HIGHER SCHOOL OF ECONOMICS
NATIONAL RESEARCH UNIVERSITY

Introduction to Programming

Streams and Strings

Sergey Shershakov

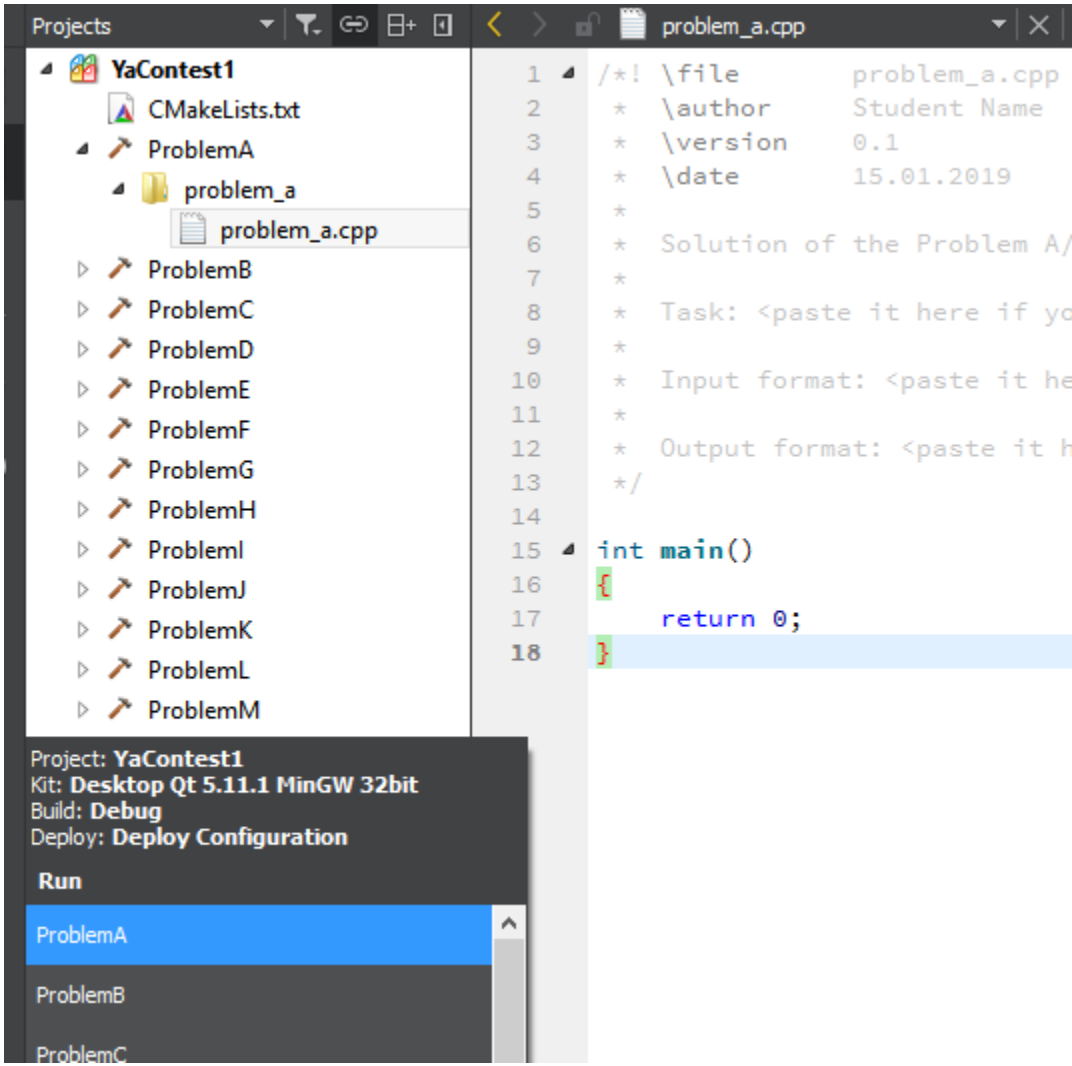
#3/17 Jan 2019

Test 2 (5 pts)

