BỘ CÔNG THƯƠNG
ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH

Bài giảng

NGÔN NGỮ LẬP TRÌNH C

THE C PROGRAMMING LANGUAGE

Lecturer : Le Ngoc Tran, PhD
Email : lengoctran@iuh.edu.vn

INDUSTRIAL UNIVERSITY OF HOCHIMINH CITY

CHƯƠNG 2

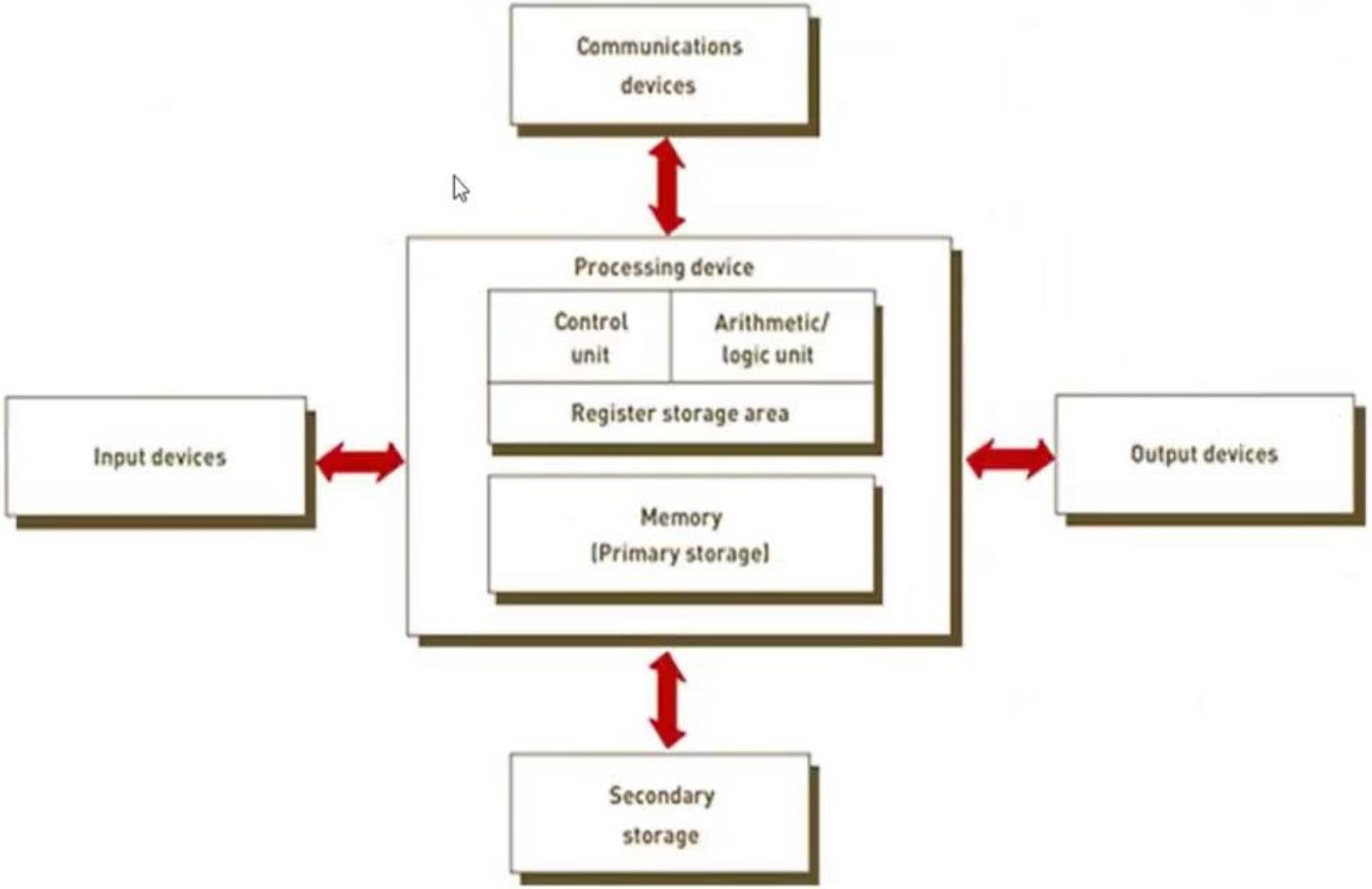
CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU TRONG C

NỘI DUNG CHƯƠNG 2

- 2.1. Biến và hằng trong lập trình c
- 2.2: Các kiểu dữ liệu trong ngôn ngữ C
- 2.3.Nhập/Xuất dữ liệu trong lập trình C
- 2.4. biểu thức (expression) và toán tử (operator)
- 2.5. chiến lược giải quyết vấn đề trong lập trình c & gỡ lỗi

