Основы языка C

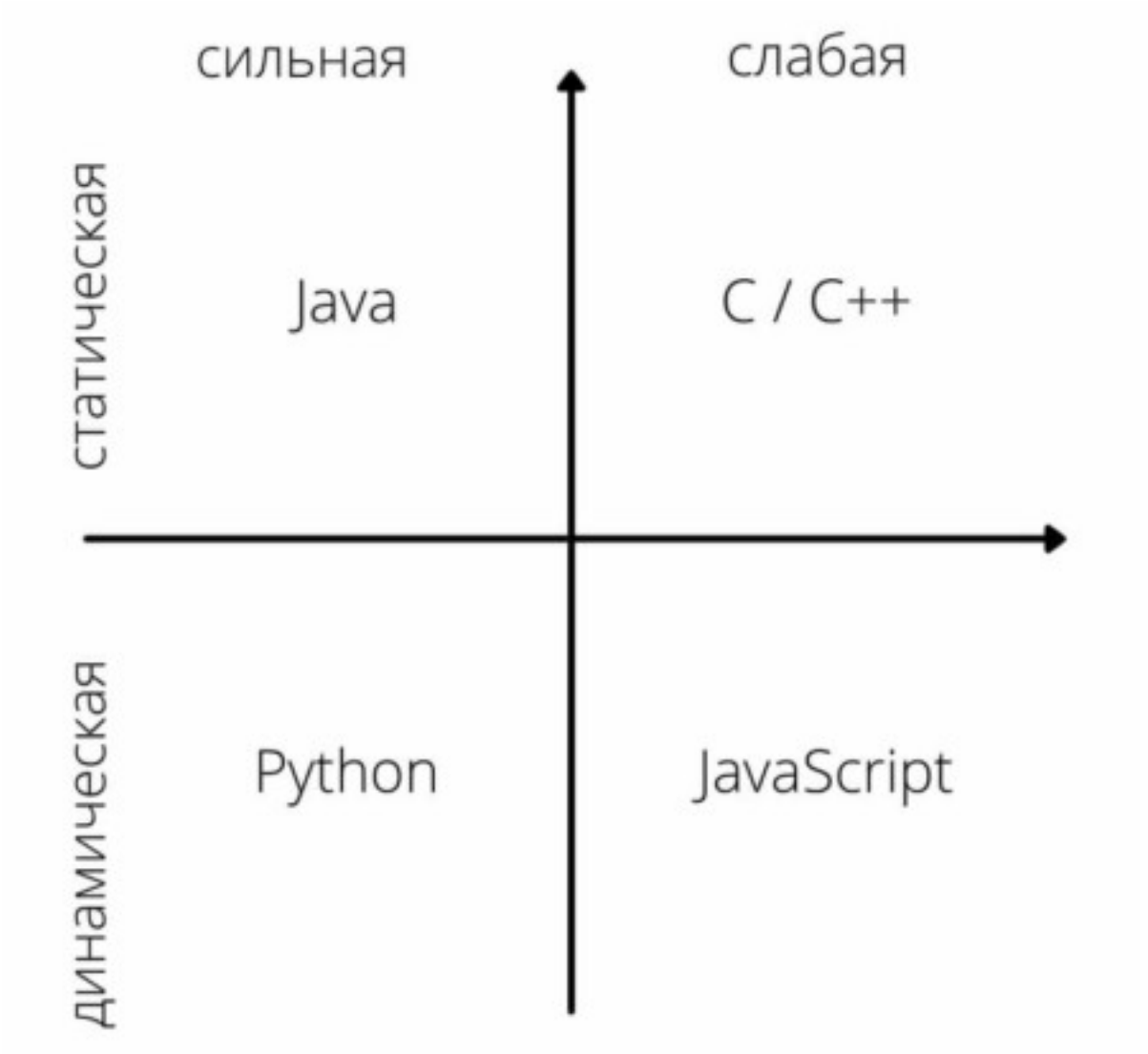
Выражения и операторы

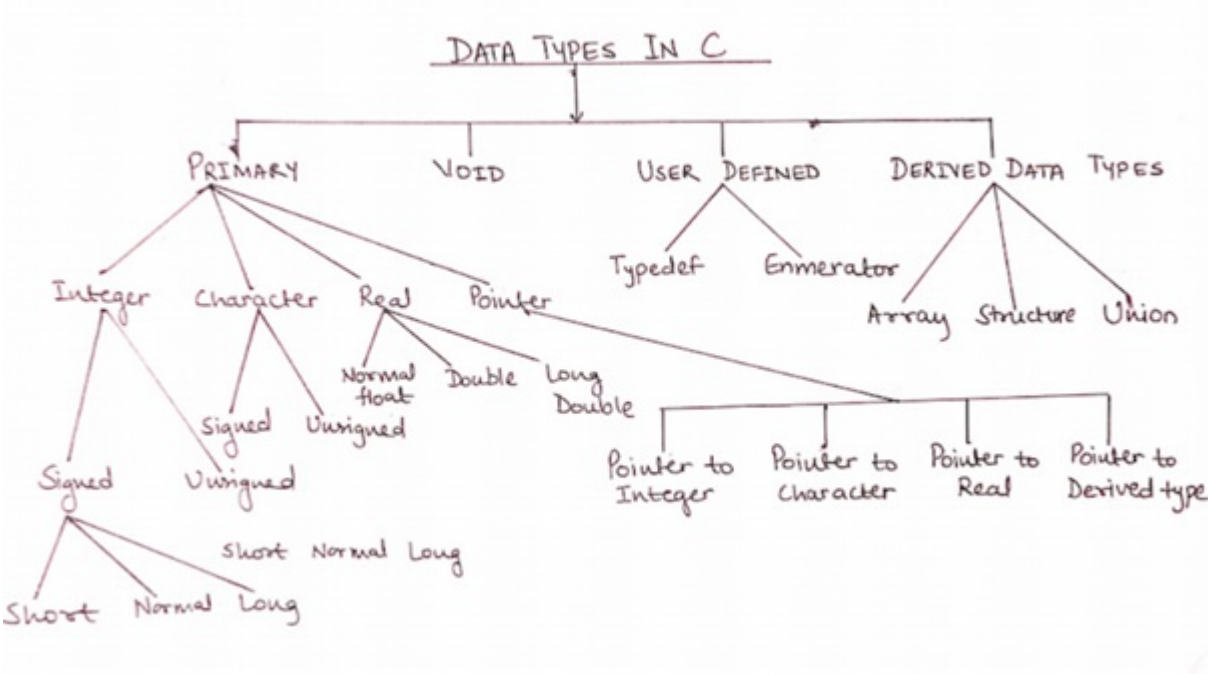
- выражение (expression): нечто, что вычисляется и имеет значение, например, `1 + 2.0`
- оператор (statement): нечто, что выполняется, например, `if(argc != 2) exit(1);` или `while(true) { puts("Привет"); }`

Условное выражение: `x > 0 ? x : -x`

Типы данных

C — язык с явной, номинативной, слабой статической типизацией:





Базовые типы данных

- K&R C:
  - `int`
  - `char`
  - `float`
  - `double`
  - `short`
  - `long`
  - `unsigned`
- C90:
  - `signed`
  - `void`
- C99:
  - `_Bool`
  - `_Complex`
  - `_Imaginary`

Целочисленные константы:

- `10` - число типа `int`
- `10U` - число типа `unsigned int`
- `10L` - число типа `long`
- `10UL` - число типа `unsigned long`
- `10LL` - число типа `long long`
- `10ULL` - число типа `unsigned long long`

Спецификаторы формата для `printf`:

- `int`: `%d`, `%u`, `%o`, `%x`
- `long`: `%ld`, `%lu`, `%lo`, `%lx`
- `long long`: `%lld`, `%llu`, `%llo`, `%llx`
- `short`: `%hd`, `%hu`, `%ho`, `%hx`

