ТИПЫ ДАННЫХ В ЯЗЫКЕ С

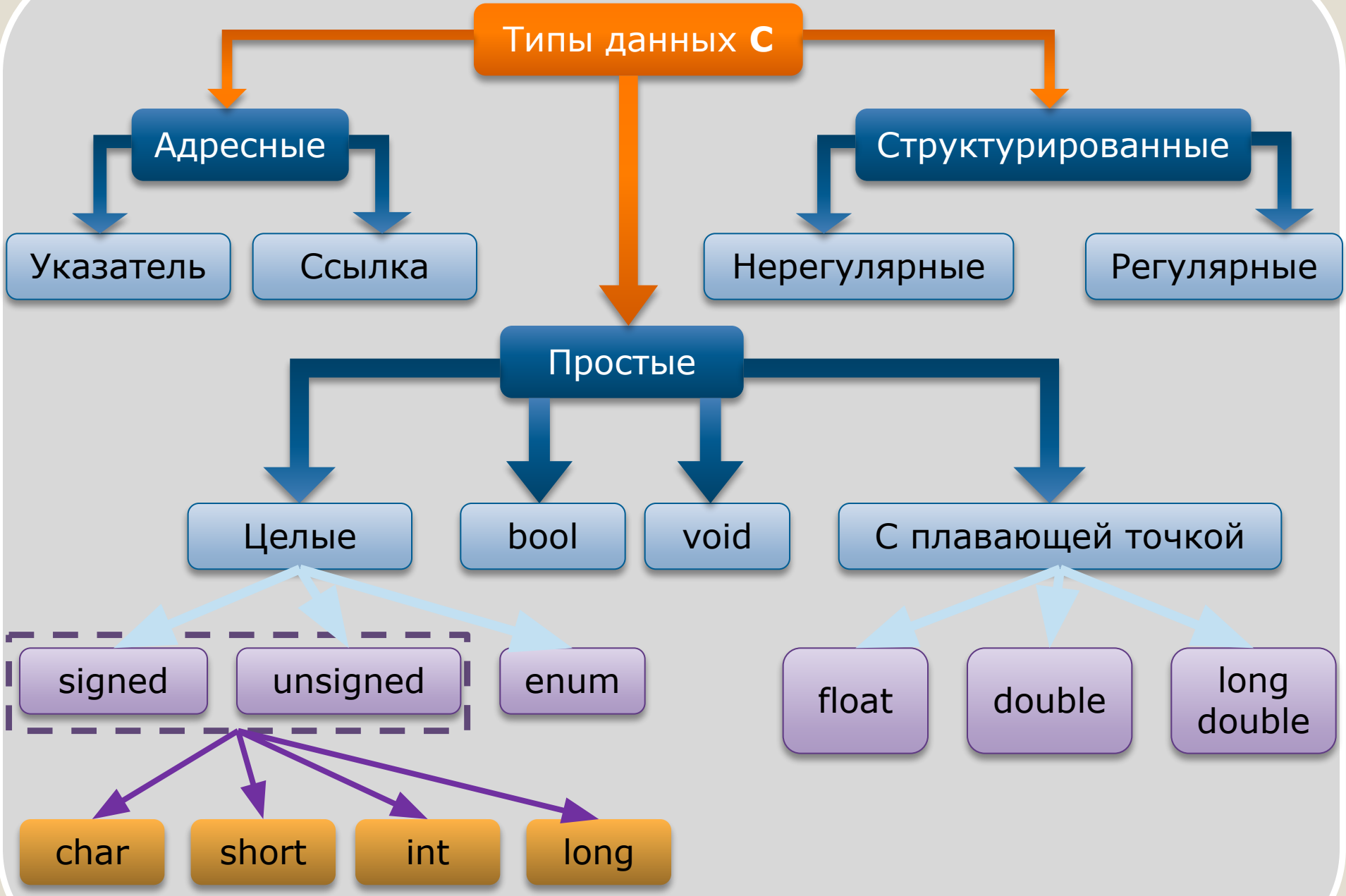
ТИПЫ ДАННЫХ В ЯЗЫКЕ С.

