unsigned long long
```

4.4. Иначе, если ранк беззнакового операнда больше или равен ранку знакового операнда, то знаковый операнд приводится к типу беззнакового операнда.

```
unsigned long a;
int b;
a + b;  // unsigned long
```

4.5. Иначе, если знаковый операнд может представлять все значения беззнакового операнда, то беззнаковый операнд приводится к типу знакового операнда.

```
long a;
unsigned short b;
a + b;  // long
```

4.6. Иначе, оба операнда приводятся к беззнаковому типу, способному представить все значения знакового операнда.

```
int a;
unsigned long b;
a + b;  // unsigned long
```

Арифметические операции с целочисленными операндами приводит операнды ранком меньше `int` (`unsigned int`) к ранку последних.

```
char a, b;
a + b;  // int
```

```
unsigned short a, b;
a + b;  // unsigned int
```

> Важно! Пункт 4.6 может привести к неочевидным результатам:

```
int a = -1;
unsigned int b = 1;
a < b;  // Ложь!
```

> Это происходит потому, что согласно пункту 4.6 знаковый операнд приводится к беззнаковому.

Преобразование отрицательного знакового в беззнаковый целочисленный тип равного ранка

Преобразование отрицательного знакового целочисленного типа в беззнаковый равного ранка происходит приравниванием содержимых ячеек. Согласно хранению отрицательных чисел знак «минус» хранится в старшем бите старшего байта. Поэтому отрицательное знаковое число в беззнаковововом представлении имеет вид `значение + UMAX + 1`, где `UMAX` — максимальное число в беззнаковом представлении.

Ассемблер

Ассемблером называется транслятор, выполняющий ассемблирование.

Ассемблирование

Ассемблированием называется трансляция исходного кода, записанного на языке ассемблера, на машинный язык. Является частным случаем компиляции.

Байт

Байтом или минимальной единицей адресации называется минимальная разница между двумя различными адресами.

Библиотека

Библиотекой называется совокупность объектного файла, не имеющих точки входа, и его заголовочного файла.

Библиотеки бывают:

- статические;
- динамические;

Бинарная операция

Бинарной операцией называется операция с двумя операндами и возвращающая один результат.

Бинарный файл

Бинарным файлом называется файл, не являющийся текстовым.

Вещественный тип данных

Вещественным типом данных называется простой тип данных, множество значений которого является подмножеством вещественных чисел R, а также *положительная бесконечность* (`+inf`), *отрицательная бесконечность* (`-inf`) и `NaN`.

В языке C представлены следующие вещественные типы данных:

Тип	Пояснение	Значащих разрядов*	Спецификатор	Суффикс
`float`	`IEEE-754` `single-precision binary floating-point format`	6 - 8	`[fgeaFGEA]`	`[fF]`
`double`	`IEEE-754` `double-precision binary floating-point format`	13 - 16	`l[fgeaFGEA]`	
`long double`	`IEEE-754` `? precision binary floating-point format`		`L[fgeaFGEA]`	`[lL]`

> Комментарии:
>
> - * значащие разряды.
> - ** в качестве `long double` может использоваться различная точность выше `double`.

Видимость

Видимостью идентификатора называется явление, при котором идентификатор может быть использован.

> См. также области видимости.

Возвращаемое значение

Возвращаемым значением в языке C называется значение выражения вызова подпрограммы. Тип возвращаемого значения подпрограммы определяется сигнатурой подпрограммы.

Возврат значения из функции осуществляется оператором return.

Пример:

```c
int sum(int a, int b)
{
    return a + b;
}

...

int a = 5, b = 6;
sum(a, b);
// sum(a, b) - вызов функции;
// 11        - возвращаемое значение;
// int       - тип возвращаемого значения.
```

Время выполнения

Временем выполнения или runtime называется интервал времени, в течение которого происходит выполнение программы.

Время жизни

Временем жизни или lifetime идентификатора в языке C называют часть времени выполнения, в течении которого сущность *гарантировано* хранится в памяти.

В языке C выделяют следующие времена жизни:

- статическое, `static` (глобальное);
- автоматическое, `auto` (локальное);

- выделенное, `allocated` (динамическое);

Статическое время жизни

Статическое время жизни эквивалентно времени выполнения.

Автоматическое время жизни

Автоматическое время жизни соответствует времени выполнения блока составной оператор.

Динамическое время жизни

> Динамическое время жизни рассматривается в следующем семестре.

Динамическое время жизни определяется началом оператором `new` и заканчивается оператором `delete`.

Время компоновки

Временем компоновки называется интервал времени, в течение которого происходит компоновка программы.

Время препроцессирования

Временем препроцессирования называется интервал времени, в течение которого происходит препроцессирование программы.

Время трансляции

Временем трансляции называется интервал времени, в течение которого происходит трансляция программы.

Вызов подпрограммы

Вызовом подпрограммы в языке C называется выражение, содержащее функтор, идентификатор подпрограммы и перечисление аргументов.

Формальная запись вызова подпрограммы в языке C:

```
подпрограмма()
```

```
подпрограмма(параметр_1, ... параметр_N)
```

Вызов подпрограммы на машине осуществляется посредством механизма стекового кадра.

Выравнивание

Выравниванием в языке C называется явление расположения переменной в памяти по адресу, кратному одному или нескольким байтам.

Выравнивание необходимо для более эффективного доступа к данным в памяти.

> Противопоставлено явлению упаковке.

Выражение

Выражением в языке C (`expression`) называется последовательность операторов и операндов, которая указывает:

- как вычислить значение,
- обозначает объект или подпрограмму,
- порождает побочный эффект,
- или комбинация вышеперечисленного.

Данные

Данными называется представление информации в формализованном виде.

Динамическая библиотека

Динамической библиотекой называется библиотека, связываемая с программой во время выполнения.

Особенности использования динамических библиотек:

- несколько программ могут использовать одну библиотеку.
- имеет меньший размер, чем статическая библиотека;
- обновление библиотеки не требует пересборки проекта.
- динамическую библиотеку может использоваться в программах, написанных на разных языках программирования.
- исполняемый файл требует наличие динамической библиотеки.

Динамический массив

Динамическим массивом в языке C называется массив, время жизни которого является динамическим.

> Динамические массивы будут изучаться в 2 части курса «Программирования на C».

Директива препроцессора

Директивой препроцессора называется строка в исходном коде, предназначенная для выполнения препроцессором, в формате:

```
#ключевое_слово параметры
```

- символ `#`;
- ноль или более символов пробелов и/или табуляции;
- одно из *ключевых слов препроцессора*;
- параметры, зависимые от ключевого слова.

Список ключевых слов:

- `define` — создание макроса;
- `undef` — удаление макроса;
- `include` — вставка содержимого указанного файла;
- `if` — проверка истинности логического выражения;
- `ifdef` — проверка существования макроса;
- `ifndef` — проверка несуществования макроса;
- `else` — ветка условной компиляции при ложности выражения `if`;
- `elif` — проверка истинности другого выражения; краткая форма записи для комбинации `else` и `if`;
- `endif` — конец ветки условной компиляции;
- `line` — указание имени файла и номера текущей строки для компилятора;
- `error` — вывод сообщения и остановка препроцессирования с ненулевым кодом возврата;
- `warning` — вывод сообщения без остановки препроцессирования;
- `pragma` — указание действия, зависящего от реализации, для препроцессора или компилятора;

Единица трансляции

Единицей трансляции называется файл, получаемый в результате препроцессирования.

Заголовочный файл

Заголовочным файлом называется текстовый файл, содержащий макросы препроцессора и объявления на языке C.

Обычно имеют расширение `.h`.

В заголовочный файл принято включать защиту от повтороного включения.

Пример заголовочного файла:

```
#ifndef A_H
#define A_H

extern FILE *fd;

typedef int type_t;

void foo(int a);

#endif // A_H
```

> См. также стандартная библиотека.

Зарезервированное ключевое слово

Зарезервированным ключевым словом называется ключевое слово, использование которого запрещено правилами синтаксиса в качестве идентификатора, определяемого пользователем.

Защита заголовочного файла

Защитой заголовочного файла или защита от повторного включения или include guard в языке C называется конструкция, предназначенная для защиты от повторного включения заголовочного файла.

Обычно защита от повторного включения имеет вид:

```
#ifndef ФАЙЛ_H
#define ФАЙЛ_H

содержимое...