$53_{10} = 110101_2$

$-53_{10} = -110101_2$

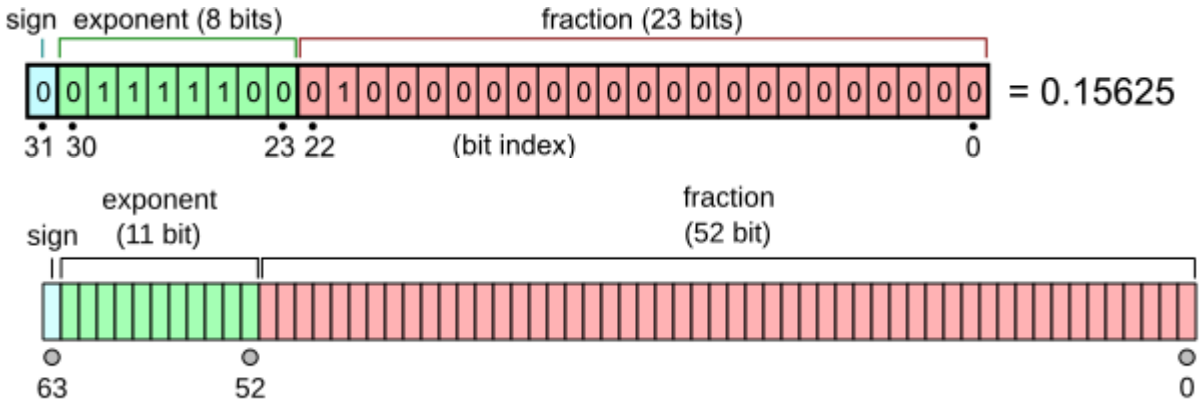
Дополнительный код

Типы фиксированной длины (`inttypes.h` / `stdint.h`)

- `int8_t`
- `int_least8_t`
- `int_fast8_t`
- `intmax_t`
- `PRId8`

Вещественные числа:

IEEE 754



- `float` - 1 бит знака, 8 бит экспоненты и 24 бита мантиссы, 7 значащих цифр
- `double` - 1 бит знака, 11 бит экспоненты, 52 бита знака, 16 значащих цифр
- `long double`

Floating point visually explained

```c
float f = 5.0f;
float e = -5.4e5;
double d = 5.0;
long double ld = 5.0L;
```

https://0.30000000000000004.com

```c
double a = 0.1;
double b = 0.2;
printf("%.17f\n", a + b);
```

Floating point numbers are a leaky abstraction

What Every Programmer Should Know About Floating-Point Arithmetic

Avoiding Overflow, Underflow, and Loss of Precision

What Every Computer Scientist Should Know About Floating-Point Arithmetic

Why IEEE floats have two zeros: +0 and -0

Examples of floating point problems

Оптимизация математических вычислений и опция -ffast-math в GCC 11

The GNU Multiple Precision Arithmetic Library:

```
#include "gmp.h"

void fact(int n)
{
    int i;
    mpz_t p;

    mpz_init_set_ui(p, 1);  /* p = 1 */
    for(i = 1; i <= n; i++)
    {
        mpz_mul_ui(p, p, i);  /* p = p * i */
    }
    printf ("%d!  =  ", n);
    mpz_out_str(stdout, 10, p);
    mpz_clear(p);
}
```

Tutorial on GMP

Структуры

Набор переменных, обычно разного типа, объединённых одним именем.

Struct declaration

```
struct student
{
    int number;
    char code[4];
    double score;
};

my_struct str = {0};
//my_struct str = {2021, "GR_C", 4.5};
//my_struct str = { .number = 2021, code = "GR_C"};
str.number = 42;
printf("%f\n", str.score);

my_struct* str2 = malloc(sizeof(my_struct));
str->number = 1;
```

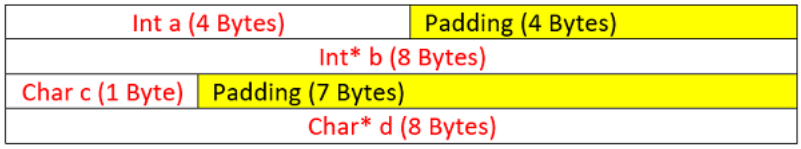
Передача в качестве аргумента функции:

```
int func(struct student s);

func(str);

int func2(struct student* s);

func2(&str);
```

Выравнивание структур The Lost Art of Structure Packing

Расставим точки над структурами C/C++

```
struct A {
    int   a;
    int*  b;
    char  c;
    char* d;
};
```




```
// libmodplug/sndfile.h

#pragma pack(1)

typedef struct WAVEEXTRAHEADER
{
    DWORD xtra_id;
    DWORD xtra_len;
    DWORD dwFlags;
    WORD  wPan;
    WORD  wVolume;
    WORD  wGlobalVol;
    WORD  wReserved;
    BYTE nVibType;
    BYTE nVibSweep;
    BYTE nVibDepth;
    BYTE nVibRate;
} WAVEEXTRAHEADER;

#pragma pack()
```

Объединения

Позволяют хранить переменные разного типа в одном и том же месте памяти, но не одновременно.

