The component base of C language

Nguyễn Dũng
Faculty of IT – Hue College of Science

Content

- A brief history of C
- Standard of C
- Characteristics of C
- The C compilation model
- Character set and keyword
- Data types
- Comment, variable, constant
- Structure of program
- Expression and operators

A brief history of C

- The C programming language is designed and developed by **Dennis Ritchie** at Bell Laboratories in the early 1970s.
- Evolved from **B** (Ken Thompson, 1970), which evolved from **BCPL** (Martin Richard, 1967).
- Designed for systems programming
  - `Operating system`
  - `Compiler`
  - `Utility program`
  - …
- It is also widely used for developing application software

Standard of C

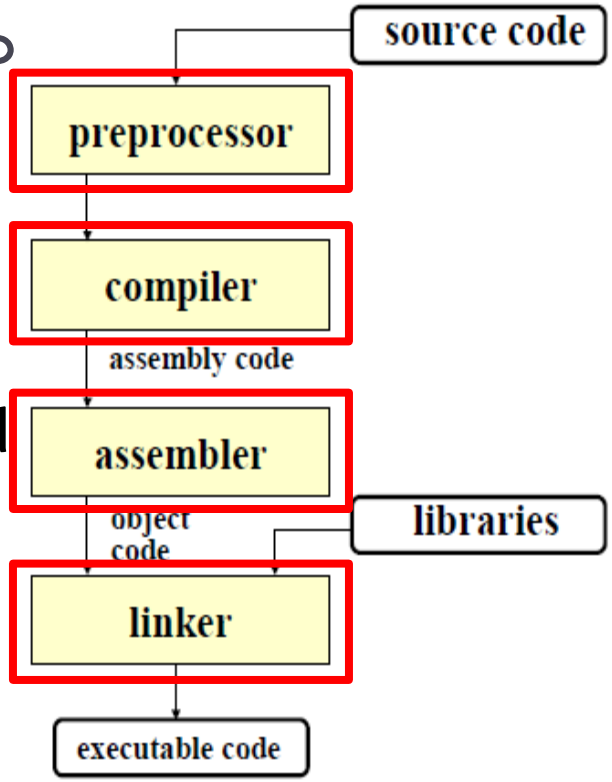
- Be Standardized in 1989 by **ANSI** (**American National Standards Institute**) known as **ANSI C**
- In 1990, the ANSI C standard was adopted by the **International Organization for Standardization** (**ISO**) is known as **C89**
- As part of the normal evolution process the standard was updated in 1995 (**C95**) and 1999 (**C99**),...

Characteristics of C

- Program written in C are very **efficent, fast** and **small in size**.
- **Structered language.** Extensive use of functions
- **C is a powerful and flexible language**. It can be used for projects as Operating System, Word Processors, Graphics,... and even compilers for another language.
- **C is highly portable language**. This means that a C program written for one computer system can be run on another computer system with a little or no modification.
- ...

The C compilation model

- The preprocessor accepts source code as input and
  - removes comments
  - extends the code according to the preprocessor directives included in the source code
- The compiler takes output of preprocessor and produces assembly code
- The assembler takes the assembly code and produces machine code (or object code)
- The linker takes the object code, joins it with other object code and libraries and produces code that can be excutable.



Character set and keywords

- Character set of C language includes the following characters:
  - **Letters: a - z, A - Z, _**
  - **Digits: 0 - 9**
  - **Punctuation: ~ ! @ # % ^ & * ( ) - + = : ; " ' < > , . ? | / \ { } [ ]**
- Keywords:

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Rules of Name

- Name in the C language is a set of characters which:
  - can contain numbers, letters and underscore
  - must be started with letters or underscore
  - can not contain special characters or space
  - isn't similar keyword
  - is case sensitive
- Example:
  - "Percent, y2x5__fg7h, _1990_tax, A" are right
  - "ngay-sinh, double, 9winter" are wrong

Comments

- There are two types of comment:
  - **comment on single line:**
    - **Syntax: //**
    - **Example: *//this is a comment line***
  - **comment on multi line**
    - **Syntax: /* */**
    - **Example:**

```
/*
    This is first comment line
    …
    This is n(th) comment line
*/
```

Structure of a C program

- A C program contains the following elements:
  - Proprocessor command
  - Type definitions
  - Function prototypes
  - Variables and constants
  - Functions
- All programs must contain a single “main” function. All functions, include “main” function, have the following formating:

```
type_return function_name(parameters)
    {
        //Body function
    }
```

Structure of a C program (cont)

- Example

```
#include <stdio.h>
#define TIMES 10
double myfunction(float);
void main() {
  double wert;
  double pi = 3.14;
  printf("Multiply by 10\n");
  wert = myfunction(pi);
  printf("%d * %f = %f\n",TIMES, pi, wert);
}
double myfunction(double zahl){
  int i;
  double count = 0;
  count = TIMES * zahl;
  return count;
}
```