2.1.Biến và hằng trong lập trình c

2.1.1. Kiến trúc logic của máy tính



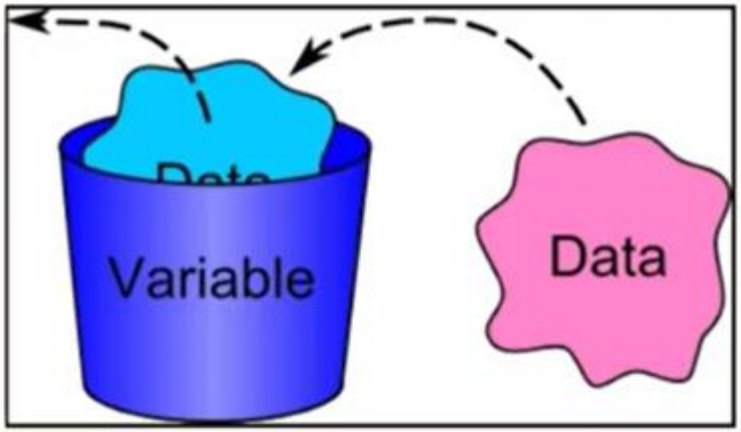
HARDWARE COMPONENTS

2.1.2. Vì sao phải có biến?

- Trả lời: khi khởi động một chương trình (game), thì máy tính cần đến một nơi để chứa chương trình (game) đó. Chỗ để chứa người ta gọi là bộ nhớ (RAM)
- Khi bộ nhớ (RAM) càng lớn thì khả năng chứa sẽ lớn hơn, và giúp cho máy tính hoạt động ổn định hơn khi vận hành những chương trình lớn.
- Tương tự, một chương trình cần sử dụng rất nhiều dữ liệu, và điều quan trọng là chúng ta cần có một nơi để chứa dữ liệu đó, gọi là **biến - variable**

2.1.3. Biến

- Định nghĩa:
  - Biến là tên của một vùng nhớ, để chứa dữ liệu.
  - Những dữ liệu có thể do người dùng nhập vào, các dữ liệu trung gian cần nơi chứa để tính toán, hay là các kết quả đầu ra sau khi đã tính toán.
- Giải sử: bạn mốn nhập vào 1 giá trị là 10 và 1 giá trị là 2 để yêu cầu máy tình xử lý phép toán 10x2.



2.1.3. Biến

- Cách khai báo biến:

Cú pháp

<kiểu dữ liệu> <tên biến>;

<kiểu dữ liệu> <tên biến 1>, <tên biến 2>;

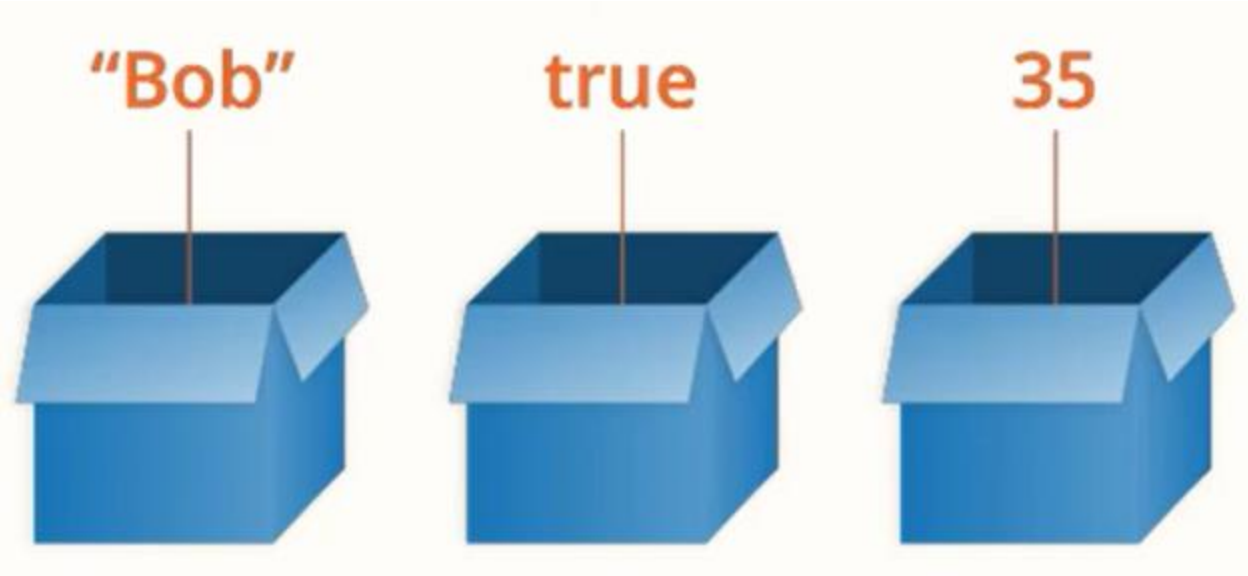
Ví dụ

int a; int b;