#endif
```

> Защита заголовочного файла может быть также реализована при помощи директивы `pragma once`.

Значащий разряд

Значащим разрядом называется разряд числа, не равный нулю или нуль между значащих разрядов.

Пример:

```
0123.004560
```

Здесь первый нуль — незначащий, 1, 2, 3, 4, 5, 6 — значащие, второй и третий нуль значащие, последний нуль — незначащий. Всего 8 значащих разрядов.

Идентификатор

Идентификатором или именем называется слово, используемое для идентификации сущности.

Идентификатор в языке C

Идентификатор в языке C может содержать только:

- символы латиницы `[a-zA-Z]`, регистр имеет значение;
- цифры *не в начале слова* `[0-9]`;
- символ нижнего подчеркивания `_`.

Регулярное выражение, описывающее идентификатор языка Си:

```
[a-zA-Z_][0-9a-zA-Z_]*
```

Идентификатор в языке C характеризуется:

- областью видимости;
- временем жизни;
- связывание.

Перечисленные параметры управляются классами памяти.

Интерпретатор

Интерпретатором называется транслятор, выполняющий интерпретацию.

Интерпретация

Интерпретацией называется выполнение исходного кода на отличном языке от машинного.

Интерфейс

Интерфейсом называется набор правил, определяющих взаимодействие с объектом.

Интерфейс в языке C

В языке C интерфейсом модуля называется заголовочный файл.

Исходный код

Исходным кодом называется текст, записанный на каком-либо языке программирования.

Класс памяти

Классом памяти в языке C для идентификатора называется совокупность времени жизни области видимости и связывания.

В языке C выделяют следующие классы памяти:

- `auto`;
- `static`;
- `extern`;
- `register`.

Класс памяти идентификатора определяется соответствующим ключевым словом.

Класс памяти auto

Доступен только для переменных внутри блока.

По умолчанию идентификатор имеет класс памяти `auto`.

Класс	Время жизни	Область видимости	Связывание
`auto`	Автоматическое	Блок	Отсутствует

Класс памяти static

Для переменной, определенной внутри блока.

Класс	Время жизни	Область видимости	Связывание
`static`	Статическое	Блок	Отсутствует

```
// a.c
static int i = 0;

void foo(void)
{
    i++;
}

// b.c
#include "a.h"

void bar(void)
{
    i++;    // ERROR
}
```

Для идентификаторов, определенных вне блока.

Класс	Время жизни	Область видимости	Связывание
`static`	Статическое	Файл	Внутреннее

```
int foo(void)
{
    static int i;   // Инициализируется нулем автоматически!
    i++;
    return i;
}

...

foo();  // 1
foo();  // 2
```

Класс памяти extern

По умолчанию подпрограммы имеют *внешнее* связывание.

Класс	Время жизни	Область видимости	Связывание
`extern`	Статическое	Файл	Внешнее/внутреннее*

* Связывание определяется предыдущим объявлением идентификатора.

Если предыдущий идентификатор имел связывание...

- внутреннее, то связывание внутреннее;
- внешнее, то связывание внешнее;
- отсутствующее, то связывание внешнее.

```
// a.h
extern int a;

// b.c
#include "a.h"

a++;
```

Класс памяти register

Рекомендация компилятору расположить переменную в регистре процессора.

Доступен только для переменных внутри блока и формальных параметров подпрограммы.

Класс	Время жизни	Область видимости	Связывание
`register`	Автоматическое	Блок	Отсутствует

Ключевое слово

Ключевым словом называется идентификатор, смысл которого зафиксирован правилами языка программирования.

К ключевым словам языка C относятся:

auto	continue	extern	int	signed	union	_Complex
break	default	float	long	sizeof	unsigned	_Imaginary
case	do	for	register	static	void	
char	double	goto	restrict	struct	volatile	
const	else	if	return	switch	while	
continue	enum	inline	short	typedef	_Bool	

Компилятор

Компилятором называется транслятор, выполняющий компиляцию.

Следует различать с системой компиляции.

Компиляция

Компиляцией называется трансляция исходного кода на машинный язык.

Компоновка

Компоновкой называется процесс преобразования оттранслированных модулей в единую программу.

В процессе компоновки компоновщик:

- объединяет несколько объектных файлов в единый исполняемый файл;
- выполняет связывание переменных и функций;
- добавляет специальный код, который подготавливает окружение для вызова функции main, а после ее завершения выполняет обратные действия.

Компоновщик

Компоновщиком называется программа, выполняющая компоновку.

Компоновщик в GNU Linux — `The GNU linker` (`ld`).

Компьютер

Компьютером будем называть вычислительную машину, состоящую из трех составляющих:

- центральный процессор (ЦПУ или `CPU`);
- оперативное запоминающее устройство (ОЗУ или `RAM`);
- система ввода-вывода (`IO`).

Так же считаем, что отдельные составляющие общаются между собой машинными словами.

Компьютерная программа

Компьютерной программой называется комбинация инструкций, позволяющая компьютеру выполнять вычисления или функции управления.

Константа

Константой (литерал) называется неизменяемая величина.

> См. также
>
> - безымянные константы.

Константы в языке C

В языке C константой называется выражение, результат которого известен на момент-трансляции и неизменяемое во время выполнения.

В языке C константой называется величина или выражение, значение которой остается неизменным во время выполнения программы.

В языке C различают:

- именованная константа — константа, связаннная с идентификатором.
- безымянная константа — литерал.

Константный указатель

Константным указателем в языке C называется неизменяемый указатель.

Константный указатель является константой.

Формальная запись константного указателя:

```
тип *const идентификатор;
```

Пример:

```
a = 5;
b = 6;

int *const p = &a;
*p = 7;     // Корректно.
p = &b;     // Некорректно.
```

> Следует отличать от указателя на константу.

Лексема

Лексемой называется единица словарного запаса языка. Лексема может быть представлена в виде пары «тип: значение». Например, слово `1234` может быть представлено в виде лексемы «число: 1234».

Лексемы языка C

К лексемам языка C относятся:

- идентификаторы;
- ключевые слова;
- литерал;
- пунктуаторы (знаки пунктуации).

Правило определения лексем в языке C

> Каждый непробельный символ добавляется к считаемому токену до тех пор, пока он не станет корректным.

Пример:

```
a+++b
```

1. `a` — начало токена;
2. `+` — `a+` не является корректным токеном, поэтому начинается новый токен — `a` `+`;
3. `a` `++` — `++` является корректным токеном — операция инкремента;
4. `a` `+++` — `+++` не является корректным токеном, поэтому начинается новый токен `a` `++` `+`;
5. `a` `++` `+b` — `+b` не является корректным токеном, поэтому начинается новый токен `a` `++` `+` `b`.

Получаем выражение:

```
a++ + b
```

Лексика

Лексикой называется совокупность лексем языка.

Лексикографический порядок

Лексикографическим порядком строк называется порядок сортировки строк по следующим правилам:

1. Если первые `m` символов строк `a` и `b` совпадают, а `m + 1` символ строки `a` больше `m + 1` символа строки `b`, то строка `b` предшествует строке `b`.
2. Если строка `a` является началом строки `b`, то строка `a` предшествует строке `b`.

> Лексикографический порядок строк соответствует порядку «как в словаре».

Пример:

```
abc < abd
abc > ab
```

Ленивая логическая схема

Ленивой логической схемой в языке С называется ленивое вычисление результата логического выражения.

Для ленивой логической схемы характерно:

- если левый операнд для операции логического "И" ложен, то правый операнд не будет вычислен и результатом выражения будет ложь.

```
false && f();   // false
```

- если левый операнд для операции логического "ИЛИ" истинен, то правый операнд не будет вычислен и результатом выражения будет истина.

```
true || f();    // true
```

Ленивые вычисления

Ленивыми вычислениями называется стратегия вычисления значения выражения, при которой вычисления откладываются до тех пор, пока не понадобится их результат.

Литерал

Литералом называется безымянная константа.

Литералы в языке С

Пример литералов в языке С:

```
#define MAGIC_NUMBER 42

int main()
{
    printf("The Answer to the Ultimate Question of Life, the Universe, and Everything is %d",
MAGIC_NUMBER);
    return 0;
}
```

Здесь, литералами являются:

- строка 1, `42` — целочисленный литерал.
- строка 5, `The Answer ... is %d` — строковый литерал.
- строка 6, `0` — целочисленный литерал.

> Замечание: именованная константа может инициализироваться литералом.