Union declaration

```
union ints
{
    uint32_t u32;
    uint16_t u16[2];
    uint8_t  u8;
}

ints u = {0x12345678};
// ints u = { .u8 = 42 };
u.u16[0] = 0x0011;
printf("s.u32 is now %x\n", s.u32);
printf("%d", (uint8_t*)&s == &s.u8);
```

ip_icmp.h

SDL_Event:

```
typedef union SDL_Event
{
    Uint32 type;              /**< Event type, shared with all events */
    SDL_CommonEvent common;   /**< Common event data */
    SDL_DisplayEvent display; /**< Display event data */
    SDL_WindowEvent window;   /**< Window event data */
    SDL_KeyboardEvent key;    /**< Keyboard event data */

    /* ... */
} SDL_Event;
```

Перечисления

Набор символьных имён, соответствующих целочисленным константам.

Enumerations

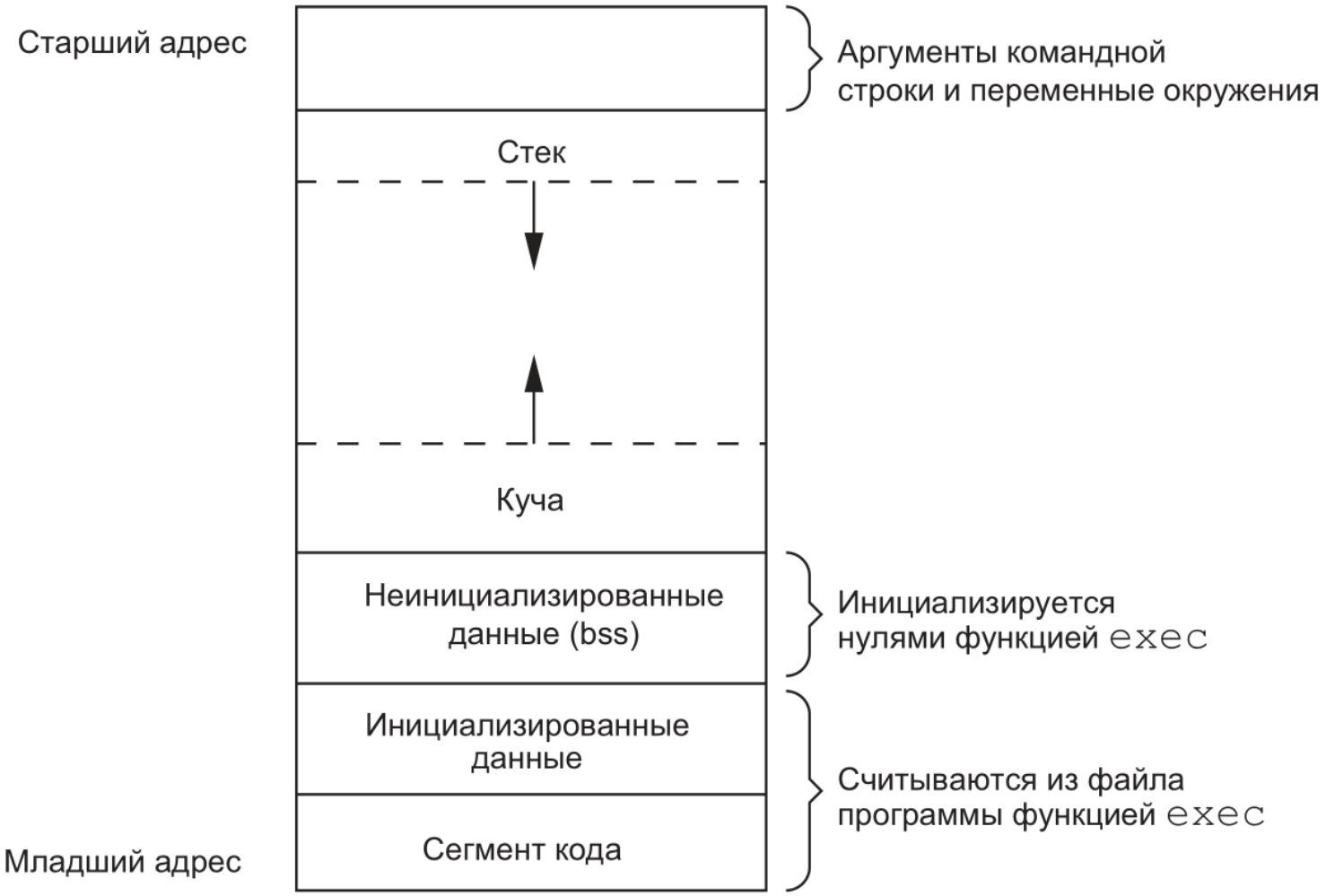
```
enum color { RED, GREEN, BLUE };
enum color r = GREEN;

enum foo
{
    A = 1,
    B,
    C,
    D = 10,
    E,
    F = E + B
}
```