int a,b; //Khai báo 2 biến cùng kiểu dữ liệu

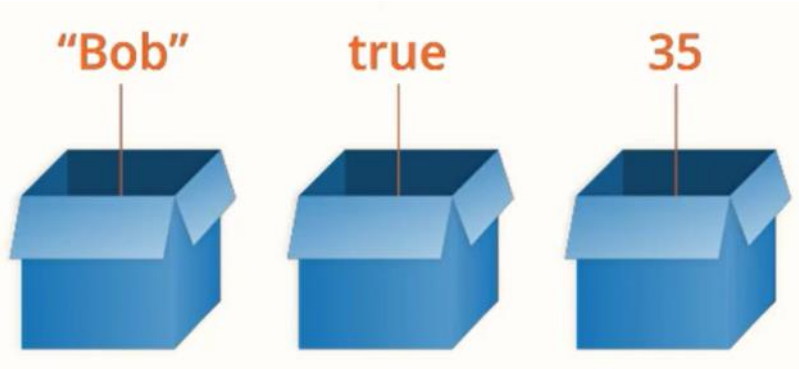
2.1.3. Biến – đặc điểm

- Mỗi một biến có kiểu dữ liệu riêng xác định giá trị và phạm vi giá trị có thể lưu vào biến
- Biến phải được khai báo rồi mới sử dụng
- Tên biến được phân biệt chữ hoa và chữ trường



2.1.3. Biến – quy ước đặt tên

- Phải bắt đầu bằng một chữ cái, hoặc dấu _
- Tên biến có thể gồm: chữ cái, chữ số, dấu _
- Tên biến không được trùng với các từ khóa của C
- Nên đặt tên biến theo cú pháp **camelCase**:
  - Ví dụ: double **toanTu1**; double **toanTu2**;



2.1.4. Từ khóa

- Từ khóa là các từ dành riêng cho ngôn ngữ C
  - Không dùng từ khóa đặt tên định danh: biến, hàm, hằng
- Một số từ khóa cơ bản trong C.

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

- **Demo sử dụng các biến: variable.c**

2.1.5. Hằng số

- Hằng số (constant)
  - Là các giá trị không thể thay đổi trong khi chương trình thực hiện
  - Có thể có một kiểu dữ liệu bất kỳ nào như kiểu dữ liệu: *số nguyên*, *số thực*, *ký tự hay chuỗi*.
- Ví dụ:
  - Số PI=3.141592
  - E=2.71828

2.1.5. Hằng số

- Các định nghĩa hằng số
  - #define **ten_hang** **gia_tri**
  - const **keu_du_lieu** **ten_hang** = **gia_tri**;
- Ví dụ:
  - #define PI 3.14
  - const int PI=3.14;

- **Demo sử dụng các hằng số: constant.c**

2.1.6. Nhập xuất dữ liệu

- Hàm **printf** để xuất dữ liệu ra màn hình console
- Hàm **scanf** dùng đọc giá trị do người dùng nhập vào từ bàn phím

demo_bai2.c

```c
#include <stdio.h>
int main(){
    printf("Xin moi nhap vao tuoi: ");
    int tuoi=0;
    scanf("%d",&tuoi);
    printf("Tuoi cua ban la: %d",tuoi);
    return 0;
}
```

2.1.7. Chuỗi định dạng dữ liệu

- Chuỗi định dạng dữ liệu dùng cho các hàm printf và scanf để định dạng xuất hay nhập liệu tương ứng
- Chuỗi định dạng dữ liệu gồm
  - %c: kiển char 1 ký tự
  - %d: kiểu int
  - %u: kiểu unsigned int
  - %f: kiểu float
  - %lf: kiểu double
  - %s: Chuỗi ký tự

[*] demo_bai2.c

```c
#include <stdio.h>
int main(){
    printf("Xin moi nhap vao diem trung binh: ");
    float diemTrungBinh;
    scanf("%f",&diemTrungBinh);
    printf("Diem trung binh cua ban la: %f",diemTrungBinh);
    return 0;
}
```

- **Demo sử dụng hàm printf và scanf: gpa.c**

2.2.Các kiểu dữ liệu trong lập trình c

2.2. Giới thiệu kiểu dữ liệu trong ngôn ngữ C

- Có thể định nghĩa loại dữ liệu (Data) máy tính có thể xử lý.
- Các loại dữ liệu mà máy tính có thể xử lý có rất nhiều, tồn tại dưới nhiều dạng khác nhau
  - Là dạng số, dạng ký tự (các chữ cái, ký hiệu đặc biệt,...)
- Trong C có 3 kiểu dữ liệu cơ sở như sau:
  - **Kiểu số nguyên**: giá trị của nó là các số nguyên như 2912, -1706,...
  - **Kiểu số thực**: giá trị của nó là các số thực như 3.1415, 29.12,-17.06,...
  - **Kiểu ký tự**: 256 ký tự trong bảng mã ASCII.