Data types

- C has the following basic

Type	Size (byt	
unsigned char	1	
[signed] char	1	-128
usigned int	2	5535
[signed] int	2	-32768→32767
unsigned short int	2	0→65535
[signed] short int	2	-32768→32767
unsigned long	4	0→4.294.967.295
[signed long]	4	-2.147.483.648→2.147.483.647
float	4	3.4E-38→3.4E+38
double	8	1.7E-308→1.7E+308
long double	10	3.4E-493→1.1E+4932

The sizes of the data types are not standardized (depend on the implementation system and compiler)

Constant

- A constant specifies a value that cannot be modified by a program
- Special constants for use with strings
  - `\n` newline
  - `\t` tabulator
  - `\r` carriage return
  - `\b` backspace
  - `\"` escape double quote
  - `\0` end string

Constant (cont)

- Syntax
  - Case 1: **#define <name_const> <value>**
    - **Example:**
      - **#define PI 3.14**
  - Case 2: **const <data_type> <name_const> = <value>;**
    - **Example:**
      - **const float PI = 3.14;**

Variable

- A variable specifies a area of memory that contains a value of a given type that can be modified by a program
- sizeof() is a function that returns the size of a given variable in bytes (the size depends on the type of the variable)
- Variables should be initialized before they are used (e.g., in the declaration) otherwise the variables contain a random value (or default value depends on compiler)

Variable (cont)

- Syntax:
  - <data_type> name_var [=init_value];
- Example:
  - int y=5;
  - unsigned int z;

Operators

- Arithmetic: **+, -, *, /, %**
- Relational comparisons: **>, >=, <, <=, ==, !=**
- Logical operators: **&&** (and), **||** (or), **!** (not)
- Assignment: **=**
- Increment and decrement: **++, --**
- Conditional evaluation: **? :**

a = (5 > 2) ? 10 : 5;

a = ???

Bit - Operators

- Include: & (and), | (or), ^ (xor), ~ (not), << (shift left), >> (shift right)

Bit1	Bit2	Bit1 & Bit2	Bit1 ^ Bit2	Bit1 | Bit2
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	1

- And bit:

```
char a = 10;
```

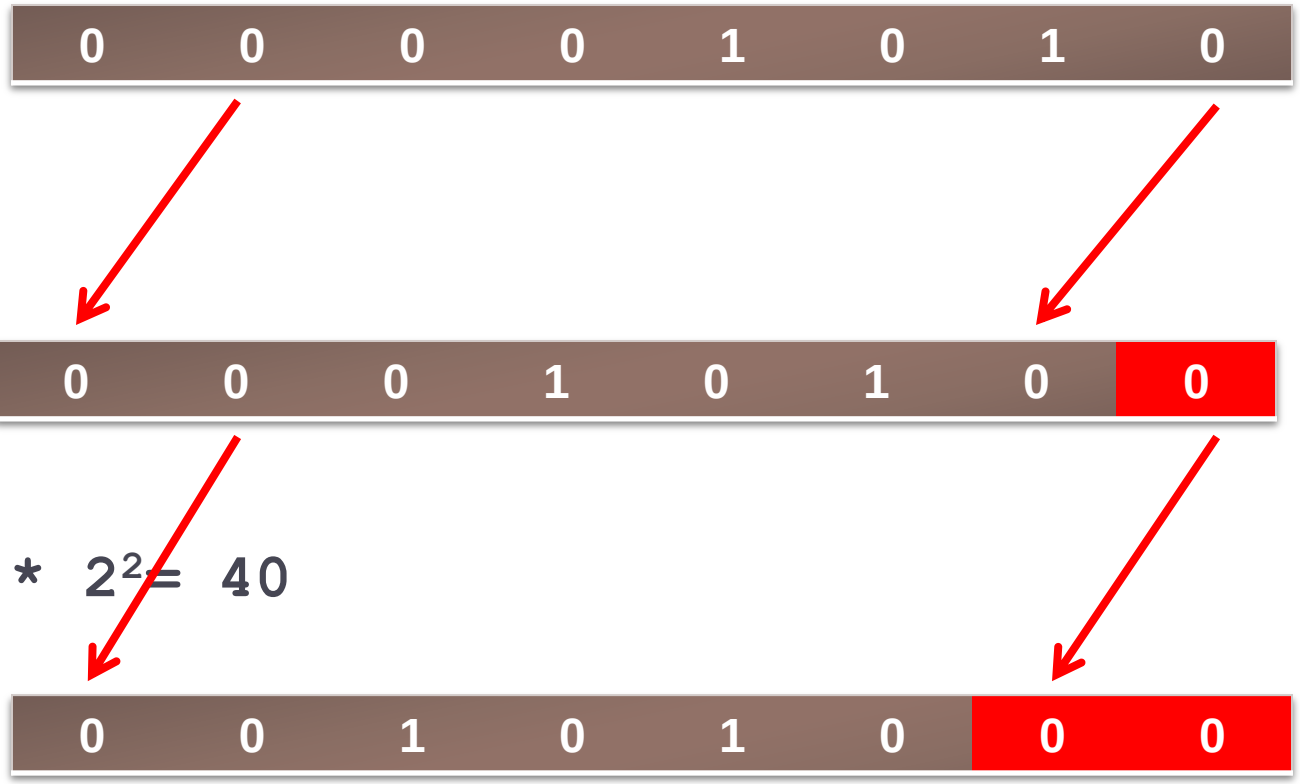

```
char b = 12;
```


```
char c = a & b;
```