https://goo.gl/forms/7PqlqR3zSZFgWxMs1

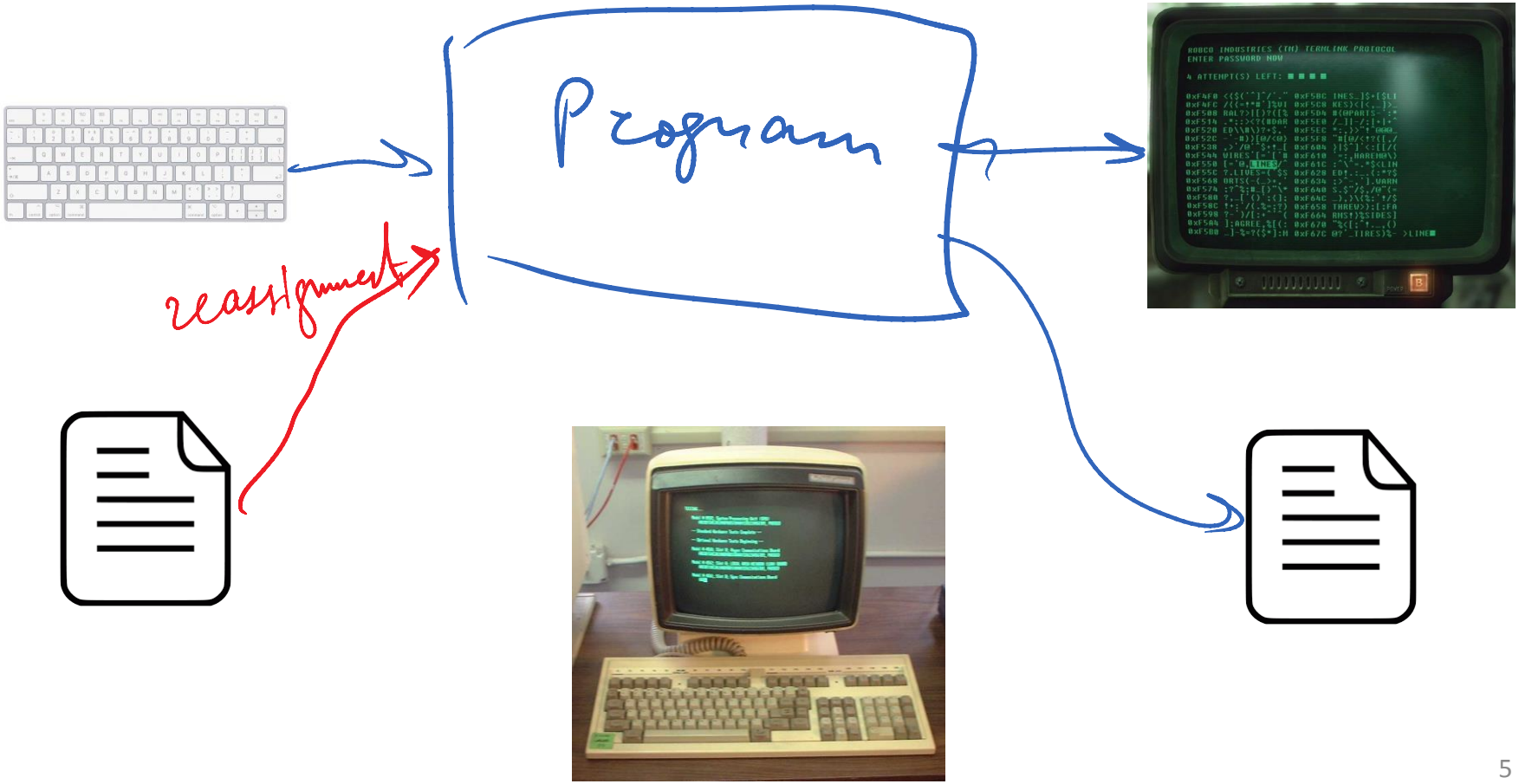
Recommended Contest Project Structure

```
~/se/progr/
    ├─ /lectures/
    ├─ /workshops/
    └─ /contests/
        ├─ /ya_contest_1/
        │   ├─ CMakeLists.txt
        │   ├─ /problem_a/
        │   │   └─ problem_a.cpp
        │   └─ /problem_b/
        │       └─ problem_b.cpp
        │       ...
        │
        └─ /ya_contest_2/
                ...
```