2.2.1 Kiểu số nguyên

- Các kiểu số nguyên (có dấu):

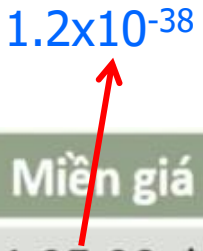
Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
Short	2	–32.768 đến 32.767
int	2 hoặc 4	-32,768 đến 32,767 hoặc -2,147,483,648 đến 2,147,483,647
long	8	-9223372036854775808 to 9223372036854775807

- Kiêu số nguyên (không dấu)

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
unsigned short	2	0 đến 65.535
unsigned int	2 hoặc 4	0 đến 65,535 hoặc 0 đến 4,294,967,295
unsigned long	8	0 đến 18446744073709551615

2.2.2. Kiểu số thực

- Các kiểu số thực (floating-point)
  - Ví dụ: 17.06



Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
float	4	1.2E-38 đến 3.4E+38
double	8	2.3E-308 đến 1.7E+308
Long double	10	3.4E-4932 đến 1.1E+4932

  - Float : độ chính xác 6 số lẻ
  - Double: độ chính xác đến 15 số lẻ
  - Long double: độ chính xác lên đến 19 số lẻ.

2.2.3. Kiểu Char (ký tự)

- Kiểu char: Miền giá trị: 256 ký tự trong bảng mã ASCII.
  - Ký tự chữ cái: a b c X Y Z

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
Char	1	-127 đến 128
Unsignchar	1	0 đến 255

- Đặc điểm
  - Tên kiểu: Char
  - Miền giá trị: 256 ký tự trong bảng mã ASCII
  - Chính là kiểu số nguyên do:
    - Lưu tất cả dữ liệu ở dạng số
    - Không lưu trực tiếp ký tự mà chỉ lưu mã ASCII của ký tự đó
- Ví dụ
  - Lưu số 65 tương đương với ký tự 'A'...
  - Lưu số 97 tương đương với ký tự 'a'.

2.2.3. Kiểu Char (ký tự)

- Bảng mã ASCII

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

2.2.4. Ý nghĩa của kiểu dữ liệu

- Cách xác định khoảng lưu trữ của một kiểu dữ liệu:
  - Ví dụ: Kiểu short có độ lớn là 2 byte (1 byte=8bit)
    - Thì khả năng lưu trữ lớn nhất sẽ là 2^16-1=65535
  - Ví dụ: kiểu char có độ lớn là 1 byte (1 byte=8bit)
    - Thì khả năng lưu trữ lớn nhất sẽ là 2^8-1=255
- Tùy thuộc vào giá trị chúng ta cần lưu trữ là gì, để xác định kiểu dữ liệu phù hợp.

2.2.5. Kiểu luận lý

- Đặc điểm
  - C ngầm định một cách không tường minh:
    - False (sai): giá trị 0
    - True (đúng): giá trị khác 0
  - C++: bool
- Ví dụ:
  - 0 (false), 1(true), 2(true), 2.5(true)
  - 1>2(0,false), 1<2(1,true)

2.2.6. Bảng các kiểu dữ liệu của C

- 4 kiểu dữ liệu chuẩn: **char, int, float và double**.
- Các dạng bổ sung **signed, unsigned, short, long** thêm vào trước các kiểu dữ liệu chuẩn để tạo ra nhiều kiểu khác nhau.

Kiểu dữ liệu	Kích thước	Ý nghĩa
char	1 byte	Kiểu ký tự (-128 ÷ 127)
int	2 byte hoặc 4 byte	Kiểu số nguyên
short	2 byte	Kiểu số nguyên
long	4 byte	Kiểu số nguyên
unsigned char	1 byte	Kiểu ký tự không dấu (0 ÷ 255)

2.2.6. Bảng các kiểu dữ liệu của C

unsigned int	4 byte	Kiểu số nguyên không dấu
unsigned short	2 byte	Kiểu số nguyên không dấu
unsigned long	4 byte	Kiểu số nguyên không dấu
float	4 byte	Kiểu số dấu chấm động (số thực)
double	8 byte	Kiểu số dấu chấm động (số thực)
long double	12 byte	Kiểu số dấu chấm động (số thực)

2.3.Nhập/Xuất dữ liệu trong lập trình c

2.3.1. Khái niệm Nhập Xuất chuẩn

- Thư viện chuẩn của C cung cấp các hàm cho phép nhập và xuất dữ liệu
  - Các hàm cũng hỗ trợ thao tác chuỗi và ký tự
  - Nhập chuẩn (Standard Input) thường là từ bàn phím
  - Xuất chuẩn (Standard Output) thường là màn hình (console)
  - Nhập xuất có thể chuyển hướng đến/từ tập tin thay vì các thiết bị chuẩn

2.3.2. Tập tin header<stdio.h>

- **#include<stdio.h>**
  - Là lệnh tiền xử lý (Preprocessor command)
  - **Stdio.h** là tập tin Header
    - Chứa macro cho nhiều chức năng nhập xuất dữ liệu
    - Gồm các hàm printf, scanf,putchar,getchar

2.3.3. Nhập xuất có định dạng

- **printf()** dùng xuất dữ liệu có định dạng
- **scanf()** dùng nhập dữ liệu có định dạng
- Format specifiers xác định định dạng mà các giá trị của biến được nhập và hiển thị.
- Cú pháp: **printf("formatted string", [argument list])**
  - Argument list bao gồm: hằng giá trị, biến, biểu thức hay lời gọi hàm phân tách nhau bằng dấu phẩy (,)
  - Formatted string: chuỗi định dạng gồm: các ký tự, lệnh định dạng, các ký tự không thể in (như khoảng trắng, tab, xuống dòng)