```
const char * const s = "lorem ispum";
// s - константа;
// "lorem ispum\0" - строковый литерал.
```

Логический тип данных

Логическим типом данных называется простой тип данных, способный хранить только два значения: `0` (`false`, ложь) и `1` (`true`, истина).

Логический тип данных в C

В C логический тип данных является подтипом целочисленного типа данных. Обозначается как `_Bool`. Вводится заголовочным файлом `stdbool.h`.

> См. также
>
> - ленивая логическая схема.

Логическое выражение

Логическим выражением называется выражение, результатом которого является логическое значение.

Логическое значение

Логическим значением называется значение *истинно* (`true`) или значение *ложно* (`false`).

Макрос

Макросом препроцессора называется символьное имя, заменяемое препроцессором.

Макросы могут иметь параметры. В соответствии с этим свойством делятся на:

- макросы-как-объекты — не имеют параметров;
- макросы-как-функции — имеют параметры;

Макросы вводятся при помощи директивы `define`.

Примеры:

```
#define PI 3.14                              // Макрос-как-объект
#define max(a, b) ((a) : (b) ? (a) > (b))    // Макрос-как-функция
```

Макросы, содержащие не литерал, рекомендуется описывать внутри `(` `)`, чтобы сохранить порядок вычислений.

Пример:

```
#define STR_MAX_LEN 255

#define STR_BUF_SIZE_1 STR_MAX_LEN + 1
#define STR_BUF_SIZE_2 (STR_MAX_LEN + 1)

...

char buf1[STR_BUF_SIZE_1 * 2];  // 255 + 1 * 2
                                // 255 + 2
                                // 257  ???
char buf2[STR_BUF_SIZE_2 * 2]   // (255 + 1) * 2
                                // (256) * 2
                                // 512  OK
```

Массив переменной длины

Массивом переменной длины или VLA в языке C называется массив, размер которого **не** известен на момент трансляции и имеющий автоматическое время жизни.

Пример VLA:

```c
int main(void)
{
    int n = 5;
    int a[n];   // VLA
}
```

Массив

Массивом называется агрегированный тип данных, обладающий следующими свойствами:

- *Линейность:* элементы массива в памяти расположены последовательно.
- *Однородность:* все элементы массива одного типа данных.
- *Произвольный доступ:* время доступа к любому элементу массива не зависит от его позиции.

Различают следующие виды массивов в зависимости от размерности:

- одномерный массив;
- многомерный массив;

В языке C

В языке C массивы представлены в виде:

- статических массивов;
- динамических массивов;
- массивов переменной длины (VLA).

> См. также:
>
> - size_t;

Матрица

Матрицей называется таблица чисел.

Матрица в языке C

В языке C матрица размером может быть представлена следующими способами:

- многомерный массив;
- одномерный массив.

Двумерная матрица в языке C

Двумерная матрица в языке C является частным случаем матрицы в языке C. Двумерная матрица может быть представлена в виде:

- многомерного массива **строк** матрицы;

```c
// 1 2 3 4
// 5 6 7 8
// 9 0 1 2
int matrix[3][4] = {
    { 1, 2, 3, 4 },
    { 5, 6, 7, 8 },
    { 9, 0, 1, 2 },
};
matrix[1][2]; // 7
```

- многомерного массива **столбцов** матрицы;

```
// 1 2 3 4
// 5 6 7 8
// 9 0 1 2
int matrix[4][3] = {
    { 1, 5, 9 },
    { 2, 6, 0 },
    { 3, 7, 1 },
    { 4, 8, 2 },
};
matrix[2][1]; // 7
```

- одномерного массива с разложением по **строкам**;

```
// 1 2 3 4
// 5 6 7 8
// 9 0 1 2
int matrix[3 * 4] = {
    1, 2, 3, 4,
    5, 6, 7, 8,
    9, 0, 1, 2,
};
matrix[1*4 + 2]; // 7
```

- одномерного массива с разложением по **столбцам**;

```
// 1 2 3 4
// 5 6 7 8
// 9 0 1 2
int matrix[4 * 3] = {
    1, 5, 9,
    2, 6, 0,
    3, 7, 1,
    4, 8, 2,
};
matrix[2*3 + 1]; // 7
```

Машинно зависимое поведение

Машинно-зависимым поведением или implementation-defined behavior в языке C называется неспецифицированное поведение поведение, явно определяемое системой компиляции.

Например, знак результата операции `%`.

```
int x = 5;
int y = -2;

x % y > 0;  // ?
```

Машинное слово

Машинным словом будем называть сообщение фиксированной длины, которым общаются отдельные состовляющие компьютера.

Машинный язык

Машинным языком называется система команд конкретной вычислительной машины.

Многомерный массив

Многомерным массивом называется массив, элементами которого являются массивы.

> См. также многомерные массивы в языке C.

Модуль

Модулем называется логически законченная часть исходного кода, обычно оформленная в виде отдельного файла/нескольких файлов.

Модуль в языке C

В языке C модулем называется совокупность из:

- единицы трансляции;
- заголовочного файла.

Неопределенное поведение

Неопределенным поведением называется поведение программы, не определенное стандартом и компилятором.

Например, использование неинициализированных переменных:

```
int a;
printf("%d\n", &a);   // ???
```

или переполнение знаковых целых типов:

```
int a = INT_MAX;
a++;    // ???
```

Неспецифицированное поведение

Неспецифицированным поведением в языке C называется поведение программы, при котором стандарт определяет несколько вариантов, а компилятор может реализовать любой вариант.

Например, порядок вычислений аргументов подпрограммы или операндов.

```
int foo(void)
{
    printf("foo ");
    return 0;
}

int bar(void)
{
    printf("bar ");
    return 0;
}

...

foo() + bar();  // foo bar
                // или
                // bar foo
```

Неявное преобразование типа

Неявным преобразованием типа в языке C называется преобразование типа не являющееся явным.

Неявное преобразование происходит в следующих случаях:

- операнды арифметической или логической операции имеют различные типы (арифметическое преобразование типа);

```
int a = 5;
double b = 6.5;

a + b;  // (double)a + b;
```

- операнды операции присваивания различаются имеют различные типы.

```
double b = 0;   // b = 0.0
```

- тип аргумента в вызове подпрограммы не соответствует типу в сигнатуре.

```
char *strchr(char *s, int c);

...

char a = 'a';
strchr(str, a);     // strchr(str, (int)a)
```

- тип выражения в операторе return не соответствует типу возвращаемого значения функции.

```
long sth(void)
{
    ...

    return 0;   // return (long)0;
}
```

Нулевой адрес

Нулевым адресом называется адрес, равный нулю.

Гарантируется, что ни один объект не будет иметь нулевой адрес.

Определяется макросом в заголовочном файле `stddef.h` как `NULL`, который равен нулю.

Нулевой указатель

Нулевым указателем называется указатель, равный нулевому адресу.

Область видимости

Областью видимости или scope идентификатора в языке C называется все области программы, в которых идентификатор видим.

В языке C выделяют следующие области видимости:

- файл;
- прототип функции.
- блок;
- функция;

Внутренняя область видимости является подмножеством внешней области видимости.

Любая область видимости начинается с объявления сущности.

Область видимости файл

Область видимости заканчается концом единицы трансляции.

```
// a.c
int a;

// b.c
int b;
```

Область видимости блок

Область видимости заканчается концом составного оператора.

```
{
    {
        int a;
    }
    int a;
}
```

Область видимости прототип функции

Область видимости заканчается концом объявления подпрограммы (функции).

```
int sum(int a, int b);

int sub(int a, int b);
```

> Не путать с определение подпрограммы (функции)!

```
// Нет ; - это определение функции
//                   V    а значит, область видимости БЛОК
int sum(int a, int b)
{
    int a;  // ERROR
}
```

Область видимости функция

Область видимости заканчивается концом тела подпрограммы.

Область видимости функция имеют только метки.

Объединение

Объединением в языке С называется тип данных, содержащий упорядоченный набор переменных, объединенных одним идентификатором и разделяющих одну область памяти.

> Необходимо различать структуры и объединения.

Формальная запись объявления объединения в языке С состоит из:

- ключевого слова `union`;
- тега объединения;
- тела объединения, состоящее из перечисления переменных, разделяющих одну область памяти.

```
union тег { переменная; ...; переменная; };

union тег идентификатор, идентификатор;
```

или совмещенное объявление:

```
union тег { переменная; ...; переменная; } идентификатор, идентификатор;
```

Объектный файл

Объектным файлом называется бинарный файл, содержащий блоки машинного кода с *неопределенными ссылками* на данные и подпрограммы, а также список своих данных и подпрограмм.

Объектный файл состоит из:

- заголовка — содержит служебную информацию;
- секций
  - `.text` — исполняемый код;
  - `.bss` — неинициализированные переменные с областью видимости статическая или файл;
  - `.data` — инициализированные переменные с областью видимости статическая или файл;
  - `.rodata` — данные только для чтения (неизменяемые данные).

Объявление

Объявлением называется точка программы, в которой объявляется значение идентификатора.

Объявления в языке C

Объявление переменной в языке C:

```
тип идентификатор;
```

Объявление подпрограммы в языке C:

```
сигнатура;
```

См. сигнатура подпрограммы.

Одномерное адресное пространство

Одномерным адресным пространством или плоской моделью памяти называется пространство элементов, каждый из которых характеризуется одним уникальным целым беззнаковым числом — адресом.

Минимальная разница между двумя различными адресами в плоской модели памяти называется байтом.

Операции в одномерном адресном пространстве

В одномерном адресном пространстве определены следующие операции:

- получение адреса сущности;
- доступ к сущности по адресу;
- сложение адреса с целым числом;
- вычитание адресом.

Одномерное адресное пространство в языке C

См. указатели в языке C.

Одномерный массив

Одномерным массивом называется массив, элементами которого не являются массивами.

> См. также одномерные массивы в языке C.

Операнд

Операндом называется элемемент данных, к которым применяется оператор.

Пример:

```
a + 5
```

- `a` — операнд;
- `+` — оператор сложения;
- `5` — операнд.

Оператор break

Оператором break в языке C называется оператор, выполняющий функцию передачи управления в конец тела оператора while, оператора do while, оператора for или оператора выбора.

Пример:

```
char *strchr(const char *s, int c)
{
    const char *p;

    for (*p = s; *p != '\0'; p++)
        if (*p == c)
            break;    // Заканчивает цикл и передает управление -+
                      //                                         |
    return (char *)p; // <---------------------------------------+
}
```

Оператор continue

Оператором continue в языке C называется оператор, выполняющий функцию передачи управления к началу тела оператора while, оператора do while или оператора for.

Оператор do while

Оператором do-while или цикл с постусловием в языке C называется оператор, предназначенный для описания циклов.

Фомальная запись оператора do while:

```
do оператор while (выражение);
```

Оператор for

Оператором for или цикл со счетчиком в языке C называется оператор, предназначенный для описания циклов.

Формальная запись оператора for:

```
for (инициализация; условие; обновление) оператор
```

- `инициализация` — выражение, которое выполнится единожды первым;
- `условие` — выражение, истинность которого определяет конец цикла;
- `обновление` — выражение, выполняемое после каждого выполнения `оператора`.

Оператор goto

Оператором goto в языке C называется оператор, передающий управление в другую точку программы.

Формальная запись оператора `goto`:

```
goto метка;
```

Формальная запись метки:

```
метка: оператор
```

Пример использования `goto` для обработки освобождения ресурсов:

```
// Нечто, обрабатывающее файл дважды.
int main(void)
{
    int rc = 0;

    FILE *fd = fopen("sth.txt", "r+t");
    if (fd == NULL)
    {
        rc = ERR_IO;
        goto exit;
    }

    rc = proccess(fd);
    if (rc != 0)
        goto release;

    rc = proccess(fd);

// Освободить ресурсы.
release:
    fclose(fd);

// Выйти из программы.
exit:
    return rc;
}
```

Оператор return

Оператором return в языке C называется оператор, прерывающий выполнение подпрограммы и осуществляющее возврат значения выражения после ключевого слова `return` в функциях и прерывание выполнение в процедурах.

Формальная запись оператора return в языке C:

```
return;           // Для процедур.
```

```
return выражение; // Для функции.
```

> Использование оператора return в процедурах прерывает выполнение процедуры.

Оператор typedef

Оператором typedef в языке C называется оператор, определяющий синоним типа данного.

Формальная запись оператора typedef:

```
typedef тип синоним;
```

Пример:

```
typedef int grade_t;
```

Или в случае со структурами:

```
typedef struct {
    char name[STUDENT_NAME_MAX_LEN];
    grade_t grade;
} student_t;    // student_t - синоним к структуре.
```

> Принято называть синонимы типа с суффиксом `_t`.

Оператор while

Оператором while или цикл с предусловием в языке C называется оператор, предназначенный для описания циклов.

Фомальная запись оператора while:

```
while (выражение) оператор
```

Оператор выбора

Оператором выбора в языке C называется оператор, позволяющий сделать выбор.

В отличие от условного оператор, оператор выбора проверяет условие равенства *управляющего выражения* с константой.

Формальные способы записи оператора выбора:

```
switch (управляющее_выражение)
{
    case литерал: операторы
}
```

```
switch (управляющее_выражение)
{
    case литерал: операторы
    ...
    case литерал: операторы
}
```

```
switch (управляющее_выражение)
{
    case литерал: операторы
    ...
    case литерал: операторы
    default: операторы
}
```

> Важное замечание: после `case литерал` следуют *операторы*.

Использование оператора выбора

1. В группе оператором после `case` последним *обычно* является оператор break.

```
switch (mark)
{
    case 5:
        printf("Отлично\n");
        break;
    case 4:
        printf("Хорошо\n");
        break
    case 3:
        printf("Удовлетворительно\n");
        break;
    case 2:
        printf("Неудовлетворительно\n");
        break;
    default:
        printf("Ошибка: неизвестная оценка %d\n", mark);
}
```

Без оператора `break` управление передается за следующую метку:

```
int a = 4;
switch (a)
{
    case 5:
        printf("Отлично\n");
    case 4:
        printf("Хорошо\n");
    case 3:
        printf("Удовлетворительно\n");
    case 2:
        printf("Неудовлетворительно\n");
    default:
        printf("Ошибка: неизвестная оценка %d\n", mark);
}

// Вывод после запуска программы:
//
//      Хорошо
//      Удовлетворительно
//      Неудовлетворительно
//      Неизвестная оценка
```

Оператор выражение

Оператором выражение в языке С называется оператор, состоящий из выражения и символа `;`.

Пример:

```
a = 6;
a + 5;
```

`a + 5` и `a = 6` - выражения.

Оператор

Оператором (`statement`) называется синтаксическая единица языка, обозначающее действие, которое требуется выполнить.

В языке С представлены следующие операторы:

- оператор выражение;
- составной оператор;
- условный оператор;
- оператор выбора;
- оператор while
- оператор do while
- оператор for
- оператор break;
- оператор continue;
- оператор return;
- оператор goto;
- пустой оператор.

> Замечание: в курсе «Программирование на Си» `операция` обозначает `оператор`, а `оператор` — `инструкцию`.

Операция sizeof

Операцией sizeof в языке С называется унарная операция, применяемая к переменной или типу данных и возвращающая размер.

Операция

Операцией (`operator`) называется конструкция в языках программирования для записи некоторых *действий*. Элементы данных, к которым применяется операция, называют операндами.

Операции в языке С

Операции в языке С представлены в таблице ниже.

Операция	Название	Нотация	Класс	Приоритет	Ассоциативность
`()`	Вызов функции	`f(a, b)`	Постфиксная	16	Левоассоциативно
`[]`	Индекс	`a[i]`	Постфиксная	16	Левоассоциативно
`.`	Прямой выбор поля	`a.b`	Постфиксные	16	Левоассоциативно
`->`	Выбор поля через указатель	`a->b`	Постфиксные	16	Левоассоциативно
`++`	Инкремент	`a++`	Постфиксные	16	Левоассоциативно
`--`	Декремент	`a--`	Постфиксные	16	Левоассоциативно
`++`	Инкремент	`++a`	Префиксные	15	Правоассоциативно
`--`	Декремент	`--a`	Префиксные	15	Правоассоциативно
`+`	Плюс	`+a`	Префиксные	15	Правоассоциативно
`-`	Минус	`-a`	Префиксные	15	Правоассоциативно
`!`	Логическое «НЕ»	`!a`	Префиксные	15	Правоассоциативно
`~`	Побитовое «НЕ»	`~a`	Префиксные	15	Правоассоциативно
`&`	Адрес	`&a`	Префиксные	15	Правоассоциативно
`*`	Разыменование	`*p`	Префиксные	15	Правоассоциативно
`sizeof`	Размер	`sizeof a`	Префиксные	15	Правоассоциативно
`()`	Преобразование типа	`(int) a`	Префиксные	14	Правоассоциативно
`*`	Умножение	`a * b`	Инфиксные	13	Левоассоциативно
`/`	Деление	`a / b`	Инфиксные	13	Левоассоциативно
`%`	Остаток	`a % b`	Инфиксные	13	Левоассоциативно
`+`	Сложение	`a + b`	Инфиксные	12	Левоассоциативно
`-`	Вычитание	`a - b`	Инфиксные	12	Левоассоциативно
`<<`	Побитовый сдвиг влево	`a << b`	Инфиксные	11	Левоассоциативно

`>>`	Побитовый сдвиг вправо	`a >> b`	Инфиксные	11	Левоассоциативно
`<`	Меньше	`a < b`	Инфиксные	10	Левоассоциативно
`<=`	Меньше или равно	`a <= b`	Инфиксные	10	Левоассоциативно
`>`	Больше	`a > b`	Инфиксные	10	Левоассоциативно
`>=`	Больше или равно	`a >= b`	Инфиксные	10	Левоассоциативно
`==`	Равно	`a == b`	Инфиксные	9	Левоассоциативно
`!=`	Не равно	`a != b`	Инфиксные	9	Левоассоциативно
`&`	Побитовое «И»	`a & b`	Инфиксные	8	Левоассоциативно
`^`	Побитовый «XOR»**	`a ^ b`	Инфиксные	7	Левоассоциативно
`|`	Побитовое «ИЛИ»	`a | b`	Инфиксные	6	Левоассоциативно
`&&`	Логическое «И»	`a && b`	Инфиксные	5	Левоассоциативно
`||`	Логическое «ИЛИ»	`a || b`	Инфиксные	4	Левоассоциативно
`?:`	Тернарный оператор	`c ? a : b`	Инфиксные	3	Правоассоциативно
`=`	Присваивание	`a = b`	Инфиксные	2	Правоассоциативно
`[?]=`	Комплексное присваивание*	`a += b`	Инфиксные	2	Правоассоциативно
`,`	Запятая	`a, b`	Инфиксные	1	Правоассоциативно

> * комплексное присваивание определено для всех арифметических и побитовых операций: `+=`, `%=`, `&=`…
> ** XOR равносильно «Исключающее ИЛИ».

> Замечание: в курсе «Программирование на Си» `операция` обозначает `оператор`, а `оператор` — `инструкцию`.

> См. также:
>
> - операция sizeof;
> - указатели;

Порядок вычисления значений операндов

В общем случае порядок вычисления значений операндов не определен. Для логических операций определена ленивая логическая схема.

Пример:

```
int f(int a)
{
    printf("%d\n", a);
    return a;
}

...

int a = 5;
int b = 6;
f(a) + f(b);
// Вывод программы:
//    5           6
//    6    или    5
```

Порядок выполнения операций

Порядок выполнения операций определен **приоритетом** и **ассоциативностью операции**

1. Если операции имеют различный приоритет, сначала выполняется операции с наибольшим приоритетом.
2. Если бинарные операции имеют равный приоритет, порядок выполнения определяется правилами ассоциативности:
    1. *Левоассоциативные операции* выполняются *слева направо*.
    2. *Правоассоциативные операции* полняются *справа налево*.

Пример:

```
a + b * c * d
```

1. Операции умножения `*` имеют больший приоритет, чем операция сложения.
2. Операция умножения `*` левоассоциативна. Выполняется `b * c`.
3. Операция умножения `*` левоассоциативна. Выполняется `(b * c) * d`.
4. Следующая операция по приоритету -- операция сложения. Выполняется `((b * c) * d) + a`

Память

Памятью будем называть одномерное адресное пространство.

Передача параметра по значению

Передачей параметра по значению называется способ передачи параметров в подпрограмму, при котором значений фактического параметра используется для инициализации формального параметра.

При вызове подпрограммы с передачей параметров по значению значение переменных, переданных в качестве аргументов, не изменяется.

Передача параметров по значению в C

> В `C99` существует **только** передача параметров по значению.

Пример:

```
int foo(int a)
{
    a++;
    return a;
}

...

int a = 5;
foo(a); // 6
a;      // 5
```

Передача параметра по ссылке

Передачей параметра по ссылке называется способ передачи параметров в подпрограмму, не являющийся передачей параметра по значению.

При вызове подпрограммы с передачей параметров по ссылке значение переменных, переданных в качестве аргументов, **может** изменится.

Передача параметров по ссылке в C

> В `C99` формально не существует передача параметров по ссылке. Однако, существует механизм передачи параметров по указателю.

Пример:

```
int foo(int *a)
{
    *a++;
    return *a;
}

...

int a = 5;
foo(&a); // 6
a;       // 6
```

Переменная объединенного типа

Переменной объединенного типа в языке C называется переменная, тип данных которой является типом объединения.

Формальная запись объявления переменной объединенного типа в языке C:

```
union тег идентификатор;
```

```
union тег идентификатор1, идентификатор2...;
```

> Следует различать тег объединения и идентификатор переменной объединенного типа.

```c
union color {
    uint32_t value;
    struct rgba {
        uint8_t red;
        uint8_t green;
        uint8_t blue;
        uint8_t alpha;
    }
}

union color color;  // OK
// ------>| |<-->|<---------
//     тег           имя
// объединения    переменной
```

> Тег объединения не конфликтует с идентификаторами переменных объединенного типа, т.к. требует использовать ключевое слово `union`.

Часто использование объединений машинно-зависимо. Пример выше зависим от порядка байтов.

```c
union color red = 255;
//            ?
// red = 255     alpha = 255
//           или
//    BE            LE
```

Переменная структурного типа

Переменной структурного типа в языке С называется переменная, тип данных которой является структурным типом.

Формальная запись объявления переменной структурного типа в языке С:

```c
struct тег идентификатор;
```

```c
struct тег идентификатор1, идентификатор2...;
```

> Следует различать тег структуры и идентификатор переменной структурного типа.

```c
struct student {
    char name[STUDENT_NAME_MAX_LEN];
    int grade;
};

struct student student; // OK
// --------->| |<---->|
//     тег        имя
//  структуры  переменной
```

> Тег структуры не конфликтует с идентификаторами переменных структурного типа, т.к. требует использования ключевого слова `struct`.

Переменная

Переменной в языке C называется именованный участок памяти.

Переменная характеризуется:

- именем (идентификатором);
- адресом;
- типом;
- значением.

Формальная запись объясления переменной:

```
тип идентификатор;
```

Перечисляемый тип данных

Перечисляемым типом данных или перечислением называется простой тип данных, каждому значению из конечного множества значений типа сопоставлен идентификатор.

Перечисляемый тип данных в C

Перечисляемый тип данных в C является подтипом целочисленного типа данных, способного представить все значения типа `int`.

Формальная запись объявления перечисляемого типа данных в C:

```
enum идентификатор { идентификатор, ..., идентификатор };
```

Пример:

```
enum day {
    MONDEY    = 1,
    THUSDAY   = 2,
    WEDNESDAY = 3,
    THURSDAY  = 4,
    FRIDAY    = 5,
    SATURDAY  = 6,
    SUNDAY    = 7
};
```

Значения, которым соответствуют идентификаторы, можно опустить: в таком случае, они будут присвоены автоматически.

Пример:

```
enum day {
    MONDEY,
    THUSDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
};
```

Подпрограмма

Подпрограммой называется именованная часть программы, содержащая набор операторов.

Подпрограммы делятся на:

- процедуры;

- функции.

Преимущества использования подпрограмм:

- декомпозиция задачи;
- уменьшение дублирования;
- повтороное использование кода в других программах;
- сокрытие деталей реализации.

Подпрограмма в языке C

> В языке C подпрограммы часто называют функцией.

Подпрограмма в языке C состоит из:

- заголовка (сигнатуры);
- тела подпрограммы.

Формальная запись подпрограммы в языке C:

```
тип идентификатор(перечисление формальных параметров)
{
    оператор1
    ...
    операторN
}
```

Здесь:

- `тип идентификатор(перечисление формальных параметров)` — сигнатура подпрограммы;
- `{ оператор1, ..., операторN }` — тело подпрограммы.

Поле структуры

Полем структуры называется переменная, входящая в состав переменной структурного типа.

Поля структуры располагаются в памяти в том порядке, в котором они были объявлены.

Структурный тип данных `A` не может содержать поле типа данных `A`:

```
struct A {
    struct A a; // Ошибка времени компиляции
}
```

Однако может быть указателем:

```
struct A {
    struct A *a; // OK
};
```

> Это возможно потому что:
>
> - указатель имеет фиксированный размер, не зависящий от содержимого структуры;
> - бесконечной рекурсии можно избежать особым значением NULL.

Порядок байтов

Порядком байтов называется правило размещения байтов многобайтовых переменных в памяти.

Порядок байтов бывает:

- от младшего к старшему (`BE` или big-endian);
- от старшему к младшему (`LE` или little-endian);

Порядок байтов является машинно-зависимым.

Пример: число `0x1234` записанное в 4-байтовую переменную в порядке `LE`:

```
0x00 0x00 0x12 0x34
----------> старшие адреса
```

В порядке `BE`:

```
0x34 0x12 0x00 0x00
----------> старшие адреса
```

Порядок байтов от младшего к старшему позволяет проводить более эффективное сравнение, в то время как порядок байтов от старшего к младшему позволяет проводить более эффективное преобразование типов.

Предопределенное имя

Предопределенным именем называется идентификатор, не являющийся ключевым словом, использование которого в качестве идентификатора, определяемого пользователем, неопределено.

> То есть это не ключевое слово, но и использовать его в качестве имени нельзя.

> Более точная формулировка см. `C99` `7.1.3`.

Преобразование типа

Преобразованием типа в языке C называется изменение типа значения выражения.

Преобразование типа может быть:

- явным;
- неявным;

Препроцессирование

Препроцессированием называется комплекс действий над исходным кодом программы перед компиляцией.

Препроцессирование в языке C представляет из себя:

- удаление комментариев;
- вставку файлов (директива `#include`);
- текстовые замены или раскрытие макросов (директива `#define`);
- условную компиляцию (директива `#if`).

Препроцессор

Препроцессором называется программа, выполняющая препроцессирование.

Препроцессор языка C в GNU Linux — `The C Preprocessor` (`cpp`).

Пробельный символ

Пробельным символом в языке C называется символ, `isspace` от которого вернет `true`.

Список пробельных символов **обычно** включает

Название	Представление в языке C
Пробел	` `
Перевод строки	`\r`
Возврат каретки	`\r`
Табуляция	`\t`

> В языке C многие пробельные символы представляют из себя escape-последовательности.

Простой тип данных

Простым типом данных называется тип данных, сущность которого не содержит в себе других сущностей.

Простые типы данных в C

В языке C существуют следующие простые типы данных:

- целочисленные типы данных;
- вещественные типы данных;
- комлексные типы данных (следует различать с составным типом данных);
- указатели;
- void.

Процедура

Процедурой в языке C называется подпрограмма, тип возвращаемого значения которой является void.

> Согласно `C99` в C нет процедур (только функции). Однако, мы вводим определение процедуры, чтобы различать `void` и не-`void` функции.

Пример процедуры в языке C:

```
void great(char *name)
{
    printf("Hello, %s!\n", name);
}
```

Пунктуатор

Пунктуатором называется символ, имеющий независимое синтаксическое и семантическое значение. В зависимости от контекста он определяет операцию для выполнения.

Пунктуаторы в языке C:

```
[ ] ( ) { } . ->
++ -- & * + - ~ !
/ % << >> < > <= >= == != ^ | && ||
? : ; ...
= *= /= %= += -= <<= >>= &= ^= |=
, # ##
<: :> <% %> %: %:%:
```

Пустой оператор

Пустым оператором в языке C называется оператор, состоящие исключительно из символа `;`. Ничего не делает.

Размер

Размером переменной называется количество байт, занимаемых переменной в памяти.

Размером типа данных называется размер переменной соответствующего типа данных.

Ранк целочисленного типа

Ранком целочисленного типа данных в языке C называется свойство целочисленного типа данных, определяющее порядок арифметических преобразований.

Ранк определен `C99` как:

- Никакие два целочисленных знаковых типа не имеет равного ранка.
- Ранк знакового типа больше ранка знакового типа с меньшей точностью (размером).
- Ранк `long long` больше ранка `long`.
- Ранк `long` больше ранка `int`.
- Ранк `int` больше ранка `short`.
- Ранк `short` больше ранка `signed char`.
- Ранк беззнакового типа равен ранку соответствующего знакового типа.
- Ранк `char` равен ранку `signed char` и `unsigned char`.
- Ранк `_Bool` меньше любого другого типа (см. логический тип данных).
- Ранк перечисляемого типа равен типу соответствующего целочисленного типа.
- Если ранк `T1` больше ранка `T2`, а ранк `T2` больше `T3`, то ранк `T1` больше ранка `T3`.

> Фактически это определяет следующий порядок арифметических преобразований типов:
>
> 1. `long long`, `unsigned long long`
> 2. `long`, `unsigned long`
> 3. `int`, `unsigned int`
> 4. `short`, `unsigned short`
> 5. `char`, `signed char`, `unsigned char` (см. символьный тип данных)
> 6. `_Bool` (см. логический тип данных).

Расширение файла

Расширением файла или расширение называется суффикс названия файла, указывающего на характеристику содержимого файла.

Файл может не иметь расширения.

> Строго говоря:
>
> - расширение файла не определяет тип файла
> - расширение файла лишь помогает пользователю при взаимодействии с файлом, и не всегда соответствует его содержимому.
>
> Можно поменять расширение у бинарного файла на текстовое, однако файл остается бинарным.

Регистр процессора

Регистром процессора называется быстродоступная память процессора.

Связывание

Связыванием или linkage идентификатора в языке C называется явление, при котором объявленный в нескольких местах идентификатор ссылается на одну и ту же сущность.

В языке C выделяют следующие типы связывания:

- внешнее (`external`);
- внутреннее (`internal`);
- отсутствие (`no linkage`).

Внешнее связывание

Связывание называется *внешним*, если на сущность ссылается идентификатор из нескольких единиц трансляции.

Внутреннее связывание

Связывание называется *внешним*, если на сущность ссылается идентификатор из одной единицы трансляции.

Отсутствие связывание

Идентификатор, *не имеющий* связывания, ссылается на уникальную сущность.

Семантика

Семантикой языка программирования называется набор правил придания смысла синтаксически правильным программам. В конечном счете определяет последовательность действий вычислительной машины.

Сигнатура подпрограммы

Сигнатурой подпрограммы или заголовок подпрограммы в языке C называется совокупность:

- имени (идентификатора) подпрограммы;
- перечисление формальных параметров подпрограммы;
- тип возвращаемого значения подпрограммы.

Сигнатура подпрограммы в C

Формальная запись сигнатуры подпрограммы в C в C:

```
идентификатор() // C89, переменное количество формальных параметров.
                // C99 - deprecated
```

```
тип идентификатор(void) // Подпрограмма без формальных параметров.
```

```
тип идентификатор(параметр1, ..., параметрN) // Подпрограмма с конечным числом формальных
параметров.
```

```
// Подпрограмма, имеющая конечное число заданных формальных параметров
// и произвольное количество произвольных параметров.
тип идентификатор(параметр1, ..., параметрN, ...)
//                                           ^
//                                буквально три точки
```

> См. подпрограммы с переменным количеством аргументов.

Пример:

```
int sum(int a, int b)
```

Здесь:

- `int` — тип возвращаемого значения;
- `sum` — имя (идентификатор) подпрограммы;
- `int a`, `int b` — список формальных параметров.

Символьный тип данных

Символьным типом данных в языке C называется целочисленный тип данных, достаточным для хранения всех символов алфавита, при этом должно гарантироваться, что их значение будет неотрицательным.

> Знак хранится в старшей разрядке. Значит, для CHAR_BIT = 8, алфавит должен умещаться в CHAR_BIT - 1 = 7 бит, что соответствует размеру символа в ASCII.

Символьными типами данных в языке C являются:

- `char`;
- `signed char`;
- `unigned char`.

> Замечание: в отличие от других целочисленных типов данных, `char` не эквивалентен `signed char`. Более того, `char` может быть как `signed`, так и `unsigned` (неспецифицированное поведение).

Символьный тип данных инициализируется символов, записанным в одинарных кавычках: `'`.

```
char a = 'a';
```

> EOF не является символом и не входит в множество значений символьного типа данных.

Синтаксис

Синтаксисом языка программирования называется набор правил, описывающий комбинации символов алфавита языка, считающиеся правильно структурированной программой (документов) или ее фрагментом.

Способы описания синтаксиса языка:

- формальные грамматики (Керниган Б.У., Ритчи Д.М. «Язык программирования C» Приложение A);
- формы Бекуса-Наура (`BNF`);
- расширенные формы Бекуса-Наура (ISO 14977);
- диаграммы Вирта.

Система компиляции

Системой компиляции (toolchain) называется набор инструментов, который используется для разработки программ.

GCC

Система компиляции для языка C в GNU Linux представлена в виде *GNU project C and C++ compiler* (`gcc`). Он включает в себя:

- компилятор,
- ассемблер,
- компоновщик (линкер),
- набор библиотек,

- отладчик,
- профилировщик.

Ключи для `gcc`

Ключ	Значение
`-E`	Препроцессирование исходного кода программы
`-S`	Трансляция на язык ассемблера
`-c`	Ассемблирование
`-o`	Задает имя выходного файла
`-std=`	Задает используемый стандарт языка C
`-W[a-z]+`	Компилятор будет выводить предупреждения
`-Werror`	Компилятор считает любое предупреждение ошибкой времени трансляции
`-g[0-3]`	Задает уровень отладочной информации
`-O[0-3]`	Задает уровень оптимизациия

Слово

Словом называется последовательность символов алфавита.

Соглашение о вызове

Соглашением о вызове называется набор правил по использованию аппаратного стека, определяющих:

- расположение аргументов подпрограммы;
- порядок передачи аргументов подпрограммы;
- какая из сторон (вызывающая или вызываемая) очищает стек.

Соглашение о вызове cdecl

- аргументы передаются через стек, справа налево;
- очистку стека производит вызывающая сторона;
- результат функции возвращается через регистр `?AX`.

См. стековый кадр.

Составной оператор

Составным оператором называется оператор, группирующий один или несколько операторов.

Составной тип данных

Агрегированным или составным типом данных называется тип данных, сущность которого содержит в себе другие сущности.

Спецификатор формата

Спецификатором формата называется строка, предназначенная для форматированного ввода/вывода.

Спецификатор имеет вид:

```
%[флаги][ширина][.точность][размер]тип
```

Спецификатор флаги

Флаги используются для вывода чисел.

Флаг	Значение	Отсутствие
`-`	Выравнивание по левому краю	Выравнивание по правому краю
`+`	Указывать знак числа	Указывать только `-`
` `	Выводить пробел перед положительными числами	
`#`	Альтернативная форма вывода (`0x`, `0`)	
`0`	Дополнять число до ширины нулями	Дополнять число до ширины пробелами

Спецификатор модификатор ширины

Определяет минимальную ширину вывода числа.

Спецификатор модификатор точности

Определяет количество знаков после запятой у вещественных типов.

Спецификатор модификатор размера

Применяет *спецификатор типа* к конкретному размеру целочисленного или вещественного значения.

Тип	Модификатор размера
`char`	`hh`
`short`	`h`
`int`	Отсутствует
`long`	`l`
`long long`	`ll`
`intmax_t`	`j`
`size_t`	`z`
`ptrdiff_t`	`t`

Спецификатор типа

Указывает на тип и на способ вывода.

Спецификатор	Значение
`d`, `i`	Целочисленное знаковое, десятичный вид
`u`	Целочисленное беззнаковое, десятичный вид
`o`	Целочисленное беззнаковое, восьмиричный вид
`x`	Целочисленное беззнаковое, шестнадцатеричный вид
`f`, `F`	Вещественное, обычная запись
`e`, `E`	Вещественное, экспоненциальная запись
`g`, `G`	Вещественное, без незначащих нулей после точки
`a`, `A`	Вещественное, шестнадцатеричный вид
`s`	Строка
`p`	Указатель
`%`	Знак `%`

Стандартная библиотека

Стандартной библиотекой языка C называется набор заголовочных файлов, определяемых стандартом.

Стандартная библиотека языка C включает в себя следующие заголовочные файлы:

Название	Предназначение
`ctype.h`	Работа с символами
`inttypes`	Вводит макросы для работы с целыми типами фиксированной длины
`limits.h`	Определяет области применения целочисленных типов
`math.h`	Основные математические функции
`stdint.h`	Вводит дополнительные целочисленные типы
`stddef.h`	Вводит основные макросы и синонимы типов
`stdio.h`	Реализует базовый ввод/вывод
`stdlib.h`	Основные функции стандартной библиотеки
`string.h`	Работа со строками
`time.h`	Работа с датами и временем

> Это не полный список!

Статическая библиотека

Статической библиотекой называется библиотека, связываемая с программой в момент компоновки.

Особенности использования статических библиотек:

- исполняемый файл содержит в себе все необходимое для запуска;
- имеет больший размер, чем динамическая библиотека;
- обновление библиотеки требует пересборки проекта.

Статический массив

Статическим массивом в языке C называется массив, размер которого известен на момент трансляции.

Одномерный статический массив

Формальная запись объявления одномерного статического массива в языке C:

```
тип идентификатор[размер];
```

`размер` должен быть константой.