INTRODUCTION TO I/O AND STREAMS

IO Model of a Program



Streams

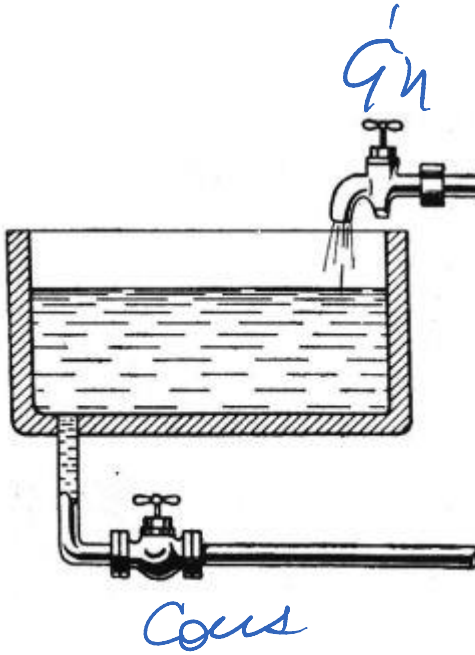
- A *stream* is an abstraction for input/output.
- Standard input and standard output are represented by global objects `std::cin` (of a type `std::istream`) and `std::cout` (of a type `std::ostream`) respectively.
- One can think of a stream as a source (input) or a destination (output) of characters of an indefinite length.

- Input data to a program from a stream:

```
int x;
double y;
std::cin >> x >> y;
```

- Output data from a program to a stream:

```
std::string userName = "Sergey";
std::cout << "Hello, " << userName;
```



https://en.cppreference.com/w/cpp/io

STRINGS AND ENCODING SCHEMES

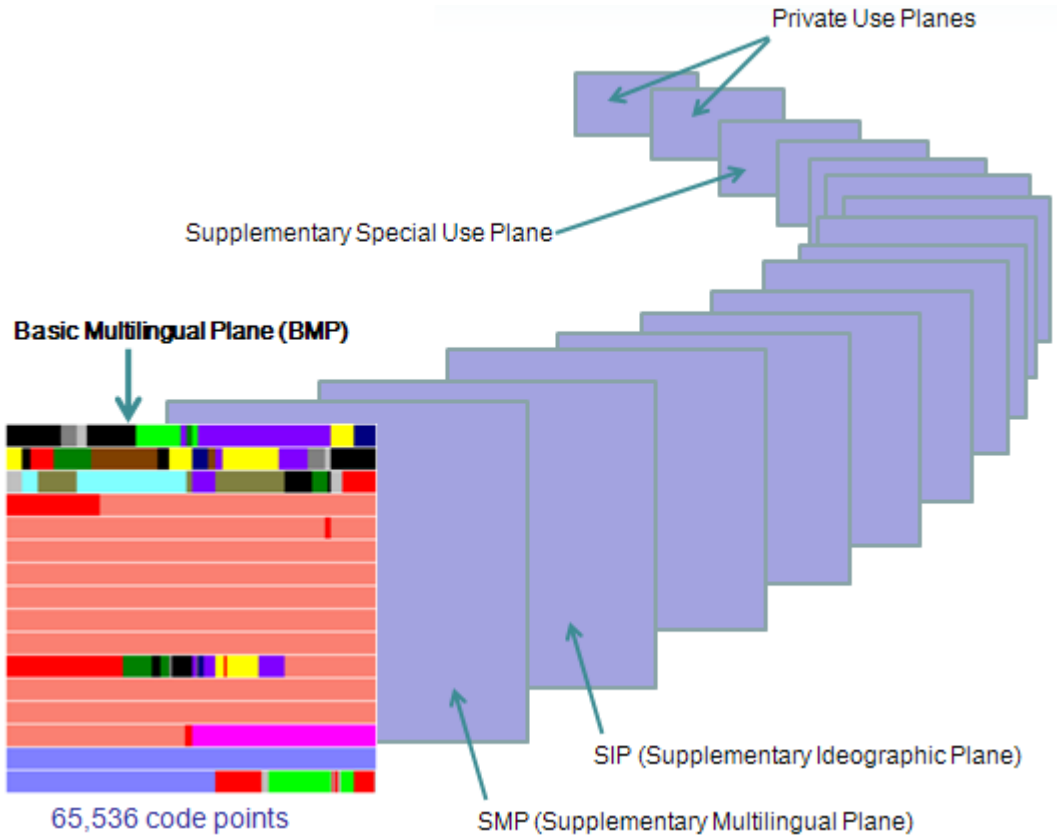
Representation of Characters

- We have a set of characters we want to use in a computer:
  - A, B, C, D, …, Z
  - a, b, c, … z
  - 0, 1, 2, … , 9
  - !, @, #, $, %, ^, &, *, (,), -, _, =, +, , ; , …
  - let's enumerate them!

