

# Типы\* языка Си

Лекция 3

\* Значений и  
функций

# План лекции

- Классификация типов данных языка Си
  - Функциональные
  - Полные – целые, с плавающей точкой, и т.п.
  - Неполные – void и ещё
  - Производные vs непроеизводные
- Представление типов в памяти
  - Представление целых и вещественных типов
- Совместимые типы, композиция типов

# Тип (данных)

- Тип (данных) – это способ доступа к значению, хранящемуся в памяти или являющемуся результатом вычислений
  - Задается выражением языка Си
- Функциональные типы – описывают функции
- Полные типы – полностью описывают объекты
- Неполные типы – описывают объекты, но не позволяют определить их размер в байтах

# Классификация целых типов

<p>Стандартные знаковые целые:</p> <ul style="list-style-type: none"><li>• signed char</li><li>• short int</li><li>• int</li><li>• long int</li><li>• long long int (C99)</li></ul>	<p>Стандартные беззнаковые целые:</p> <ul style="list-style-type: none"><li>• unsigned char</li><li>• unsigned short int</li><li>• unsigned int</li><li>• unsigned long int</li><li>• unsigned long long int (C99)</li></ul>	Стандартные целые
<p>Расширенные знаковые целые: например, <code>__int64</code> и т.п.</p>	<p>Расширенные беззнаковые целые: например, <code>__uint64</code> и т.п.</p>	
Знаковые целые	Беззнаковые целые	

# Свойства целых типов 1/2

- Диапазона `char` достаточно для представления всех элементов основного набора символов
  - В 99% случаев основной набор символов = набор ASCII из 128 элементов (1963г.)
- Символы основного набора  $\geq 0$ 
  - Знак остальных символов – implementation defined
- `sizeof(char) == sizeof(signed char)`

# Свойства целых типов 2/2

- Если  $T$  -- знаковый целый тип,  $UT$  – соотв. беззнаковый целый тип, то
  - $\text{sizeof}(T) == \text{sizeof}(UT)$
  - Диапазон неотрицательных значений  $T \subseteq$  диапазон значений  $UT$
  - Представление  $x \geq 0$  совпадает для  $T$  и  $UT$
- Вычисления со значениями типа  $UT$  выполняются по модулю  $1 +$  максимум диапазона значений  $UT$ 
  - Никогда не приводят к переполнению

# Типы с плавающей точкой

- Вещественные типы с плавающей точкой
  - float
  - double
  - long double (C99)
- Диапазон значений float  $\subseteq$  диапазон значений double  $\subseteq$  диапазон значений long double
- C99: комплексные типы с плавающей точкой `_Complex float` и т.д.

# Базовые типы, СИМВОЛЬНЫЕ ТИПЫ

- Базовые типы = char + знаковые целые типы + беззнаковые целые типы + вещественные типы
  - Базовые типы имеющие одинаковое представление все равно разные
- СИМВОЛЬНЫЕ ТИПЫ= { char, signed char, unsigned char }
- Выбор implementation defined:
  - Диапазон значений char = диапазон значений signed char
  - Диапазон значений char = диапазон значений unsigned char

# Перечислимый тип, целые типы

- Перечисление (enum) – это множество именованных целых констант
- Перечислимый тип задается перечислением
- Целые типы = { char } + знаковые целые типы + беззнаковые целые типы + перечислимые типы

# Вещественные и арифметические типы, void

- Вещественные типы = целые типы + вещественные типы с плавающей точкой
- Арифметические типы = целые типы + типы с плавающей точкой
  - Включают комплексные типы в C99
  - До C99 то же, что вещественные типы
- Тип void
  - Пустое множество значений
  - Неполный тип (не имеет размера)

# Производные типы

- Производные типы строятся из функциональных, полных и неполных типов
  - Тип может быть одновременно производным и функциональным, производным и полным, производным и неполным
- Тип-массив
- Тип-структура
- Тип-объединение
- Функциональный тип
- Тип-указатель

# Тип-массив

- Непрерывно размещенный в памяти набор элементов одного типа
- Тип элементов
  - Полный
  - Массивы неполных и функциональных типов запрещены
- Число элементов
  - Если число элементов не указано, то получается неполный тип-массив
- «Массив типа T», «целый массив», «вещественный массив», и т.п.

# Тип-структура

- Последовательно размещенная в памяти непустая последовательность именованных элементов
- Типы элементов
  - Могут быть разными
  - Все кроме последнего должны быть полными
  - Последний может быть полным или неполным типом-массивом
    - Например, `struct TCharBuffer { int Size; char Data[]; };`

# Тип-объединение

- Набор именованных значений, размещенных в памяти с перекрытием
- Типы элементов
  - Могут быть разными
  - Должны быть полными

# Функциональный тип

- Функция, возвращающая указанный тип
- Функция характеризуется
  - Возвращаемым типом
  - Числом параметров
  - Типами параметров
- «Функция, возвращающая T»

# Тип-указатель

- Полный тип, значения которого указывают (ссылаются) на значения заданного типа
  - Размер указателя известен независимо от типа указываемых значений
- Тип указываемых значений может быть любым
  - В том числе, неполным
- «Указатель на T», «указатель на int», «указатель на указатель», и т.д.