```
char array[sizeof(int)]; // Допустимая запись
```

Доступ к одномерном массиву осуществляется операцией индекса:

```
// array = { 1, 2, 3, 4, 5 }
array[2];   // 3
```

Инициализация одномерного статического массива

Инициализация одномерного статического массива:

```
int array[5] = { 1, 2, 4, 5, 3 };
// { 1, 2, 4, 5, 3 }
```

Инициализация одномерного статического массива нулями:

```
int array[5] = { 0 };
// { 0, 0, 0, 0, 0 }
```

Выделенные инициализаторы:

```
int array[5] = { [2] = 4, [0] = 1 };
// { 1, 0, 4, 0, 0 }
```

Многомерный статический массив

Формальная запись объявления многомерного статического массива в языке C:

```
тип идентификатор[размер1]...[размер4]
```

Например, многомерный целочисленный массив размерностью `3x2x4`:

```
int array[3][2][4];
```

Доступ к многомерному массиву осуществляется операцией индекса:

```
// array:
//  1 2 3
//  4 5 6
array[1][2]; // 6
```

Многомерные статические массивы используются как способ представления матриц в языке C.

Инициализация многомерного статического массива

Инициализация многомерного статического массива:

```
int array[2][3] = { {1, 2, 3}, {4, 5, 6} };
// 1 2 3
// 4 5 6
```

Размер массива объявленного с инициализатором

Размер статического массива может быть определен автоматически при инициализации:

```
int array[] = { 1, 2, 3 };
// { 1, 2, 3 }
```

Однако это возможно, если массив не является элементом другого массива:

```
int array[][3] = {
    { 1, 2, 3 },
    { 4, 5, 6 },
}
```

```
// Ошибка времени трансляции
int array[][] = {
    { 1, 2, 3 },
    { 4, 5, 6 },
}
```

Стековый кадр

Стековым кадром называется механизм передачи аргументов и выделения области памяти с использованием аппаратного стека.

В стековом кадре размещаются:

- значения аргументов подпрограммы;
- адрес возврата;
- автоматические переменные;
- иные данные, связанные с вызовом подпрограммы.

> Порядок роста аппаратного стека является машинно-зависимым.

Пример **на машине автора (у Вас может быть иначе)**. Для программы на C:

```
int foo(int a, int b)
{
    int c;
    int d;
}

foo(4, 5);
```

Стековый кадр для `foo`:

```
|                |   ^ Младшие адреса
+----------------+   |
|      ...       |   |
+----------------+
|       c        |
+----------------+
|       d        |
+----------------+
| Адрес возврата |
+----------------+
|       4        |
+----------------+
|       5        |
+----------------+
|      ...       |   |
+----------------+   |
|                |   V Старшие адреса
```

Строка

Строкой в языке C называется последовательность символов, содержащая заканчивающаяся на единственный нулевой символ.

> В других языках программирования строка может иметь иное представление.

Строка как массив символов

В языке C строка хранится как массив символов, причем последний символ обязательно является нуль терминалом.

Массив символов может инициализироваться строковым литералом. Причем стоит учитывать, что строковый литерал сам по себе содержит в конце терминальный нуль.

```
char s[] = "foo";   // 'f' 'o' 'o' '\0'
sizeof(s);  // 4
```

Строка как указатель

В языке C строка может быть представлена как указатель на первый символ. Например, указатель на первый символ строкового литерала.

```
char *s = "foo bar"
sizeof(s);  // 4 = sizeof(char *)
            // не sizeof("foo bar")!
```

Функции для работы со строкой

Для работы со строкой в языке C существует заголовочный файл `string.h`.

Основные функции `string.h`:

Название	Предназначение
`strcpy`	Копирование строки
`strncpy`	Копирование строки с защитой
`strlen`	Длина строки
`strcat`	Конкатекация (объединение) строк
`strncat`	Конкатекация (объединение) строк с защитой
`strcmp`	Сравнение строк в лексикографическом порядке
`strncmp`	Сравнение первых n символов строк в лексикографическом порядке
`strtok`	Простейший лексический анализатор
`strchr`	Поиск первого вхождения символа в строке
`strrchr`	Поиск последнего вхождения символа в строке
`strspn`	Длина префикса строки
`strcspn`	Длина префикса строки
`strpbrk`	Поиск первого символа из набора в строке
`strstr`	Поиск подстроки в строке

Строковый литерал

Строковым литералом в языке C называется строка, являющаяся литералом.

Формальная запись строкового литерала:

```
"печатаемые-символы..."
```

В памяти строковый литерал представлен массивом символов, оканчивающийся терминальным нулем.

Попытка изменения строкового литерала неопределено.

```
char *s = "foo"; // 'f' 'o' 'o' '\0'
```

Пример:

```
const char * const s = "lorem ispum";
// s - константа
// "lorem ispum" - литерал.
```

Два и более строковых литерала, расположенных рядом, объединяются:

```
printf("foo" "bar");    // foobar
```

Структура

Структурой или записью или структурный тип данных в языке C называется тип данных, содержащий упорядоченный набор переменных, объединенных одним идентификатором и разделяющих различные области памяти.

> Необходимо различать структуры и объединения.

Формальная запись объявления структуры в языке C состоит из:

- ключевого слова `struct`;
- тега структуры;
- тела структуры, состоящее из перечисления полей структуры.

```
struct тег { поле; ...; поле; };

struct тег идентификатор, идентификатор;
```

или совмещенное объявление:

```
struct тег { поле; ...; поле; } идентификатор, идентификатор;
```

Например:

```
struct student { // Тип - структура student.
    char name[STUDENT_NAME_MAX_LEN];
    int grade;
} engineer, programmer; // Переменные с типом структуры student.
```

Адрес переменной структурного типа совпадает с адресом первого поля.

> В отличие от массива, поля структуры в памяти не обязательно располагаются *линейно* из-за явления выравнивания. Именно поэтому *нельзя делать предположения о размере переменной структурного типа*. Определить размер переменной структурного типа можно только при помощи операции sizeof.

Структурный тип и оператор typedef

В языке C часто используется синоним типа структуры:

```
typedef struct { переменная; ...; переменная; } тип;
```

Например:

```
typedef struct {
    char name[STUDENT_NAME_MAX_LEN];
    int grade;
} student_t; // Синоним типа к структуре выше.
             // Можно объявлять множество переменных
             // с таким типом.
student_t student;  // Непосредственно сама структура.
```

Тег объединения

Тегом объединения называется идентификатор типа объединения.

> Следует различать тег объединения и идентификатор переменной объединенного типа.

Тег структуры

Тегом структуры называется идентификатор структурного типа.

> Следует различать тег структуры и идентификатор переменной структурного типа.

> Тег структуры не конфликтует с идентификаторами переменных структурного типа.