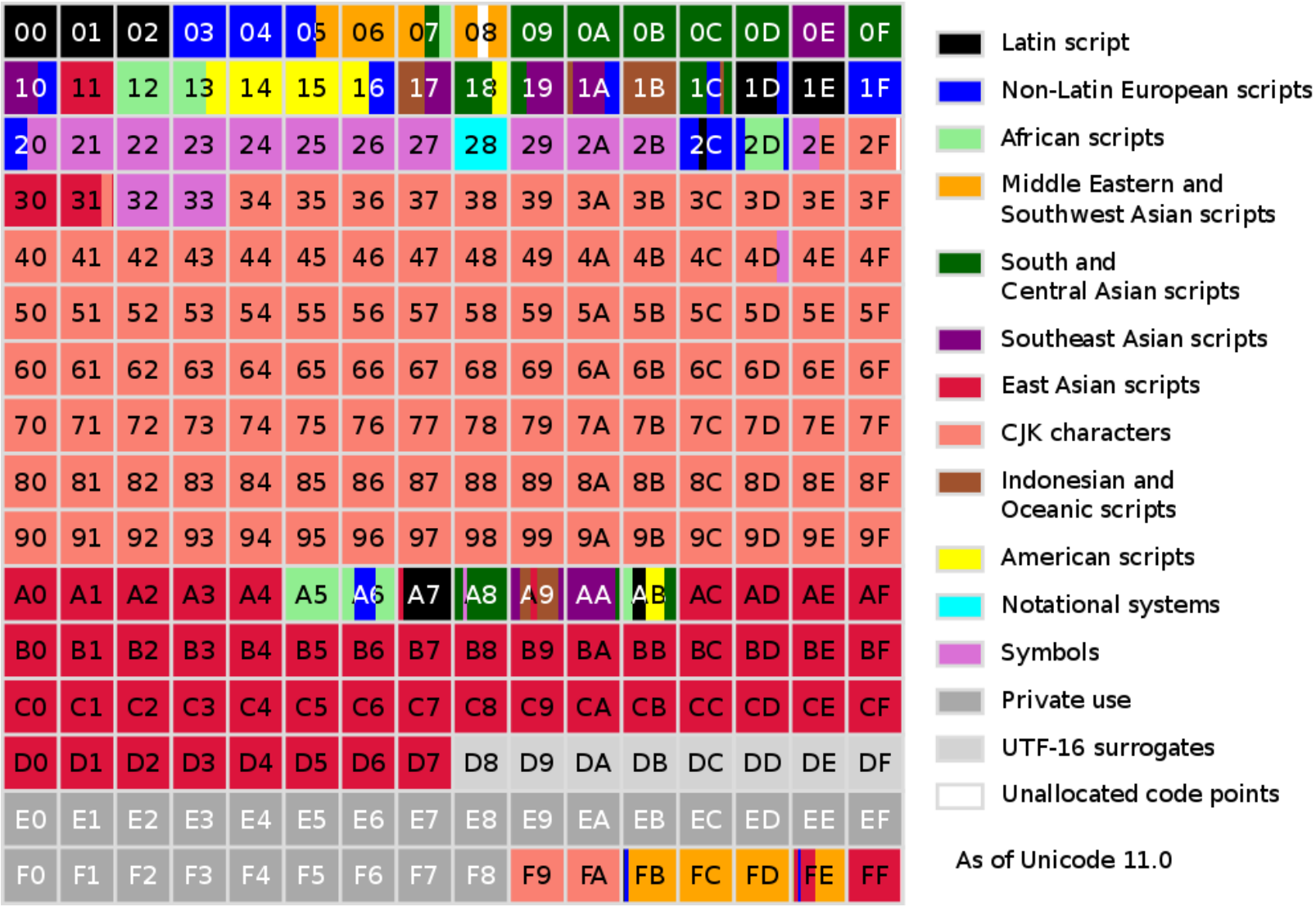
- What about national alphabets?
  - А, Б, В, … Я, а, б, в, … я
  - Α , Β , Γ , Δ , Ε , …, Ω , α , β , γ , … ξ , ω
  - …