Псевдонимы типов

Typedef declaration

```
typedef int int_t;

int_t i = 42;

typedef char* string;
string s1, s2;
```

Указатели



Pointer declaration

```c
int n = 0;
int* pc = &n;

*pc = 2;

float *p, **pp, f;

p++;

void* pv = NULL;
char* s;

s++;
```

Pointer Arithmetic: GCC

Функции работы с динамической памятью malloc

```c
#include <stdlib.h>

void* malloc(size_t size);
```

```c
int *p = malloc(5 * sizeof(int));
```

calloc

```c
#include <stdlib.h>

void* calloc(size_t num, size_t size);
```

```c
int *p = calloc(5, sizeof(int));
```

realloc

```c
#include <stdlib.h>

void* realloc(void* ptr, size_t new_size);
```

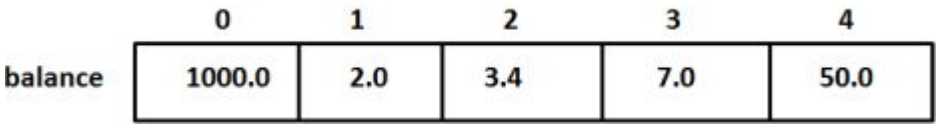
free

```c
#include <stdlib.h>

void free(void* ptr);
```

Массивы

Раздел 6.2.5.20 Стандарта:

> An array type describes a contiguously allocated nonempty set of objects with a particular member object type, called the element type. The element type shall be complete whenever the array type is specified. Array types are characterized by their element type and by the number of elements in the array.



Инициализация C99+

```c
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

Синтаксический сахар доступа к массиву Раздел 6.5.2.1 Стандарта:

> E1[E2] is identical to (*((E1)+(E2)))

```
int array[3];

array[2];
2[array];
*(&array[0] + 2);
```

Многомерные массивы

```
int a[3][4] = {
   {0, 1, 2, 3} ,
   {4, 5, 6, 7} ,
   {8, 9, 10, 11}
};
```

Row-major order:

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

sizeof `sizeof` работает на массивах, но не работает на указателях (точнее, всегда возвращает одно и то же для любого указателя).

Количество элементов в массиве: `sizeof array / sizeof array[0]`

Неопределенная длина Раздел 6.2.5.22 Стандарта:

> An array type of unknown size is an incomplete type. It is completed, for an identifier of that type, by specifying the size in a later declaration.

```
double average(int arr[], int size)
{
   double sum = 0;
   for(int i = 0; i < size; ++i)
      sum += arr[i];
   return sum / size;
}

int main()
{
   int balance[5] = {1000, 2, 3, 17, 50};
   printf("Average value is: %f", average(balance, 5));
}
```

Массивы переменной длины (VLA) C99+