2.3.4. Mã định dạng

Format Specifier	Type
%c	Character
%d	Signed integer
%e or %E	Scientific notation of floats
%f	Float values
%g or %G	Similar as %e or %E
%hi	Signed integer (short)
%hu	Unsigned Integer (short)
%i	Unsigned integer
%l or %ld or %li	Long
%lf	Double

2.3.4. Mã định dạng

Format Specifier	Type
%lu	Unsigned int or unsigned long
%lli or %lld	Long long
%llu	Unsigned long long
%o	Octal representation
%p	Pointer
%s	String
%u	Unsigned int
%x or %X	Hexadecimal representation
%n	Prints nothing
%%	Prints % character

Ví dụ về định dạng (Formated)

CHƯƠNG TRÌNH DEMO **formatted-ouput.c**

2.3.5. Modifier trong hàm printf()

- '-' Modifier
  - dữ liệu được canh trái
- Field Width Modifier
  - Có thể được dùng với kiểu dữ liệu float, double, mảng ký tự
  - có thể dùng với kiểu integer để định nghĩa kích thước của trường dữ liệu
- Precision Modifier
  - Có thể được dùng với kiểu float, double, mảng ký tự
  - nếu dùng với float, double, xác định số lượng ký tự số được in ở bên phải của giá trị

2.3.5. Modifier trong hàm printf()

- '0' Modifier
  - ký tự đệm ngầm định trong trường là khoản trắng
  - được dùng để thay ký tự đệm ngầm định
- '*'Modifier
  - Nếu người dùng không muốn xác định độ dài của trường trước nhưng muốn chương trình xác định, thì modifier được dùng.

Ví dụ sử dụng Modifier

CHƯƠNG TRÌNH DEMO **modifier-formatted-ouput.c**

2.3.6. Hàm scanf()

- Hàm scanf() dùng đọc dữ liệu do người dùng nhập vào
- Cú pháp: **scanf("Formatted String",argument list)**
  - Formatted string: giống với printf()
  - Argument list: chứa danh sách các con trỏ (Pointer) trỏ tới các biến.

Ví dụ sử dụng Hàm scanf()

CHƯƠNG TRÌNH DEMO **input.c**

2.3.7. Buffered I/O (Nhập/Xuất có dùng bộ đệm dữ liệu)

- Được dùng để đọc và ghi các ký tự ASCII
- Bộ đệm (buffer) là vùng lưu trữ tạm thời ở trong bộ nhớ hoặc ở trên Card điều khiển thiết bị
- Buffered I/O được chia thành:
  - Console I/O
  - Buffered File I/O

2.3.8. Console I/O

- Các hàm Console I/O chuyển hoạt động của chúng đến các thiết bị nhập/xuất chuẩn của hệ thống
- Các hàm console I/O gồm:
  - getchar(): đọc 1 ký tự từ bàn phím
  - putchar(): hiển thị 1 ký tự ra màn hình

CHƯƠNG TRÌNH DEMO **buffered-io.c**

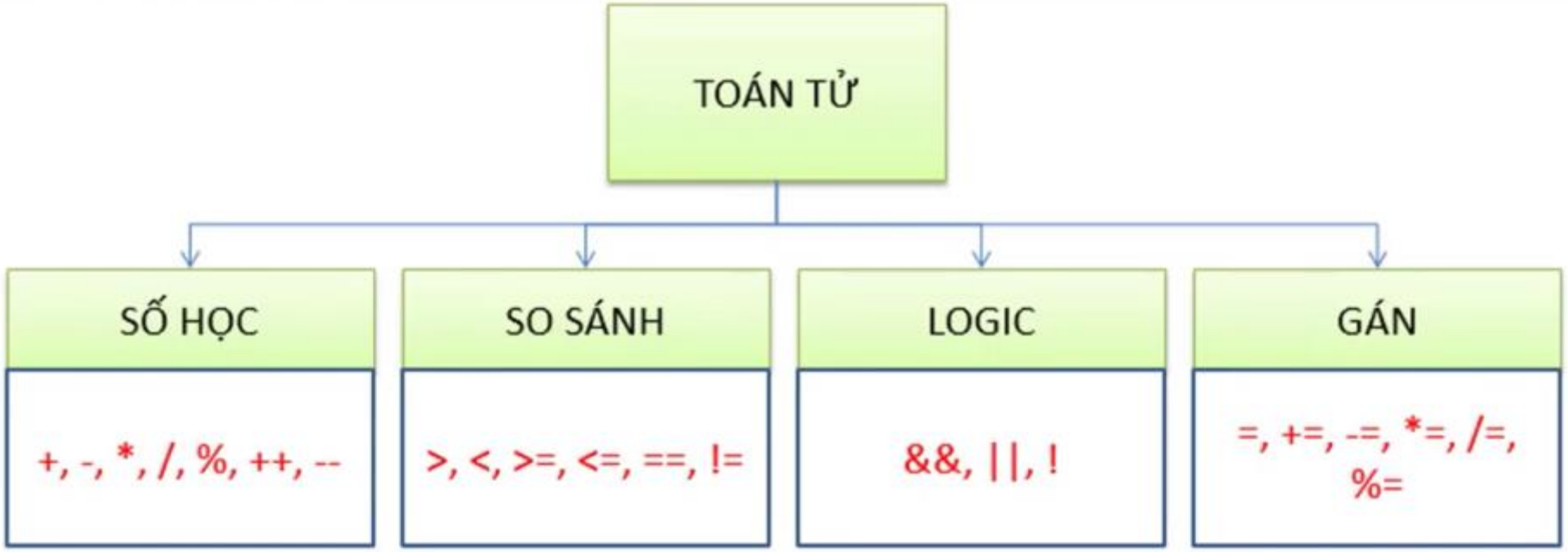
2.4. biểu thức (expression) và toán tử (operator)