Unicode Standard

- Codes almost all known symbols using quite big numbers (millions)
- Structured by using planes and blocks
- Can be implemented by different character encodings



Basic Multilingual Plane



8-bit code pages

- Each character is coded by a number from 0..255
- Too small to fit even the most basic characters
- Each alphabet has its own coding standard:
  - ASCII — basic set of 127 characters inc. Latin
  - CP866 — obsolete “DOS” Cyrillic codepage
  - CP1251 (ANSI) — Windows-based Cyrillic codepage
  - KOI8-R — obsolete Linux-based Cyrillic codepage
  - …

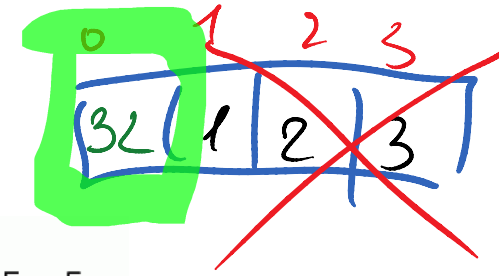
ASCII Table

ASCII Code Chart

	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		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
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
8	80	81	82													

- uses only 128 positions (… bits?)
- contains Latin symbols, numbers, basic symbols
- see Ya.Contest 1 Problems M and N

CP1251 Codepage



Codepage 1251 - Cyrillic Windows

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-		0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F
1-	0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
2-	0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3-	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4-	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5-	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6-	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7-	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	007F
8-	Ђ 0402	Ѓ 0403	‚ 201A	ѓ 0453	„ 201E	… 2026	† 2020	‡ 2021	€ 20AC	‰ 2030	Љ 0409	‹ 2039	Њ 040A	Ќ 040C	Ћ 040B	Џ 040F
9-	ђ 0452	‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014	0098	™ 2122	љ 0459	› 203A	њ 045A	ќ 045C	ћ 045B	џ 045F
A-	00A0	Ў 040E	ў 045E	Ј 0408	¤ 00A4	Ґ 0490	¦ 00A6	§ 00A7	Ё 0401	© 00A9	Є 0404	« 00AB	¬ 00AC	- 00AD	® 00AE	Ї 0407
B-	° 00B0	± 00B1	І 0406	і 0456	ґ 0491	µ 00B5	¶ 00B6	· 00B7	ё 0451	№ 2116	є 0454	» 00BB	ј 0458	Ѕ 0405	ѕ 0455	ї 0457
C-	А 0410	Б 0411	В 0412	Г 0413	Д 0414	Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C	Н 041D	О 041E	П 041F
D-	Р 0420	С 0421	Т 0422	У 0423	Ф 0424	Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C	Э 042D	Ю 042E	Я 042F
E-	а 0430	б 0431	в 0432	г 0433	д 0434	е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C	н 043D	о 043E	п 043F
F-	р 0440	с 0441	т 0442	у 0443	ф 0444	х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C	э 044D	ю 044E	я 044F

