PSL

ECUE apprentissage de la programmation - the c++ language

Valérie Roy et Basile Marchand
MINES ParisTech

Plan

1. Introduction
   - What is the c-language ?
   - What is the c++-language ?
   - character set of c++ programs

Plan

1 Introduction
What is the c-language ?
What is the c++-language ?
character set of c++ programs

the c language

the C-language

- designed by *Dennis Ritchie* at Bell Labs in 1970s for the UNIX operating system
- first c tutorial co-written by Brian Kernighan in 1978 [1]
- standardization committee initiated in 1983
- the standard ANSI-C completed in 1989 (C89)
- the c-language is still alive
- last ANSI-C standard in 2018 (a simple *bugfix* release)
- the c language is involved (syntax, implementation, libraries, etc.) in *a lot of* other languages (c++, JavaScript, c#, java, Go, Python, Rust, etc.)

in this course

- we will not learn the c language itself (it is outside the scope of this course)
- we will focus on the c basics appearing in the c++ language

1. the historic piece can be found here https ://archive.org/details/TheCProgrammingLanguageFirstEdition

Plan

1 Introduction
- What is the c-language ?
- What is the c++-language ?
- character set of c++ programs

What is the c++ language ?

c++ is a **general-purpose strongly-typed** programming language

c++ is a **high-level** programming language (object-oriented programming, templates, ...)

c++ based on the **c** programming language

the **c** programming language is described in `ISO`[a]`/IEC`[b] `9899:1999 Programming languages`

a. International Organization for Standardization
b. International Electrotechnical Commission

the **c standard library** is a subset of the **c++ standard library**

```c
/* in c programs */
#include <math.h>
```

```c
// in c++ programs
#include <cmath>
```

International Standards of the c++ language

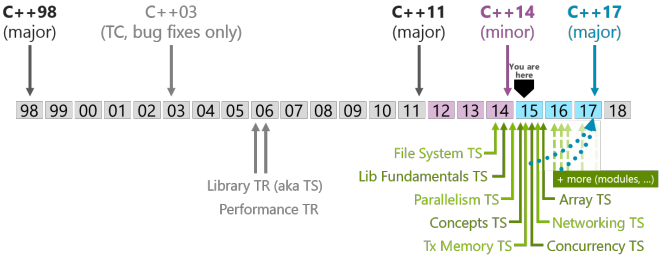

Figure 1 – c++ milestones

there have been a *several* c++ programming language standards by ANSI [a]

name	date	standard
c++98	1998	ISO/IEC 14882 :1998
c++03	2003	ISO/IEC 14882 :2003
c++TR1	2007	ISO/IEC TR 19768 :2007
c++11	2011	ISO/IEC 14882 :2011
c++14	2014	ISO/IEC 14882 :2014(E)
c++17	2017	

a. American National Standards Institute

c++ standardization

c++11 refers to the **2011 version** of the c++ programming language

most of the compilers do not implement by default the new c++11 standard

to **compile** your source files using the c++11 **standard** : you (may) have to **pass** an **option** to the **compiler**.

With the g++ compiler on linux :

```
g++ -std=c++11 FILE.cpp
```

in conclusion, this course will focus on c++11 : when a feature only exists in c++11 a will (try to) warn you

Modern c++ language Stroustrup's Advice

c++ is particularly suited for **resource-constrained** applications

it is not an easy language to lear : programmers **must takes the time** to master the language

the c++ language has **dramatically improved** over the years

c++ **evolves** regularly modern c++ versions (such as c++11) are **far better** for writing quality software than previous versions

better programming styles and techniques, more elegant, correct, maintainable, and more efficient code

An emphasis on stability

because **billions** of lines of c++ are deployed world-wide : c++ puts emphasis on **stability**

thus **standards-conforming** code you write today will still work a couple of decades from now

however if you stick to **older styles**, you will be writing **lower-quality** and **worse-performing** code

you will **do better** writing software with **modern c++**

Plan

1 Introduction
What is the c-language ?
What is the c++-language ?
character set of c++ programs

character set of c++ programs

Basic source character set of c++ programs

programs are written in **text files**

inside a **text file** a **code** is **associated** to **characters**

the **basic set** of **characters** of a **c++ program** consists of **96 characters** :

- the **space** character
- several control characters[a] (horizontal and vertical tab, backspace, delete, new-line, ...)
- the following 91 **graphical** characters :

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
_ { } [ ] # ( ) < > \% : ; . ? * + - / ^ & | ~ ! = , \ " '
```

a. non-printing

this **set** of characters is encoded by the **US-ASCII**[a] character-encoding scheme

a. American Standard Code for Information Interchange

the **US-ASCII** (American Standard Code for Information Interchange [2]) aka **UTF8-1**

this code **pairs** each **character** with a **code**

it is based on the **English alphabet**

it is a **7-bits code** points

it comprises **128** ($2^7 = 128$) **code points** [a]

a. in hexa : **range 0** to **7F**

2. see http ://www.iana.org/assignments/character-sets/character-sets.xhtml

USASCII code chart

Bits b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 ↓ Row ↓ Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	¤	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

Figure 2 – The 7-bits $b_7 \ldots b_1$ ASCII character set encoding from 0 to 127 (7F in hexa, DEL)

exemples

for example :

char	dec	hex	oct	bin
!	33	21	41	(0)10 0001
"	34	22	42	10 0010
$	36	24	44	10 0100
'	39	27	47	10 0111
)	41	29	51	10 1001
*	42	2A	52	10 1010
/	47	2F	57	10 1111
0	48	30	60	11 0000
4	52	34	64	11 0100
9	57	39	71	11 1001
:	58	3A	72	11 1010
;	59	3B	73	11 1011
=	61	3D	75	11 1101
A	65	41	101	100 0001
B	66	42	102	1000010

every c++ **implementation** must support :

- characters from the basic source character set (ASCII)
- a way to name other characters (the universal character \uffff or \Uffffffff)

BUT the mapping from characters (in your file), to source characters (used at compile time) is **implementation defined**

```
#include <iostream>
int main() {
   std::cout << "\u00A4" << '␣' << "\u00A6" << '␣'
             << "\u00A8" << '␣' << "\u00B4" << '␣'
             << "\u00B8" << '␣' << "\u00BC" << '␣'
             << "\u00BD" << '␣' << "\u00BE" << '␣'
             << std::endl;
}
```

```
$$ g++ file.cpp
$$ ./a.out
???
```

be careful when you use, inside your programs, **characters** that are **not** ASCII

c++ lexical units

c++ lexical units are : header (after a `#include` directive), **identifiers** (including the **keywords**), **numbers**, **character**, **string** literals, **operators**, **punctuators**, ...

Keywords

alignas	continue	friend	register	true
alignof	decltype	goto	reinterpret_cast	try
asm	default	if	return	typedef
auto	delete	inline	short	typeid
bool	do	int	signed	typename
break	double	long	sizeof	union
case	dynamic_cast	mutable	static	unsigned
catch	else	namespace	static_assert	using
char	enum	new	static_cast	virtual
char16_t	explicit	noexcept	struct	void
char32_t	export	nullptr	switch	volatile
class	extern	operator	template	wchar_t
const	false	private	this	while
constexpr	float	protected	thread_local	
const_cast	for	public	throw	

a program obeys to syntaxic rules (the c++ grammar)