# Представление типов\* 1/2

\* в памяти

- Значение – это непрерывная последовательность байтов памяти
- Битовое поле – это непрерывная последовательность битов памяти
  - Используются довольно редко
- Размер значения (битового поля) – это длина этой последовательности
- Значения битовых полей хранятся внутри значений целых типов
  - Для беззнаковых битовых полей хранится двоичная запись

# Представление типов 2/2

- Для `unsigned char` хранится двоичная запись значения
- Любое значение типа `T` размером `N` байтов можно скопировать в массив типа `unsigned char[N]`
  - Значение этого массива называется двоичным представлением значения типа `T`
  - Значения, отличные от NaN-ов и имеющие одинаковое двоичное представление, равны
  - Равные значения могут иметь разное двоичное представление
- Некоторые значения типа `unsigned char[N]` могут не быть двоичным представлением никакого значения типа `T` – это т.н. особые значения
- Доступ и изменение двоичного представления особых значений иначе, чем через символьный тип, ведет к `undefined behavior`

# Представление структур и объединений

- Значение структур и объединений может содержать выравнивающие байты
  - Значения выравнивающих байтов не определены
- Значение структуры и объединения никогда не является особым
  - Даже если значение какого-то их элемента является особым
- Двоичное представление элемента объединения может быть короче, чем двоичное представление всего объединения
  - Значение неиспользуемых байтов объединения не определено

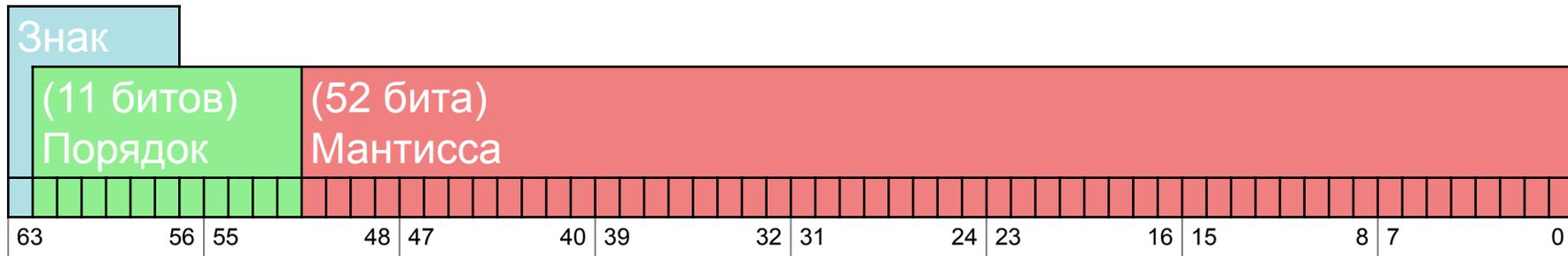
# Представление беззнаковых целых ТИПОВ

- Двоичное представление беззнакового целого типа != unsigned char делится на
  - Значащие биты (обязательно)
    - Значащие биты представляют степени 2 от 1 до  $2^{(\text{число значащих битов} - 1)}$
  - Выравнивающие биты (как правило отсутствуют)
    - Значение выравнивающих битов не определено
- Двоичное представление unsigned char не содержит выравнивающих битов
  - См. предыдущие слайды про представление типов

# Представление знаковых целых типов

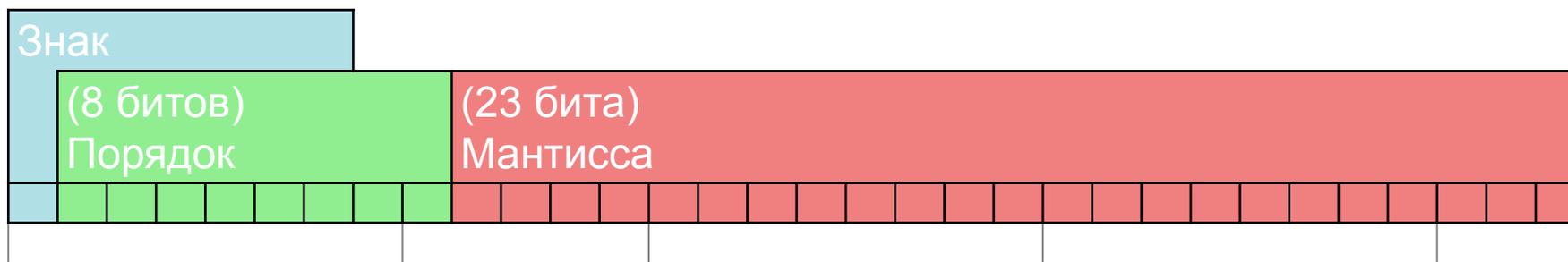
- Значащие биты (обязательно)
  - Значащие биты представляют степени 2 от 1 до  $2^{(\text{число значащих битов} - 1)}$
- Знаковый бит (обязательно)
  - 0 --> значением является число  $C$ , записанное в значащих битах
  - 1 --> значением является число
    - $-C$  – «знак и абсолютная величина»
    - $C - 2^{(\text{число значащих битов})}$  – «дополнительный код»
    - $C - 2^{(\text{число значащих битов})} + 1$
- Выравнивающие биты (как правило отсутствуют)
  - Значение выравнивающих битов не определено
- Implementation defined -- выбор представления отрицательных чисел
  - В 99% случаев используется дополнительный код
- Implementation defined – возможные особые значения
  - Если доп. код, то знак=1 + значащие биты=0
  - Если «знак и абсолютная величина», то знак=1 + значащие биты=0
  - Иначе знак=1 + значащие биты=1
- Если значение не является особым и не доп. код, то это ноль со знаком