Все типы языка **С** *можно разделить на* ***основные*** *и* ***составные****.*

В языке **С** *определено* ***шесть основных типов данных*** *для представления целых, вещественных, символьных и логических величин.*

На основе этих типов программист может вводить описание ***составных типов****.*

К составным типам относятся ***массивы, перечисления, функции, структуры, ссылки, указатели, объединения и классы.***


Типы данных C
Адресные
Структурированные
Указатель
Ссылка
Нерегулярные
Регулярные
Простые
Целые
bool
void
С плавающей точкой
signed
unsigned
enum
float
double
long double
char
short
int
long

Основные типы данных.

- Основные (*стандартные*) типы данных:
  - **int** (целый)
  - **char** (символьный)
  - **wchar_t** (расширенный символьный)
  - **bool** (логический)
  - **float** (вещественный)
  - **double** (вещественный с двойной точностью)
  - **void** (пустой тип)

Основные типы данных часто называют *арифметическими,* поскольку их можно использовать в арифметических операциях.

для уточнения формата внутреннего представления и диапазона значений стандартных типов существует четыре ***спецификатора типа*.**

- спецификаторы типа:
  - **short** (короткий)
  - **long** (длинный)
  - **signed** (знаковый)
  - **unsigned** (беззнаковый)

Спецификаторы типа.

Целый тип (int).

Размер типа **int** не определяется стандартом, а зависит от компьютера и компилятора.

- Для 16-разрядного процессора под величины этого типа отводится 2 байта, для 32-
- независимо от разрядности процессора

Спецификатор **short** перед именем типа указывает компилятору, что под число требуется отвести 2 байта.

- независимо от разрядности процессора

Спецификатор **long** означает, что целая величина будет занимать 4 байта.

- независимо от разрядности процессора

Таким образом, на 16-разрядном компьютере эквиваленты **int** и **short int**, а на 32-разрядном — **int** и **long int**.

Внутреннее представление

При использовании спецификатора **signed** старший бит числа интерпретируется как знаковый

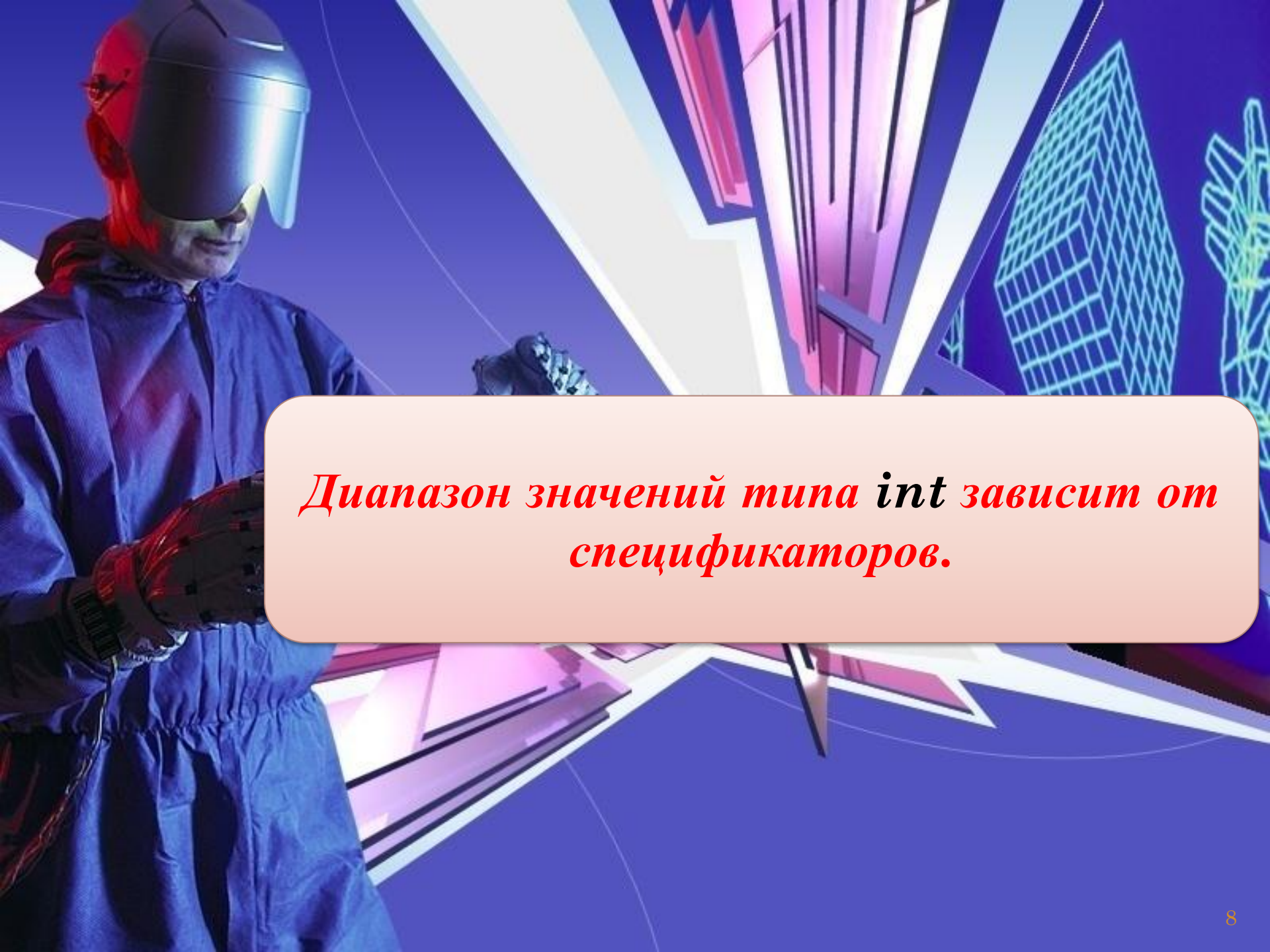
- 0 — положительное число, 1 — отрицательное

