Немного про типы

```
auto x;
if ( )
   x = f();
else
   x = g();
```

type of_unqual
(тот же самый type_of но только без модификаций)

```
typeof(f()) x;
if (...)
   x = f();
else:
   x = g();
```

```
int * const p = &a;
typeof(p)  z = &b;   (полностью копирует тип)
typeof_unqual(p) y; // int *  (снимает модификаторы верхнего уровня)
const int * const p = &a;
typeof(*p) // const int  (Т.к. убрали указатель)
```

(13) в C++ исп-ся decltype (там все не так просто)

```
int a[256], b[256], c[256]
for (size_t i=0, i<256, i++)
{
   c[i] = a[i] + b[i]
}
```

размер массива / размер элемента
sizeof(a) / sizeof(*a)

! работает только с массивами

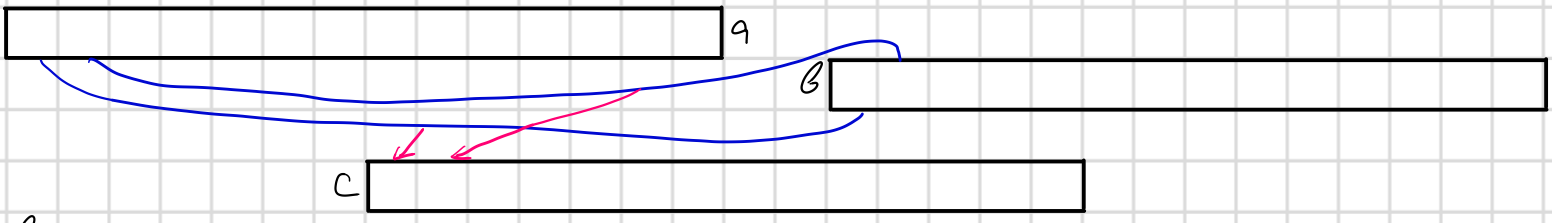
будет работать хорошенько :)
(будет выполнять по 4 команды и 64 итерации)

```
void f(const int *a, const int *b, int *c)
for (size_t i=0, i<256, i++)
{
   c[i] = a[i] + b[i]
}
```

будет плёха :(
(Компилятор не поставит плюшки)

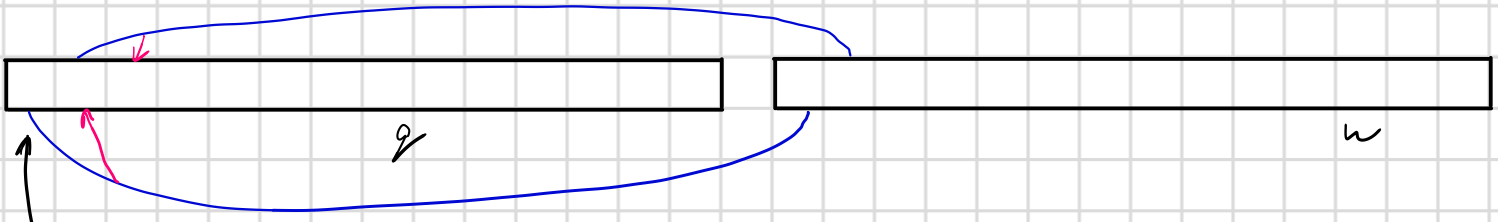
```
int q[257], w[256]
f(w, q, q+1)
```

Бодренько:



Все без проблем т.к. не пересекаются

Плохо:



Не можем взять 4 или более пары для вычисления

Как фиксить, зная что мы не пересекаем массивы?

```
void f(const int *restrict a, const int *restrict b, int *restrict c)
```

restrict - даем обещание компилятору и он оптимизирует (указатель не перекрывается с другими)

(13) если указатели пересекаются, то все равно можно restrict, но при условии что данные не изменяются

(13) в C → restrict

в C++ → __restrict

Строки

Char – тип с неизвестным знаком

Вместо можно uint8_t / int8_t

Char c = 0x30 или '0' ← символ

Код символа '0'

явная длина

"ABC" <=> 'A', 'B', 'C', 0 ← Хранение строки

Значение (означает что строка закончилась (нул для длины строки))

неявная длина

"ABC" <=> 'A', 'B', 'C' и длина строки (3) (неизвестно где хранится)

Char * , но "ABC" → const char *

Как посчитать длину строки руками?

```
size_t strlen(const char *p)
{
    size_t
    for(x=0; p[x]; x++);
    return x;
}
```

(1)

или

```
size_t strlen(const char *p)
{
    const char *z=p
    while(*z)
        z++;
    return z-p
}
```

(2)