2.4.1. Biểu thức (expression)

- Biểu thức là tập hợp các toán tử (operator) và toán hạng (operand)
- Ví dụ: $5 + 6$
  - Trong biểu thức
    - **+** là toán tử (thực hiện phép cộng trên toán hạng)
    - **5** và **6** là toán hạng

2.4.2. Toán tử

- Trong C, có thể phân ra 4 nhóm toán tử sau



2.4.2. Toán tử - Toán tử số học

- Toán tử số học là các phép toán thao tác trên các số nguyên và số thực

+	Tính tổng của 2 số
-	Tính hiệu của 2 số
*	Tính tích của 2 số
/	Tích thương của 2 số
%	Thực hiện chia có dư của 2 số
++	Tăng giá trị của biến lên 1 đơn vị
--	Giảm giá trị của biến xuống 1 đơn vị

2.4.2. Toán tử - Toán tử số học 2 ngôi

- Toán tử 2 ngôi là các toán tử thao tác trên 2 toán hạng
- Ví dụ: tìm kết quả cho các phép tính sau

Biểu thức	Đáp án
15 + 5 + 9 / 3 – 10 + 2 * 2	17
25 / 5 + 6 – 4 % 2 + 3 * 6	29
14 + 16 + 30 % 5 - 8	22
33 – 3 + 4 – (12 + 3) / 3	29
12 + 14 – 8 % 5 + 12 * 2	47
13 + 14 % 3 * 2 – 15	2

2.4.2. Toán tử - Toán tử 1 ngôi (Unary operator)

- Toán tử 1 ngôi là các toán tử chỉ có 1 toán hạng
  - **++** (tăng 1 đơn vị), **--** (giảm 1 đơn vị)
  - Đặt trước toán hạng
    - ✓ Ví dụ **++**x hay **--**x: thực hiện tăng/giảm **trước**.
  - Đặt sau toán hạng
    - ➢ Ví dụ x**++** hay x**--** : thực hiện tăng/giảm **sau**.
- Ví dụ: int a=8

```
a++; //a bằng 9
```

CHƯƠNG TRÌNH DEMO SỬ DỤNG TOÁN TỬ

arithmetic-opts.c

2.4.3. Toán tử so sánh (Comparison Operator)

- Toán tử so sánh là các phép toán so sánh hai toán hạng

==	So sánh bằng
>	So sánh lớn hơn
>=	So sánh lớn hơn hoặc bằng
<	So sánh nhỏ hơn
<=	So sánh nhỏ hơn hoặc bằng
!=	So sánh khác

2.4.3. Toán tử so sánh (Comparison Operator)

- Tìm kết quả cho các biểu thức

Biểu thức
25 + 3 * 2 != 18
5 * 3 + 2 == 11
12 - 8 <= 3 * 2
5 > 14 / 2
6 + 1 >= 7
36 / 2 – 14 != 35
45 – 5 + 12 < 19

2.4.3. Toán tử so sánh (Comparison Operator)

- Tìm kết quả cho các biểu thức

Biểu thức	Kết quả
25 + 3 * 2 != 18	true
5 * 3 + 2 == 11	false
12 - 8 <= 3 * 2	true
5 > 14 / 2	false
6 + 1 >= 7	true
36 / 2 – 14 != 35	true
45 – 5 + 12 < 19	false

CHƯƠNG TRÌNH DEMO SỬ DỤNG TOÁN TỬ SO SÁNH **arithmetic-opts.c**

2.4.4. Toán tử logic (Logic Operator)

- Là sự kết hợp của 2 hay nhiều biểu thức
- Kết quả trả về của toán tử logic là giá trị đúng (True) hoặc sai (false)

Toán tử logic	Kết quả trả về
&&	Trả về giá trị true khi tất cả biểu thức tham gia biểu thức có giá trị true
||	Trả về giá trị true khi có 1 biểu thức tham gia biểu thức có giá trị là true
!	Lấy giá trị phủ định của biểu thức

2.4.4. Toán tử logic &&

- Ví dụ: Xét học bổng cho học sinh dựa vào DTB và Hạnh kiểm

DTB >=8 && HK ==“Tốt”