Спецификатор **unsigned** позволяет представлять только положительные числа

- поскольку старший разряд рассматривается как часть кода числа

По умолчанию *все целочисленные типы считаются знаковыми, то есть спецификатор* ***signed*** *можно опускать.*

Целый тип (int).



Диапазон значений типа **int** *зависит от спецификаторов.*

Константам, встречающимся в программе, приписывается тот или иной тип в соответствии с их видом.

Если этот тип по каким-либо причинам не устраивает программиста, он может явно указать требуемый тип с помощью суффиксов.

- L, l (long) и U, u (unsigned)
- например, константа 32L будет иметь тип long и занимать 4 байта

Можно использовать суффиксы L и U одновременно

- например, 0x22UL или 05Lu.

*Типы **short int, long int, signed int** и **unsigned int** можно сокращать до **short, long, signed** и **unsigned** соответственно.*

Целый тип (int).

Под величину символьного типа отводится количество байт, достаточное для размещения любого символа из набора символов для данного компьютера.

- Как правило, это 1 байт.

Тип **char**, как и другие целые типы, может быть со знаком или без знака.

- **signed -** можно хранить значения в диапазоне от -128 до 127.
- **unsigned -** значения могут находиться в пределах от 0 до 255.

Величины типа **char** применяются также для хранения целых чисел, не превышающих границы указанных диапазонов.

Символьный тип (char).

Тип **wchar_t** предназначен для работы с набором символов, для кодировки которых недостаточно 1 байта

- например, **Unicode**

Размер этого типа зависит от реализации

- как правило, он соответствует типу **short**

Строковые константы типа **wchar_t** записываются с префиксом **L**

- например, L"Game"

Расширенный символьный тип (wchar_t).

Величины логического типа могут принимать только значения **true** и **false**, являющиеся зарезервированными словами

- Внутренняя форма представления значения **false** — **0** (нуль).
- Любое другое значение интерпретируется как **true**.

*При преобразовании к целому типу **true** имеет значение **1**.*

Логический тип (bool).

Типы данных с плавающей точкой
(**float**, **double** и **long double**)
хранятся в памяти компьютера иначе, чем целочисленные

- Внутреннее представление вещественного числа состоит из двух частей — **мантиссы** и **порядка**.

Мантисса — это число, большее **1.0**, но меньшее **2.0**

- Поскольку старшая цифра **мантиссы** всегда равна **1**, она не хранится.
- Длина **мантиссы** определяет точность числа, а длина **порядка** — его диапазон.

Типы с плавающей точкой.

тип	размер (байт)	мантисса (бит)	порядок (бит)
float	4	23	8
double	8	52	11
long double	10	64	15

Типы с плавающей точкой.

Константы с плавающей точкой имеют по умолчанию тип **double**

Можно явно указать тип константы с помощью суффиксов ***F, f (float)*** *и* ***L, l (long)***

- *Например,*
  - *константа* **2E+6L** *будет иметь тип* ***long double***
  - *константа* **1.82f** *— тип* ***float***

Типы с плавающей точкой.