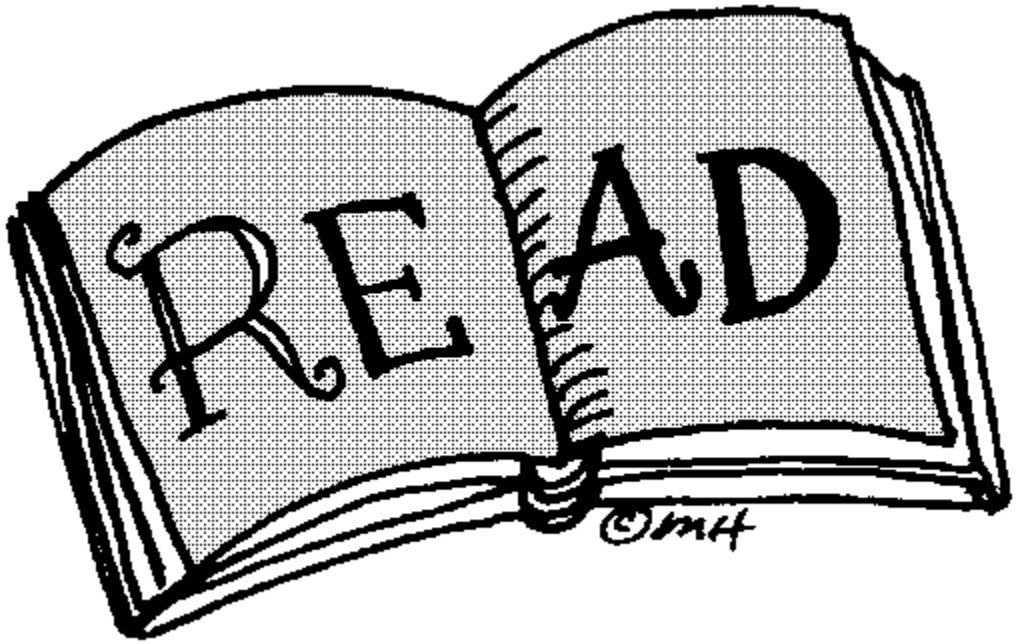
Bit - Operators

- Shift bit:

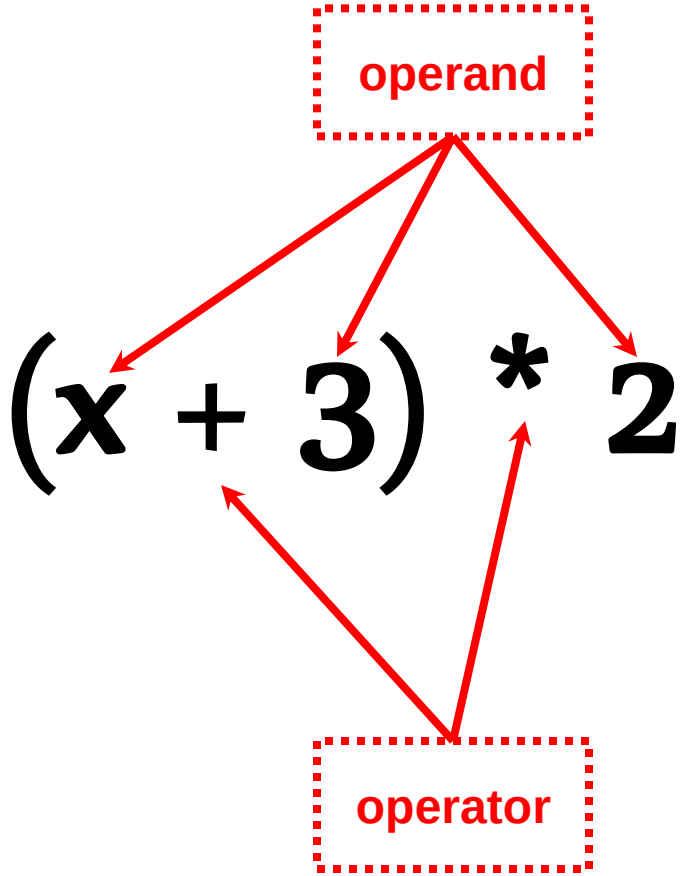
```
char a = 10;
```



Priority of operators



Expression



Type convertions (casts)

- In C program, the type of a value can be changed during the run time of a program, this is known as type convertion or type cast.
- The change can be explicit (Programers do it)
  - **<data_type_new> = (data_type_new) <data_type_old>;**
  - **Example:**

```
int a = 5;
float b = (float) a;
```

- Or the change can be implicit (Compiler does it)
  - **int → long →float → double → long double**
  - **Example**

```
int a = 5;
float b = 6.0;
float  c = a + b; // a will be casted to float
```

Questions and answers