Demo App: Output Characters of the Current Codepage

C:\programs\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe

```
 (32)  ! (33)  " (34)  # (35)  $ (36)  % (37)  & (38)  ' (39)
( (40)  ) (41)  * (42)  + (43)  , (44)  - (45)  . (46)  / (47)
0 (48)  1 (49)  2 (50)  3 (51)  4 (52)  5 (53)  6 (54)  7 (55)
8 (56)  9 (57)  : (58)  ; (59)  < (60)  = (61)  > (62)  ? (63)
@ (64)  A (65)  B (66)  C (67)  D (68)  E (69)  F (70)  G (71)
H (72)  I (73)  J (74)  K (75)  L (76)  M (77)  N (78)  O (79)
P (80)  Q (81)  R (82)  S (83)  T (84)  U (85)  V (86)  W (87)
X (88)  Y (89)  Z (90)  [ (91)  \ (92)  ] (93)  ^ (94)  _ (95)
` (96)  a (97)  b (98)  c (99)  d (100) e (101) f (102) g (103)
h (104) i (105) j (106) k (107) l (108) m (109) n (110) o (111)
p (112) q (113) r (114) s (115) t (116) u (117) v (118) w (119)
x (120) y (121) z (122) { (123) | (124) } (125) ~ (126) ⌂ (127)
А (128) Б (129) В (130) Г (131) Д (132) Е (133) Ж (134) З (135)
И (136) Й (137) К (138) Л (139) М (140) Н (141) О (142) П (143)
Р (144) С (145) Т (146) У (147) Ф (148) Х (149) Ц (150) Ч (151)
Ш (152) Щ (153) Ъ (154) Ы (155) Ь (156) Э (157) Ю (158) Я (159)
а (160) б (161) в (162) г (163) д (164) е (165) ж (166) з (167)
и (168) й (169) к (170) л (171) м (172) н (173) о (174) п (175)
░ (176) ▒ (177) ▓ (178) │ (179) ┤ (180) ╡ (181) ╢ (182) ╖ (183)
╕ (184) ╣ (185) ║ (186) ╗ (187) ╝ (188) ╜ (189) ╛ (190) ┐ (191)
└ (192) ┴ (193) ┬ (194) ├ (195) ─ (196) ┼ (197) ╞ (198) ╟ (199)
╚ (200) ╔ (201) ╩ (202) ╦ (203) ╠ (204) ═ (205) ╬ (206) ╧ (207)
╨ (208) ╤ (209) ╥ (210) ╙ (211) ╘ (212) ╒ (213) ╓ (214) ╫ (215)
╪ (216) ┘ (217) ┌ (218) █ (219) ▄ (220) ▌ (221) ▐ (222) ▀ (223)
р (224) с (225) т (226) у (227) ф (228) х (229) ц (230) ч (231)
```

```
char ch = (char)i;
std::cout << ch << " (" << i << ")\t";
```

Types for Representing Characters

Type	Size	Range	Codepages (examples)
`char`	8 bit	0..255	Fixed width: ASCII (ext), CP1251, KOI8-r Variable width: UTF-8
`char16_t`	16 bit	0..65535	Fixed width: UCS-2 “Variable width” (surrogate pairs): UTF-16
`char32_t`	32 bit	0.. 4294967295	UTF-32
`wchar_t`	16 or 32 bit		UCS-2

'\' '|' '_' '"' 'f'

- UCS-2 — 2 bytes per symbol, fixed size
- UTF-16 — 2 or 4 bytes (surrogate pairs) per symbol, variable size
- UTF-32 — 4 bytes per symbol, fixed size
- UTF-8 — 1 to 4 bytes per symbol, fixed size

```
char ch = 'A';
```

UTF-8

- 1 to 4 bytes per symbol:
  - 1 byte codes ASCII symbols (0..127)
  - 2 bytes code 1 920 symbols including: Latin-script alphabets, and also Greek, Cyrillic, Coptic, Armenian, Hebrew, Arabic, …, *Combining Diacritical Marks* (é, è, ê, ā, á, ă, ï, ü, ë, й, …)
  - 3 bytes code the rest of the Basic Multilingual Plane including Chinese, Japanese, Korean
  - 4 bytes for other symbols such as math symbols, emoji etc.

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

ex_1.cpp | <Select Symbol>

```cpp
/*! \file        ex_1.cpp
 * \author      Sergey Shershakov
 * \version     0.1
 * \date        15.01.2019
 *
 * Prints printable characters of the current codepage.
 * Печатает печатное.
 */

#include <iostream>

int main()
{
    const short MAX_IN_THE_ROW = 8;
```