```
float read_and_process(int n)
{
    float vals[n];

    for (int i = 0; i < n; ++i)
        vals[i] = read_val();

    return process(n, vals);
}
```

Flexible array member CPython/Include/stringobject.h

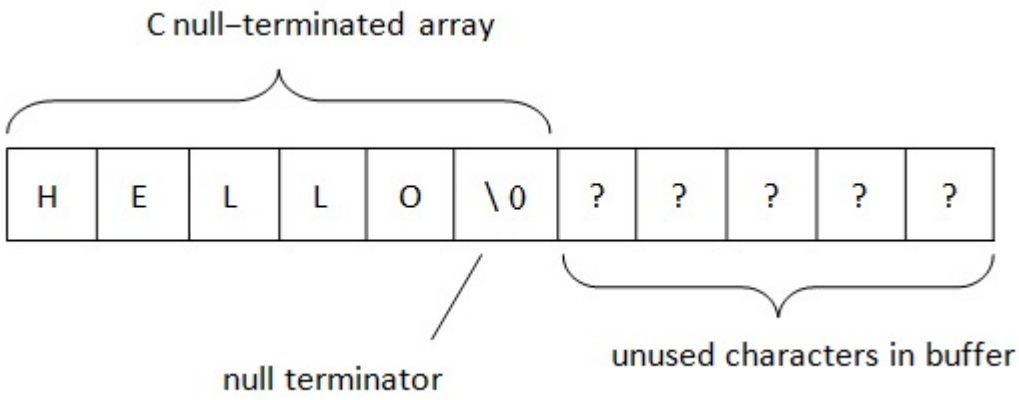
```
typedef struct {
    PyObject_VAR_HEAD
    long ob_shash;
    int ob_sstate;
    char ob_sval[1];

    /* Invariants:
     *     ob_sval contains space for 'ob_size+1' elements.
     *     ob_sval[ob_size] == 0.
     *     ob_shash is the hash of the string or -1 if not computed yet.
     *     ob_sstate != 0 iff the string object is in stringobject.c's
     *       'interned' dictionary; in this case the two references
     *       from 'interned' to this object are *not counted* in ob_refcnt.
     */
} PyStringObject;
```

Строки

Строка в C - нуль-терминированный массив символов, `char` либо `wchar_t` (т.н. multibyte character sequence, MBCS).