Giá trị của DTB	Toán tử logic	Giá trị của HK	Kết quả biểu thức
Điểm trung bình >=8	&&	Hạnh kiểm = tốt	**True (đạt học bổng)**
Điểm trung bình <8	&&	Hạnh kiểm = tốt	**False (không đạt học bổng)**
Điểm trung bình >=8	&&	Hạnh kiểm = khá	**False (không đạt học bổng**
Điểm trung bình <8	&&	Hạnh kiểm = khá	**False (không đạt học bổng)**

2.4.4. Toán tử logic ||

- Ví dụ: xét kết quả đầu vào của sinh viên dựa trên tổng điểm của một số môn học:

Toán + lý + hóa >=21 || Toán + hóa + sinh >=21

Biểu thức 1	Toán tử logic	Biểu thức 2	Kết quả
True	||	True	True (đậu)
True	||	False	True (đậu)
False	||	True	True (đậu)
False	||	False	False (không đậu)

2.4.5. Bảng luận lý của toán tử AND (&&) và OR (I I)

Toán tử &&		
Biểu thức A	Biểu thức B	A && B
True	True	True
True	False	False
False	True	False
False	False	False

Toán tử ||		
Biểu thức A	Biểu thức B	A || B
True	True	True
True	False	True
False	True	True
False	False	False

CHƯƠNG TRÌNH DEMO SỬ DỤNG TOÁN TỬ SO SÁNH **logic-opts.c**

2.4.6. Toán tử logic Bitwise

- Toán tử logic Bitwise xử lý dữ liệu dưới dạng nhị phân (Binary) tương ứng.

Toán tử	Kết quả
AND (NUM1 & NUM2)	Trả về 1 nếu cả hai toán hạng là 1
OR (NUM1 | NUM2)	Trả về 1 nếu có một toán hạng là 1
NOT (~ NUM1)	Đảo giá trị của toán hạng (Từ 0 thành 1 và 1 thành 0)
XOR (NUM1 ^ NUM2)	Trả về 1 nếu các toán hạng khác giá trị nhau (nếu num1 = 0 và num2 = 1 hoặc num1 = 1 và num2 = 0

2.4.6. Toán tử logic Bitwise

- Ví dụ về toán tử logic Bitwise

10 & 15→ 1010 & 1111→1010 → 10

10 | 15→ 1010 | 1111→1111 → 15

10 ^ 15→ 1010 ^ 1111→0101 → 5

~ 10 → ~1010 →1011 → -11

2.4.7. Toán tử gán (Assignment Operator)

- Toán tử gán dùng gán giá trị, biến hay biểu thức cho một biến
- Cú pháp
  - <biến> = <giá trị>;
  - <biến> = <biến>;
  - <biến> = <biểu thức>;
- Ví dụ
```
int a = 5;
int b = a;
int c = a+b
```

2.4.8. Toán tử gán rút gọn (Shorthand assignment)

- Các phép gán

Phép gán	Ví dụ:	Kết quả:
+=	a += 10 (a = a + 10)	a == 30
-=	a -= 10 (a = a - 10)	a == 20
*=	a *= 10 (a = a * 10)	a == 200
/=	a /= 10 (a = a / 10)	a == 20
%=	a %= 10 (a = a % 10)	a == 0

CHƯƠNG TRÌNH DEMO SỬ DỤNG TOÁN TỬ GÁN **assignment-opts.c**

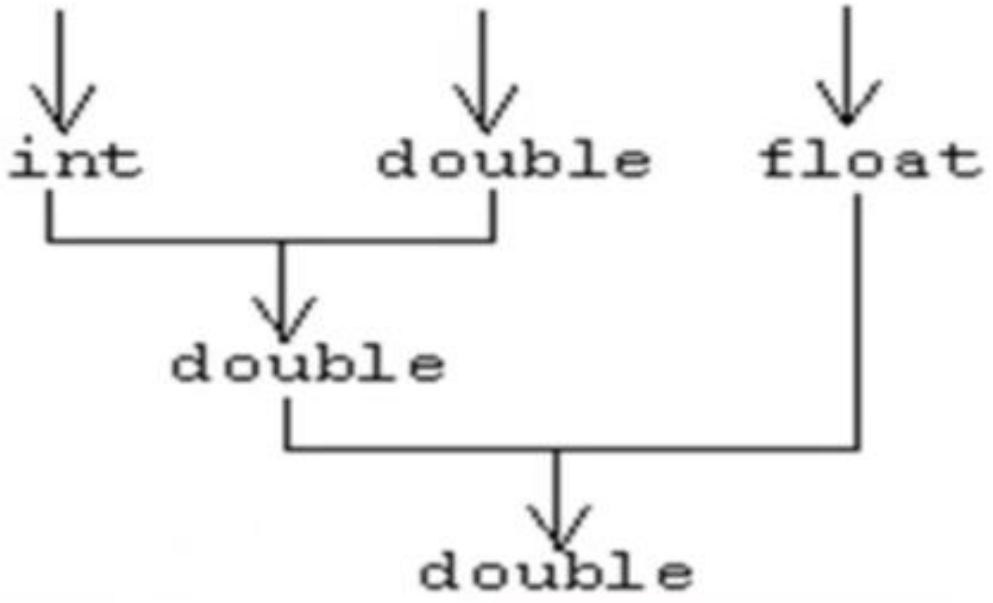
2.4.8. Chuyển kiểu (Type conversion)

