Система типов

1) int (-32767 ... 32767) целое число с таким диапазоном но по факту 32 bit

в реальности (не в с/сп) (-32768 ... 32767) - дополнение до двух

2) Short (int) (можно без int) - не больше чем int - 16 bit

3) long (int) (можно без int) - на винде: 32 bit

на Unix: 32-bit на x32

64-bit на x64

4) long long (int) - 64 bit

5) char - 8 bit (типа байт) уе-то может быть 32 bit

byte - min адресуемая память

signed char → (-128...127)

unsigned char → (0 ... 255)

! можно слева/справа unsigned с тем же размером но в плюс

6) _Bool (на C) 8 bit т.к. меньше не адресовать (у 1 бита нет адреса)

можно bool при #include <stdbool.h> + добавится True и False

Это все без подключения доп. файлов

7) size_t / ptrdiff_t при #include <stddef.h>

size_t - (беззнаковый) тип соотв. битности программы (для индексации по большим массивам) (32 bit на 32 бит. коде и 64)

ptrdiff_t - знаковый аналог

2) при <stdint.h>:

- int X_t , X: 8, 16, 32, 64
- (знаковый / без) uint X_t ,

3) только в C23
_BitInt(X), X - любое кол-во бит которое поддерживает компилятор
(signed и unsigned)

Как писать/читать и константы:

1) X:

unsigned int ↓ long ↓

X = 3; можно 3u; или 3l; или 3llu; и т.д.

↑ значения которые помещаются в int/long/long long

2) в _BitInt:

X = 3wb; или 3uwb;

3) printf("%i", X)

или %d ← число со знаком

без знака: %u

%x вывод в 16-ой системе

и

%X печать больших или маленьких букв

Есть тип: short то ко всему %h{...}
char -//- //- //- //- %hh{...}
long -//- //- //- //- %l{...}
long long -//- //- //- //- %ll{...}
size_t -//- //- //- //- %z{...}
ptrdiff_t -//- //- //- //- %t{...}

intX_t то: #include <inttypes.h>

+ printf("%", space PRI{...}X, X)

"u/x/X"

4) scanf("%" SCN{...}X, &X); если IntX_t

Иначе:

int: scanf("%i", &X);
(для 10 с.с и 16 сс. но и 8 с.с.)

или

%d - 10 с.с.

или

%X - 16 с.с.

X = 20 ≡ X = 0X14 = 1aB (16 с.с. 16 с.с.), но X = 010 (8 с.с т.к. префикс 0)

В C_{23}: X = 0b10 11; (2 с.с.)

5) __int128 нельзя (побьют)

Стандарт типы IEEE - 754

1) float = single pr. 32 bit и точность 24 bit. $10^{\approx 38}$

2) double = double pr. 64 bit; точность 53 bit; $10^{\approx 308}$

3) long double = double pr. (64 bit) / quad pr. (128 bit) / extended pr. (80 bit)

по дефолту дроби - double; X = 1,5f; (float) или X = 1,5ld; (long double)

3 _Float16 half pr. но не везде

Вывод:

%f - float, %lf - double, %Lf - long double

Поддержка комплексной арифметики:

_Complex float X = 1.5f - 2.3if; (float; мнимая часть)

очень точная, но медленная. За счет команд компилятору можно ускорить

в C++ нет унитарных, но есть фантик (=> нельзя ускорить) но в Clang есть арифметика из C

```
int x = -2;
```


не существует отриц. констант

Т int 8bit, то −128 не лезет в int => идет в long long

для чисел без знака унарный тоже самый

```
int x = -2u;
```

```
int x, y;
typedef unsigned int uint;
```

создается тип uint

```
uint x; = unsigned int x;
```