```
char* str = "hello";
```

```
#include <wchar.h>
wchar_t* wstr = L"hello";
```



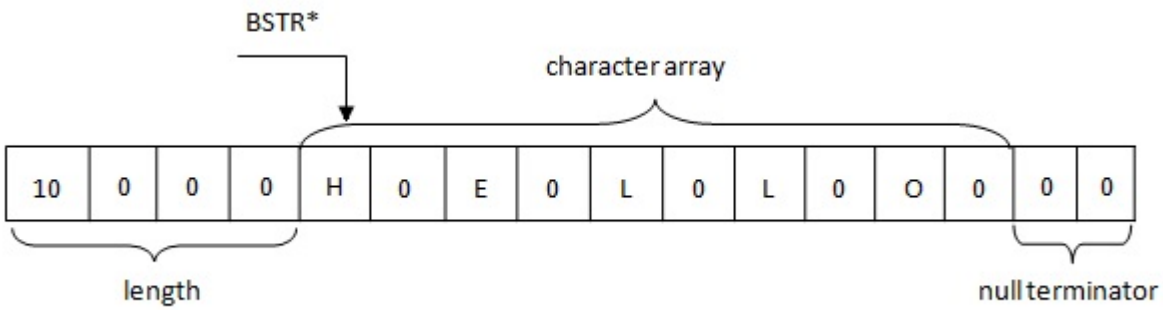
Алгоритм маляра Шлемиля (Спольски Дж. Джоэл о программировании).

Poul-Henning Kamp, The Most Expensive One-byte Mistake

Список строковых функций: https://en.wikipedia.org/wiki/C_string_handling

WinAPI: A/W варианты (например, `CreateFileA` vs `CreateFileW`), тип `TCHAR` и макросы `UNICODE`, `_T/TEXT`. https://habr.com/ru/post/164193

COM-строки (BSTR)

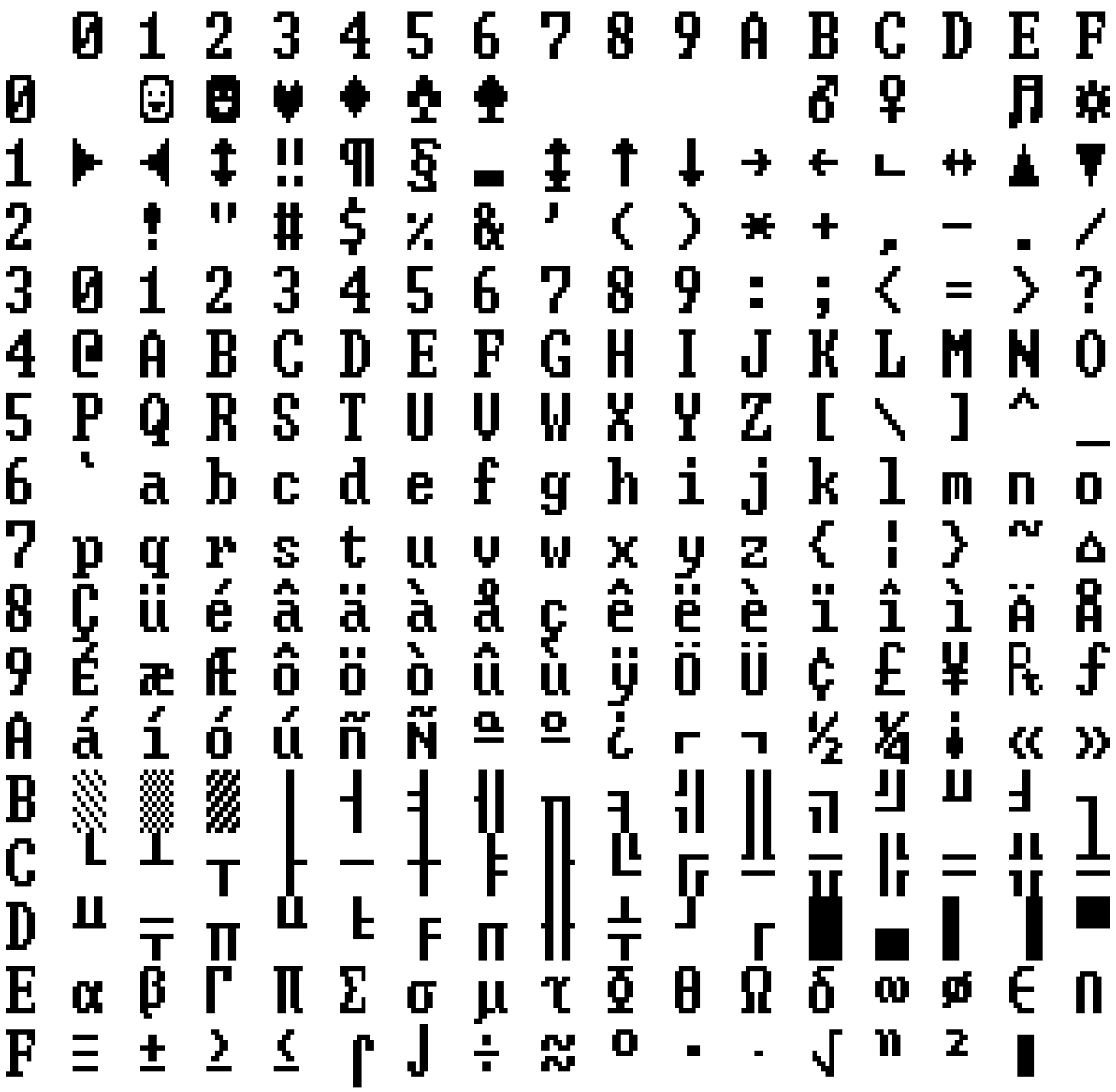


Unicode и кодировки Абсолютный минимум, который каждый разработчик должен знать о Unicode и кодировках, оригинал

Абсолютный минимум об Unicode на 2023 год (всё ещё — никаких оправданий!), оригинал

https://asciitable.com

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SPC	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL



- **МИФ**: Unicode - это просто 16-ти битный код, где каждый символ занимает 16 бит и содержит 65536 возможных символов.
- Code points. Cyrillic Capital Letter A - U+0410.
- Кодировки: UTF-8, UTF-16 (LE/BE), UTF-7, UCS-4 и т.д.
- “🤦”.length == 7
- Нет смысла держать строку, не зная в какой она кодировке.
- Unicode sandwitch