Нахождение первого вхождения

```
const char *strchr(const char *p, char c)
{
    size_t
    for (x=0; p[x]≠c; x++)
    if p[x]==c: return x
    else: nullptr
}
```

~

```
Char *a...
```

делаем каст ↙ возвращает const ↙

```
Хотим: char *b=(char *)strcht(a, 'i');
```

~

```
int strcmp(const char *a, const char * b)
```

Сравнение строк ↗

0: равны, 0< если лексикограф. a>b, >0 b>a

a ≤ b ⟺ strcmp(a,b) ≤ 0

~

```
% c
Char c=0x30
printf("%c", c); → 0
printf("%i", c); → 48 (т.к. char - целый)
```

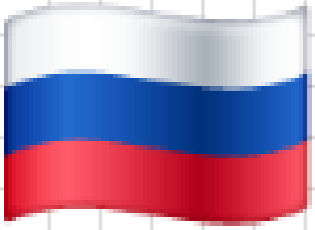
%s (напечатать символьную строку) (идет до нуль-терминатора и печатает %c)

```
char *p;
printf("%s", p) и scanf("%s", p)
```

нет &

↑ читает до пробельного символа

Кодировки



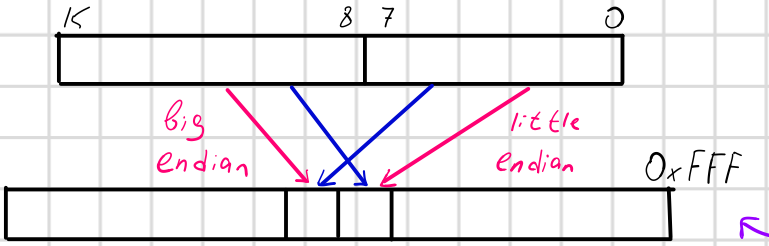
ASCII 0...127 128...255 (для доп. символов)

Codepage (cp866, cp1251, КОИ-8) ✅

DBCS (double byte character set) Кодировка с плавающей размерностью (для доп. языков)

Unicode (символы кодируются 16 битами)

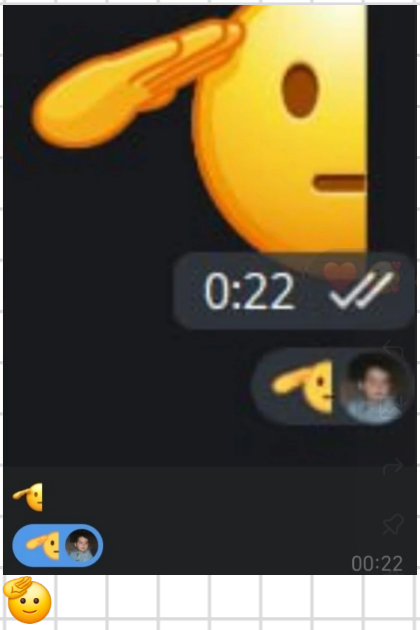
Unicode 2 (есть 16 бит для символа целиком, а есть 16 бит начало и 16 конец (2 символа)



BOM 0xFEFF ❌

↪ UTF-16

(13) Большинство железа использует little endian (иначе нужно перед ставлять байты)



VCS-2 ← убого :)

UTF-8 ← божественно 

- совпадает кодами с ASCII
- на символ 1-4 байта (Теперь байты а не биты)
- кодируем все языки нормально

❗ Использовать UTF-8 и не страдать

Сборка программы

Файл a.c

```
int f(int);

int main(void)
{
   return f(3);
}
```

Файл b.c

```
int f(int x)
{
   return x+2;
}
```

a.c → a.obj
b.c → b.obj
→ my.exe

```
int main(void)
{
   printf("%i\n", f(3));
   return 0;
}
```

линкер должен собрать printf.
Это ищется в .lib (стандартные системные ф-ии)

a.c → a.obj
b.c → b.obj
.lib
→ my.exe

ссылка

gdi32.dll → MessageBox (ищем динамические библиотеки)

Как сделать динамическую библиотеку?

расширение: .dll (Win) .so (Unix)

можно скомпилировать {...}.c в {...}.dll

Компиляция

Компиляция:

```
clang -c main.c -o main.obj -m64 -O2 -std=c23
```

- `-c` — compile
- `-o` — указываем имя
- `-m64` — битность сборки
- `-O2` — стандартные оптимизации
- `-std=c23` — стандарт

Вызов линкера:

```
lld-link /subsystem:console /defaultlib:libcmt main.obj /out:main.exe /MACHINE:X64
```

`/MACHINE:X64` — битность

для 32 бит: 32 или 86

(13) если не указывать `-c` при компилировании, то линкер не нужен + нужно имя.exe

! нужно научиться компилить полным образом