Тип	Диапазон значений	Размер (байт)
bool	true и false	1
signed char	–128 … 127	1
unsigned char	0 … 255	1
signed short int	–32 768 … 32 767	2
unsigned short int	0 … 65 535	2
signed long int	–2 147 483 648 … 2 147 483 647	4
unsigned long int	0 … 4 294 967 295	4
float	3.4e–38 … 3.4e+38	4
double	1.7e–308 … 1.7e+308	8
long double	3.4e–4932 … 3.4e+4932	10

Сводная таблица.

Кроме перечисленных, к основным типам языка относится тип **void**

- множество значений этого типа пусто

Тип **void** используется:

- для определения функций, которые не возвращают значения,
- для указания пустого списка аргументов функции,
- как базовый тип для указателей и в операции приведения типов

Тип void.

Примечание

В стандарте ANSI диапазоны значений для основных типов не задаются, определяются только соотношения между их размерами

- sizeof(float) < sizeof(double) < sizeof(long double)
- sizeof(char) < sizeof(short) < sizeof(int) < sizeof(long)

Минимальные и максимальные допустимые значения зависят от реализации и приведены в заголовочных файлах:

- *для целых типов — **<limits.h> (<climits>)**,*
- *для вещественных типов — **<float.h> (<cfloat>)**,*
- *а также в шаблоне класса **numerlc_limits***

Структура программы в языке С

Программа на языке С состоит из:

- *функций*
- *описаний*
- *директив препроцессора*.

Простейшее определение функции имеет следующий формат:

```
тип_возвращаемого_значения имя ([ параметры ]){

    операторы, составляющие тело функции

}
```

Как правило, функция используется для вычисления какого-либо значения, поэтому перед именем функции указывается его тип.

Простейшее определение функции

Функции мы рассмотрим несколько позже, ниже приведены лишь самые необходимые сведения

- если функция не должна возвращать значение, указывается тип void;
- тело функции является блоком и, следовательно, заключается в фигурные скобки;
- функции не могут быть вложенными;
- каждый оператор заканчивается точкой с запятой (кроме составного оператора).