```
struct student {
    char name[STUDENT_NAME_MAX_LEN];
    int grade;
};

struct student student; // OK
// --------->| |<---->|
//     тег        имя
//  структуры  переменной
```

Текстовый файл

Текстовым файлом называется файл, который содержит преимущественно печатные символы.

> Формально, единственный способ отличить текстовый файл от бинарного — экспертная оценка. С точки зрения файловой системы бинарные и текстовые не отличаются.

Тело подпрограммы

Телом подпрограммы в языке С называется набор операторов, которые содержит подпрограмма.

Пример:

```
int sum(int a, int b)
{
    int c;
    c = a + b;
    return c;
}
```

Здесь

- `int sum(int a, int b)` — сигнатура подпрограммы;
- `{ ... }` — тело подпрограммы.

Терминальный нуль

Терминальный нуль или нулевой символ в языке С называется непечетаемый символ, обозначающий конец строки. Обозначается как управляющая последовательность `\0`.

Терминальный нуль имеет код `0` и является ложным.

Тип ptrdiff t

Типом ptrdiff_t называется синоним типа возвращаемого значения операцией разности для указателей.

Тип size t

Типом size_t называется синоним типа возвращаемого значения операцией sizeof.

Тип данных

Типом данных называется множество значений и операций над ними.

Типы данных делятся на:

- простые;
- составные.

Точка входа

Точкой входа называется точка программы, с которой начинается ее выполнение.

Точка входа в языке C

В языке C точкой входа является подпрограмма `main`.

Транслятор

Транслятором называется компьютерная программа, выполняющая трансляцию.

Трансляция

Трансляцией называется процесс преобразования исходного кода, написанной на одном языке программирования, в другой.

Требование к выравниванию

Требованием к выравниванию в языке C называются правила выравнивания данных в памяти.

> Важно: требование к выравниванию машинно-зависимо!

Пример требований к выравниванию.

Переменные располагаются на адресах, кратным своим размерам. Для этого используются *байты выравнивания*.

Пример:

```
char a;
int b;  // Пусть sizeof(b) = 4
```

Указатель на константу

Указателем на константу в языке C называется указатель, содержащий адрес константы.

> Указатель на константу **не** является константой.

Формальная запись указателя на константу:

```
const тип *идентификатор;
```

Пример:

```
const a = 5;
const b = 6;

const int *p = &a;
p = &b;    // Корректно.
*p = 7;    // Некорректно.
```

> Следует отличать от константного указателя.

Указатель

Указателем в языке C называется переменная или выражение, обозначающее адрес объекта.

Указатели бывают:

- типизированными — имеют тип;
- бестиповыми — `void *`;

С точки зрения константности, указатели могут быть:

- константный указатель;
- указатель на константу;

Любой указатель может иметь особое значение NULL.

Адресная арифметика в языке C

В языке C определены следующие операции с указателями:

Операция	Название	Объяснение
`&a`	Адрес	Возвращает адрес сущности `a`
`*pa`	Разыменование	Доступ к сущности по адресу `pa`
`pa + C`	Сложение адреса с константой	Для *типизированного указателя*: равносильно `pa + C * (размер_типа * байт)` Для *бестипового указателя*: равносильно `pa + C * байт`
`pa - pb`	Разность указателей	`pa` и `pb` должны быть одного типа. Равносильно `(pa - pb) / (размер_типа * байт)`

> См. также:
>
> - ptrdiff_t.
> - массив.

Унарная операция

Унарной операцией называется операция с одним операндом и возвращающая один результат.

Упаковка

Упаковкой в языке C называется явление, при котором данные в памяти расположены строго друг за другом.

Упаковка необходимо для экономии используемой памяти.

> Противопоставлено явлению выравнивания.

Управляющая последовательность

Управляющей последовательностью или экранирующей последовательностью или escape-последовательностью называется последовательность символов, теряющих свое индивидуальное значение в группе и обретая новое.

Управляющие последовательности в языке C

Последовательность	Значение
`\a`	Звонок (предупреждение)
`\b`	Backspace
`\f`	Подача страницы
`\n`	Новая строка
`\r`	Возврат каретки
`\t`	Горизонтальная табуляция
`\v`	Вертикальная табуляция
`\'`	Одиночная кавычка
`\"`	Двойная кавычка
`\\`	Обратная косая черта
`\?`	Литерал вопросительного знака
`\ooo`	Символ ASCII в восьмеричной нотации
`\xhh`	Символ ASCII в шестнадцатеричной нотации
`\xhhhh`	Символ юникода в шестнадцатеричном формате

Условный оператор

Условным оператором в языке C называется оператор, позволяющий сделать выбор.

Формальные записи условного оператора:

```
if (выражение)
    оператор
```

```
if (выражение)
    оператор
else
    оператор
```

> Замечание: запись `if ... else if ...` является комбинацией вышеперечисленных

```
if (выражение)
    оператор
else
    if (выражение)
        оператор
```

> Замечание: фигурные скобки `{ }` не входят в состав условного оператора, а являются составным оператором.

```
if (выражение)
{
    оператор
    оператор
}
```

Файл

Файлом называется именованная область данных на носителе информации.

Файл обладает следующими свойствами:

- имя файла (включает в себя расширение файла);
- тип файла;
- размер файла;
- права доступа;
- и т.д.

Файлы бывают следующих типов:

- обычные файлы:
  - текстовые файлы;
  - бинарные файлы;
- директории;
- специальные файлы.

Файловая система

Файловой системой -- специальный компонент операционной системы, назначение которой состоит в том, чтобы обеспечить пользователю интерфейс для работы с данными, хранящимися на носителе информации.

Файловая система включает в себя:

- совокупность всех файлов на носителе;
- структуры данных, используемые для управления файлами;
- интерфейс для взаимодействия с файлами.

Фактический параметр

Фактическим параметром подпрограммы или аргумент подпрограммы называется значение формального параметра при вызове подпрограммы.

Пример:

```
int sum(int a, int b)
{
    return a + b;
}

...

int c = 5;
int d = 6;

sum(c, d);
// Аргументы функции sum:
// a = 5
// b = 6
// (не c и d!)
```

> Следует различать фактический параметр и формальный параметр.

Передача фактических параметров

Передача фактических параметров может

Формальный параметр

Формальным параметром подпрограммы называется переменная, указанная в сигнатуре подпрограммы.

Пример:

```
int sum(int a, int b)
{
    return a + b;
}
// Формальные параметры функции sum:
// int a
// int b
```

> Следует различать фактический параметр и формальный параметр.

Функция

Функцией в языке C называется подпрограмма, тип возвращаемого значения которой **не** являющееся void.

Целочисленный тип данных

Целочисленным типом данных называется простой тип данных, множество значений которого является подмножеством целых чисел Z.

Целочисленные типы данных в C

В языке C представлены следующие целочисленные типы данных:

Тип	Пояснение	Минимальный размер в битах	Модификатор размера	Суффикс
`char`	Всегда байт. Может быть как `signed`, так и `unsigned`	8	`%c`	
`signed char`	`char`, но гарантировано знаковый.	8	`%hh`	
`unsigned char`	`char`, но гарантировано беззнаковый	8	`%hhu`	

`short`
`short int`
`signed int`
`signed short int`
	`short` больше `char`, но меньше или равен `int`	16	`%h`	

`unsigned short` `unsigned short int`	Беззнаковый `short`	16	`%hu`	

`int`
`signed`
`signed int`
	Больше или равен `short` и меньше или равен `long`	16		

`unsigned` `unsigned int`	Беззнаковый `int`	16	`%u`	`[uU]`

`long`
`long int`
`signed long`
`signed long int`
	Больше или равен `int` но меньше `long long`	32	`%l`	`[lL]`

`unsigned long` `unsigned long int`	Беззнаковый `long`	32	`%lu`	`[uUlL]`

`long long`
`long long int`
`signed long long`
`signed long long int`
	Больше `long`	64	`%ll`	`[(ll)(LL)]`

`unsigned long long` `unsigned long long int`	Беззнаковый `long long`	64	`%llu`	`[uU(ll)(LL)]`

Комментарии:

1. Конкретный размер переменной целочисленного типа данных (кроме семейства `char`) определяется машиной (неспецифицированное поведение).
2. Модификатор — см. «Модификатор размера» у спецификатора формата.
3. Суффикс — суффикс при записи литералов.
4. Суффиксы записаны в форме регулярного выражения.
5. Строго говоря, размер `char` в битах равен `CHAR_BITS`, однако стандартом определено `CHAR_BITS = 8`.

См. также:

> - преобразование типов;
> - логический тип;
> - тип перечисления;
> - size_t;
> - ptrdiff_t;
> - целые с фиксированной длиной.

Целочисленный тип фиксированной длины

Целочисленным типом фиксированной длины называется целочисленный тип, имеющий размер N битов.

Целочисленные типы фиксированной длины определены в заголовочном файле `inttypes.h`.

Знаковый целочисленный тип фиксированной длины имеет вид

```
intN_t
```

где `N` — количество бит, которые содержатся в переменной данного типа.

Беззнаковый целочисленный тип фиксированной длины имеет вид

```
uintN_t
```

где `N` — количество бит, которые содержатся в переменной данного типа.

Имеет модификаторы размера для вывода, определяемый макросом `PRI[тип][N]` в заголовочном файле `inttypes.h`, где тип — модификатор типа (`d`, `u`, `x`, `o`), а `N` — количество битов в переменной заданного целочисленного типа фиксированной длины.

Явное преобразование типа

Явным преобразованием типа в языке C называется преобразование типа при помощи операции преобразования типа.

Пример:

```
double height_cm = 175.6;
int height_m = (int)(height_cm / 100.0); // Явное преобразование типа
                                         // double в int.
// height_m = 1
```

> Важно помнить, что преобразование вещественного типа в целочисленный тип данных осуществляется *отбрасыванием дробной части числа*.

Язык ассемблера

Языком ассемблера называется язык программирования низкого уровня. Команды языка ассемблера фактически один к одному соответствуют командам процессора.

Язык программирования

Языком программирования называется формальный язык, предназначенный для записи компьютерных программ, определяющий набор лексических, синтаксических и семантических правил.