```
$ echo "абвгдеёжзийклмнопрстуфхцчшщыъьэюя" | hexdump -C
00000000  d0 b0 d0 b1 d0 b2 d0 b3  d0 b4 d0 b5 d1 91 d0 b6  |................|
00000010  d0 b7 d0 b8 d0 b9 d0 ba  d0 bb d0 bc d0 bd d0 be  |................|
00000020  d0 bf d1 80 d1 81 d1 82  d1 83 d1 84 d1 85 d1 86  |................|
00000030  d1 87 d1 88 d1 89 d1 8b  d1 8a d1 8c d1 8d d1 8e  |................|
00000040  d1 8f 0a                                          |...|
00000043
```

Библиотеки для обработки Unicode

- GLib
- libicu
- utf8.h (header-only)
- GNU libunistring

Конвертация между различными кодировками

```c
#include <iconv.h>

int main()
{
    const char* in = "Вопрос!";
    char out[1024];

    /* ! */
    char* in_ptr = in;
    char* out_ptr = &out[0];

    size_t inbytes = strlen(in);
    size_t outbytes = sizeof out / sizeof out[0];

    iconv_t cd = iconv_open("koi8-r", "utf-8");
    iconv(cd, &in_ptr, &inbytes, &out_ptr, &outbytes);
    iconv_close(cd);

    printf("%s\n", out);
}
```

7 ложных предположений о том, как устроены строки

Символы Unicode: о чём должен знать каждый разработчик

UTF-8 Everywhere Manifesto

How does UTF-8 turn “😂” into “F09F9882”?

Кодировка данных: кириллица

Plain Text - Dylan Beattie - NDC Copenhagen 2022

Unicode в C // Демо-занятие курса «Программист C»

Длинные строковые константы

```c
static const char* str = "my very very "
    "very very very "
    "long string constant";
```

Функции

С не является функциональным ЯП (но см. ОУ Функциональное программирование на С)

C gibberish <-> english

Функции не могут возвращать массивы (раздел 6.9.1.3 Стандарта).

Соглашения вызова

Название	Аргументы	Управление стеком	Использование
cdecl	Справа налево	Вызывающая функция	По умолчанию в C и C++
stdcall	Справа налево	Вызываемая функциия	Системные функции WinAPI
fastcall	Первые два через регистры RCX и RDX, остальные справа налево	Вызывающая функция	По умолчанию в компиляторах Borland
thiscall	`this` в RCX, остальные справа налево	Вызывающая функция	По умолчанию для методов классов в C++

Функции с переменным числом аргументов

```c
#include <stdarg.h>

double average(int count, ...)
{
    va_list ap;
    int j;
    double sum = 0;

    va_start(ap, count);
    for (j = 0; j < count; j++)
    {
        sum += va_arg(ap, double);
    }
    va_end(ap);

    return sum / count;
}

int main()
{
    printf("%f; %f\n",
        average(2, 1.0, 2.0),
        average(3, 1.0, 2.0, 3.0));

}
```

Файлы

`FILE*` - т.н. *opaque pointer*, непрозрачный указатель (тж. “чеширский кот”).

```c
#include <stdio.h>

FILE* stdin;
FILE* stdout;
FILE* stderr;

FILE* fopen(const char* filename, const char* mode);
// mode: "r", "w", "a"; modifiers: "+", "b", "x"
int fclose(FILE* fp);
```

```c
int fflush(FILE* fp);

int fgetc(FILE* fp);
char* fgets(char* str, int num, FILE* fp);
size_t fread(void* ptr, size_t size, size_t count, FILE* fp);
int fscanf(FILE* fp, const char* format, ...);


int fputc(int character, FILE* fp);
int fputs(const char* str, FILE* fp);
size_t fwrite(const void* ptr, size_t size, size_t count, FILE* fp);
int fprintf(FILE* fp, const char* format, ...);

int fseek(FILE* fp, long int offset, int origin);
// origin: SEEK_SET, SEEK_CUR, SEEK_END
long int ftell(FILE* fp);
```

```c
int ch;
while((ch = fgetc(fp)) != EOF)
{
    /* ... */
}
```

_FILE_OFFSET_BITS