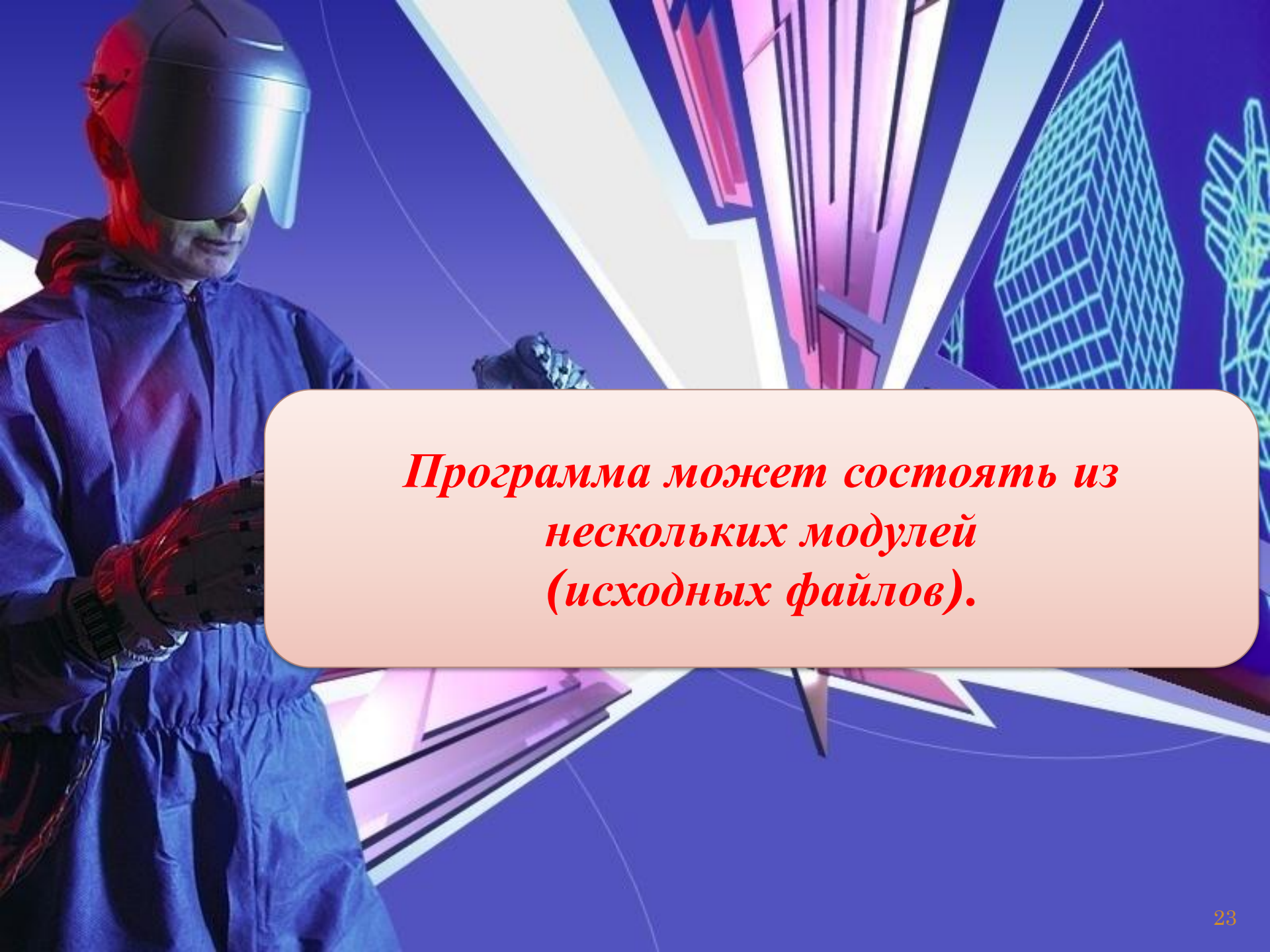
Простейшее определение функции

директивы препроцессора

описания

```
int main(){
операторы главной функции
}
int f1(){
операторы функции f1
}
int f2(){
операторы функции f2
}
```

Пример структуры программы



Программа может состоять из нескольких модулей (исходных файлов).

В языке C++ нет встроенных средств ввода/вывода

- **ввод/вывод** осуществляется с помощью функций, типов и объектов, содержащихся в стандартных библиотеках.

Используется два способа:

- функции, унаследованные из языка C,
- объекты C++.

Замечания о вводе/выводе

Основные функции ввода/вывода в стиле С:

- **int scanf (const char* format, ...) // ввод**
- **int printf(const char* format, ...) // вывод**

Данные функции выполняют форматированный ввод и вывод произвольного количества величин в соответствии со строкой формата **format**.

- Строка формата содержит:
  - ***символы***, *которые при выводе копируются в поток (на экран) или запрашиваются из потока (с клавиатуры) при вводе,*
  - ***спецификации преобразования***, *начинающиеся со знака* **%**, *которые при вводе и выводе заменяются конкретными величинами.*

Пример программы, использующей функции ввода/вывода в стиле C:

```
#include <stdio.h>
int main(){
    int i;
    printf("Введите целое число\n");
    scanf("%d", &i);
    printf("Вы ввели число %d, спасибо!", i);
    return 0;
    }
```

ввод/вывод в стиле C

ввод/вывод в стиле C

А вот как выглядит та же программа с использованием *библиотеки классов* **C++**:

```
#include <iostream.h>
int main(){
    int i;
    cout << "Введите целое число\n";
    cin >> i;
    cout << "Вы ввели число " << i << ", спасибо!";
    return 0;
    }
```

ввод/вывод в стиле C++

Заголовочный файл **<iostream.h>** содержит описание набора классов для управления вводом/выводом.

- В нем определены стандартные объекты-потоки:
  - **cin** для ввода с клавиатуры
  - **cout** для вывода на экран
- а также операции:
  - помещения в поток **<<**

В дальнейшем изложении будут использоваться оба способа

- но в одной программе смешивать их не рекомендуется

ввод/вывод в стиле C++

Модификаторы формата применяются для управления шириной поля, отводимого для размещения значения.