BOM

std::endl

InsertNonFull1.xibt | xiexample2.pyd | p1.bin | Startup | ex_1.cpp

Edit As: Hex | Run Script | Run Template

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: EF BB BF 2F 2A 21 20 5C 66 69 6C 65 20 20 20 20  ï»¿/*! \file
0010h: 20 20 20 65 78 5F 31 2E 63 70 70 0D 0A 20 2A 20     ex_1.cpp.. *
0020h: 20 5C 61 75 74 68 6F 72 20 20 20 20 20 53 65 72   \author     Ser
0030h: 67 65 79 20 53 68 65 72 73 68 61 6B 6F 76 0D 0A  gey Shershakov..
0040h: 20 2A 20 20 5C 76 65 72 73 69 6F 6E 20 20 20 20   *  \version
0050h: 30 2E 31 0D 0A 20 2A 20 20 5C 64 61 74 65 20 20  0.1.. *  \date
0060h: 20 20 20 20 20 31 35 2E 30 31 2E 32 30 31 39 0D       15.01.2019.
0070h: 0A 20 2A 0D 0A 20 2A 20 20 50 72 69 6E 74 73 20  . *.. *  Prints
0080h: 70 72 69 6E 74 61 62 6C 65 20 63 68 61 72 61 63  printable charac
0090h: 74 65 72 73 20 6F 66 20 74 68 65 20 63 75 72 72  ters of the curr
00A0h: 65 6E 74 20 63 6F 64 65 70 61 67 65 2E 0D 0A 20  ent codepage...
00B0h: 2A 20 20 D0 9F D0 B5 D1 87 D0 B0 D1 82 D0 B0 D0  *  ÐŸÐµÑ‡Ð°Ñ‚Ð
00C0h: B5 D1 82 20 D0 BF D0 B5 D1 87 D0 B0 D1 82 D0 BD  µÑ‚ Ð¿ÐµÑ‡Ð°Ñ‚Ð½
00D0h: D0 BE D0 B5 2E 0D 0A 20 2A 2F 0D 0A 0D 0A 23 69  Ð¾Ðµ.... */....#i
00E0h: 6E 63 6C 75 64 65 20 3C 69 6F 73 74 72 65 61 6D  nclude <iostream
00F0h: 3E 0D 0A 0D 0A 69 6E 74 20 6D 61 69 6E 28 29 0D  >....int main().
0100h: 0A 7B 0D 0A 20 20 20 20 63 6F 6E 73 74 20 73 68  .{..    const sh
0110h: 6F 72 74 20 4D 41 58 5F 49 4E 5F 54 48 45 5F 52  ort MAX_IN_THE_R
0120h: 4F 57 20 3D 20 38 3B 0D 0A 0D 0A 0D 0A 20 20 20  OW = 8;......
```

Introducing Arrays

- *Array* (plain C-style) is a set of objects of the same type placed in adjacent memory cells.

- `int nameOfArray[5];`

- `int nameOfArray[5] = {9, 10, 30, 40, 42};`

Array Initialization

- `int x[4] = {3, 6, 8, 10};  // okay`
- `int y[4];  // okay`
- `y[4] = {5, 6, 7, 9};  // not allowed`
- `y = cards;  // not allowed`

x[0]

- `double points[5] = {3.0, 1.2}; // fewer is ok`
- `short stars[] = {1, 5, 3, 8};  // is ok, compiler counts`
  `// the number of elements for you`

- How to find out the number of items?

```
short things[] = {1, 7, 2, 10};
int elNum = sizeof(things) / sizeof(short);
```

8 2 = 4

Is a String an Array of Symbols?

- `char a[8] = {'b', 'e', 'a', 'u', 'x', ' ', 'I', 'I'};  // not a string!`
- `char s[8] = {'f', 'a', 't', 'e', 's', 's', 'a', '\0'}; // a string!`

```
char str1[8] = "Star";
```

S	t	a	r	\0	X	3	

Class `std::string`

- `std::string` → `std::basic_string<char>`

8-bit

- represents strings with 8-bit characters:
  - fixed-width: *ASCII*, *CP1251*, *CP866*,…
  - variable-width: *UTF-8*

\0

```
std::string s = "I'm a string!";
```

- has an internal buffer to store a continuous sequence of characters
  - `s.c_str()` returns a pointer to the buffer
- has a number of useful methods and operations:
  - `s.size()`, `s.length()` for a string's length;
  - `s.clear()` clears a string;
  - `s.empty()` test if a string is empty;
  - `s[10]` gives an individual (11th) symbol in the sequence;
  - `s + "Abs"` — concatenation, gives a new string.

Reading String Data from a Stream

- Reading strings word by word:

```
std::string s1, s2;
std::cin >> s1 >> s2;       // "Abc Xyz"
```

- Reading a whole line containing spaces (see Ya.Contest 1 Problems E):

```
std::string bigStr;
std::getline(cin, bigStr); // "Hello World!"
```

- Reading a text character by character (see Ya.Contest 1 Problems H):

```
char ch;
std::cin >> std::noskipws;
std::cin >> ch;
std::cin >> ch;
…
```

or

```
char ch2 = std::cin.get();
std::cin.get(ch2)
```