# Представление double – стандарт IEEE 754



Порядок	Мантисса 0	Мантисса != 0	Формула
0x000	0 и -0	Денормализов. числа	$(-1)^{\text{знак}} \cdot 2^{\text{порядок}-1022} \cdot (0.\text{мантисса})_{(2)}$
0x001 ... 0x7fe	Нормализованные числа		$(-1)^{\text{знак}} \cdot 2^{\text{порядок}-1023} \cdot (1.\text{мантисса})_{(2)}$
0x7ff	$+\infty$ или $-\infty$	NaN	
$3\text{ff}0\ 0000\ 0000\ 0000_{(16)} = 1$	$0000\ 0000\ 0000\ 0000_{(16)} = 0$	$7\text{ff}0\ 0000\ 0000\ 0000_{(16)} = \infty$	
$3\text{ff}0\ 0000\ 0000\ 0001_{(16)} \approx 1.0000000000000002$	$8000\ 0000\ 0000\ 0000_{(16)} = -0$	$\text{ff}f0\ 0000\ 0000\ 0000_{(16)} = -\infty$	
		$3\text{fd}5\ 5555\ 5555\ 5555_{(16)} \approx 1/3$	

# Представление float – стандарт IEEE 754



Порядок	Мантисса 0	Мантисса != 0	Формула
0x00	0 и -0	Денормализов. числа	$(-1)^{\text{знак}} \cdot 2^{\text{порядок}-126} \cdot (0.\text{мантисса})_{(2)}$
0x01 ... 0xfe	Нормализованные числа		$(-1)^{\text{знак}} \cdot 2^{\text{порядок}-127} \cdot (1.\text{мантисса})_{(2)}$
0xff	$+\infty$ или $-\infty$	NaN	

# Совместимость struct, union, enum 1/2

- Типы T1 и T2 совместимы, если выполнены условия
  - T1 и T2 не имеют тэга, либо тэг совпадает
  - T1 и T2 являются полными
  - Элементы T1 и T2 взаимно однозначно соответствуют друг другу
  - В каждой паре соответствующих элементов
    - Типы являются совместимыми
    - Имена совпадают
    - Если элементы пары – это битовые поля, то их ширина совпадает
    - Если T1 и T2 являются enum, то элементы пары имеют одинаковое значение
  - Если T1 и T2 являются struct, то порядок элементов в T1 и в T2 совпадает
- Спецификаторы типа, квалификаторы типа, и деклараторы могут дополнительно ограничивать совместимость

# Совместимость struct, union, enum 2/2

- Если объект или функция имеют внешнее связывание, то они должны быть объявлены с совместимыми типами
- В противном случае поведение программы не определено

# Композиция типов 1/2

Композицией совместимых типов T1 и T2 называется тип, построенный по правилам:

- Если T1 (или T2) – это массив фиксированного размера, то T1 (соотв. T2)
  - C99: Если T1 (или T2) – это массив переменного размера, то T1 (соотв. T2)
- Если T1 (или T2) – это прототип функции (т.е. функция с списком формальных параметров), то T1 (соотв. T2)
- Если T1 и T2 – это прототипы функций, то тип формального параметра композиции является композицией типов формальных параметров
- Если T1 и T2 – это производные типы, то правила применяются рекурсивно к типам, от которых произведены T1 и T2

# Композиция типов 2/2

- Единица трансляции А

```
int f(int(*)(), double(*)[3]);
```

- Единица трансляции Б

```
int f(int*)(char *), double(*)[]);
```

- Композиция типов функции f

```
int f(int*)(char *), double(*)[3]);
```

# Заключение

- Классификация типов данных языка Си
  - Функциональные
  - Полные – целые, с плавающей точкой, и т.п.
  - Неполные – void и ещё
  - Производные vs непроеизводные
- Представление типов в памяти
  - Представление целых и вещественных типов
- Совместимые типы, композиция типов