Модификаторы — это одно или два числа

- первое задает минимальное количество позиций, отводимых под число
- второе — сколько из этих позиций отводится под дробную часть числа (точность)

Модификаторы формата

Если указанного количества позиций для размещения значения недостаточно, автоматически выделяется большее количество позиций:

- **%-minC** или **%minC**;
- **%-minC.precisionC** или **%min.precisionC.**

C	спецификация формата
min	десятичное число, задающее минимальную ширину поля
precision	десятичное число, смысл этого модификатора зависит от спецификации формата, с которой он используется
Символ минус (-)	указывает на то, что значение выравнивается по левому краю и, если нужно, дополняется пробелами справа.
При отсутствии минуса значение выравнивается по правому краю и дополняется пробелами слева.	

Модификаторы формата

Модификатор precision

при выводе строки	• спецификация **%s** — **precision** указывает максимальное число символов для вывода • спецификации **%f** или **%e** — **precision** указывает количество цифр после десятичной точки
при выводе вещественного числа	• спецификации **%d** или **%G** — **precision** указывает максимальное количество значащих цифр, которые будут выведены • спецификации **%d** или **%i** — **precision** указывает минимальное количество выводимых цифр • если число представляется меньшим числом цифр, чем указано в **precision**, выводятся ведущие (начальные) нули
при выводе целого числа	

Модификаторы формата

*Перед спецификацией могут использоваться префиксы **l** и **h***

h

- *с типам и **d, i, o, x, X** указывает на тип аргумента **short int***
- *с типом **u** – **unsigned short int***

l

- *с типом **u** – **unsigned long int***
- *с типам и **e, E, f, g, G***

- *например, **%lf, %hu***

Специ-фикация	Пояснение
c	*аргумент рассматривается как отдельный символ*
d, i	*аргумент преобразуется к десятичному виду*
e, E	*аргумент, рассматриваемый как переменная типа float или double, преобразуется в десятичную форму в виде* **[-]m.nnnnnne[±]xx**, *где длина строки из* ***n*** *определяется указанной точностью.* *Точность по умолчанию равна 6.*
f	*аргумент, рассматриваемый как переменная типа float или double, преобразуется в десятичную форму в виде* **[-]mmm.nnnnn**, *где длина строки из* ***n*** *определяется указанной точностью.* *Точность по умолчанию равна 6*
g,G	*используется формат* %e *или* %f, *который короче; незначащие нули не печатаются*

Модификаторы формата

Специ-фикация	Пояснение
o	*аргумент преобразуется в беззнаковую восьмеричную форму (без лидирующего нуля)*
p	*вывод указателя в шестнадцатеричном формате (эта спецификация не входит в стандарт)*
s	*аргумент является строкой: символы строки печатаются до тех пор, пока не будет достигнут нулевой символ или не будет напечатано количество символов, указанное в спецификации точности*
u	*аргумент преобразуется в беззнаковую десятичную форму*
x, X	*аргумент преобразуется в беззнаковую шестнадцатеричную форму (без лидирующих Ox)*
%	*выводится символ* **%**

Модификаторы формата

Пример:

```
#include <stdio.h>
int main(){
	int int1 = 45,  int2 = 13;
	float f = 3.621;
	double dbl =2.23;
	char ch = 'z', *str = "ramambahari";
		printf("int1 = %d|  int2 = %3d| int2 = %-4d|\n", int1, int2,
int2);
		printf("int1 = %X|  int2 = %3x| int2 = %4o|\n",   int1,  int2,
int2);
		printf("f = %f|  f = %4.2f|  f = %6.1f|\n", f, f, f);
		printf("f = %g| f = %e| f = %+E|\n", f, f, f);
		printf("dbl = %5.2lf| dbl = %e| dbl = %4.1G|\n", dbl, dbl, dbl);
		printf("ch = %c| ch = %3c|\n", ch, ch);
		printf("str = %14s|\nstr = %-14s|\nstr = %s|\n", str, str, str);
return 0;
}
```

Модификаторы формата

Результат работы программы:

```
int1  = 45| int2 =         13|  int2 = 13    |
int1 =  2D| int2 =          d|  int2 = 15    |
f = 3.621000 |  f = 3.62     |  f =    3.6   |
f = 3.621    |  f = 3.621000e+000          |  f = +3.621000E+000|
dbl = 2.23   |  dbl = 2.230000e+000        |  dbl = 2             |
ch = z|   ch = z   |
str =             ramambahari|
str = ramambahari            |
str = ramambahari |
```

Модификаторы формата