- Chuyển kiểu tự động thực hiện khi các kiểu dữ liệu thuộc cũng nhóm (nhóm số nguyên, số thực) và kiểu dữ liệu có kích thước nhỏ sang kiểu dữ liệu có kích thước lớn hơn.
- Chuyển kiểu tự động được thực hiện khi ước lượng các biểu thức với các trường hợp sau:
  - Kiểu **char** và **short** được chuyển thành **int** và **float** chuyển thành **double**
  - Nếu một trong hai toán hạng là **double** thì kiểu của toán hạng còn lại chuyển thành **double** và kết quả là **double**
  - Nếu một trong hau toán hạng là **long**, thì kiểu của toán hạng còn lại chuyển thành **long** và kết quả là **long**
  - Nếu một trong hai toán hạng là **unsigned**, thì kiểu của toán hạng còn lại chuyển thành **unsigned** và kết quả là **unsigned**

2.4.8. Chuyển kiểu (Type conversion)

- Ví dụ:

```
char ch;
int i;
float f;
double d;
result = (ch/i) + (f*d) - (f+i);
```



2.4.9. Ép kiểu (Type casting)

- Ép kiểu được thực hiện khi chuyển từ kiểu dữ liệu có kích thước lớn hơn sang kiểu có kích thước nhỏ hơn và kiểu số thực sang kiểu số nguyên
- Ép kiểu cần phải được khai báo tường minh:

(type) <biến hoặc biểu thức>

- Ví dụ:

```
float x,f
f = 3.14159;
x = (int) f;
```

2.4.10. Độ ưu tiên của các toán tử

- Độ ưu tiên thiết lập tập hợp phân cấp các toán tử được ưu tiên thực hiện
- Thứ tự ưu tiên thực hiện của toán tử trong biểu thực sẽ thay đổi khi sử dụng dấu ngoặc ()
- Độ ưu tiên của các toán tử số học

Loại toán tử	Toán tử	Thứ tự ước lượng
Một ngôi (Unary)	- ++ --	Từ Phải sang trái
Hai ngôi	* / %	Từ Trái sang phải
Hai ngôi	+ -	Từ Trái sang phải
Hai ngôi	=	Từ Phải sang trái

2.4.10. Độ ưu tiên của các toán tử

- Độ ưu tiên của toán tử logic

Toán tử	Độ ưu tiên
NOT (!)	1
AND (&&)	2
OR (||)	3

- Khi biểu thức sử dụng nhiều loại toán tử, độ ưu tiên của các toán tử như sau:

Toán tử	Độ ưu tiên
Số học	1
So sánh	2
Logic	3

2.4.10. Độ ưu tiên của các toán tử

- Tính giá trị của biểu thức sau:

$$2*3+4/2 > 3 \text{ AND } 3<5 \text{ OR } 10<9$$

- Các bước tính toán biểu thức:

$$[2*3+4/2] > 3 \text{ AND } 3<5 \text{ OR } 10<9$$

$$[[2*3]+[4/2]] > 3 \text{ AND } 3<5 \text{ OR } 10<9$$

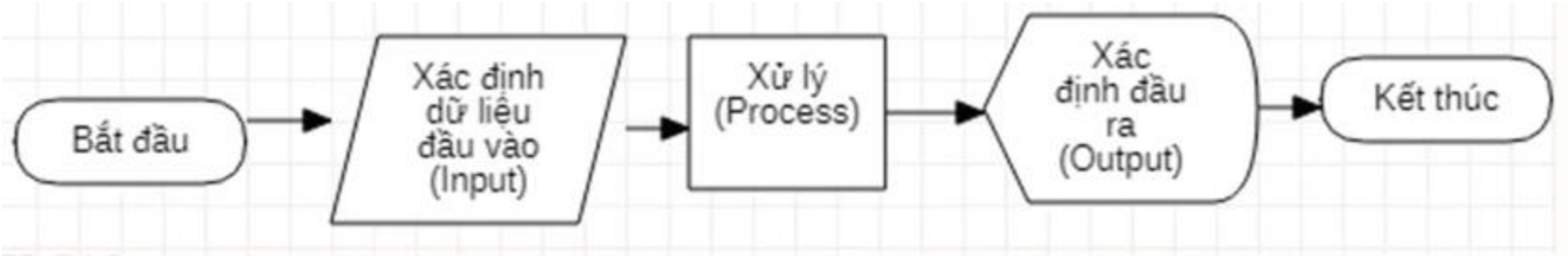
$$[6+2] >3 \text{ AND } 3<5 \text{ OR } 10<9$$

$$[8 >3] \text{ AND } [3<5] \text{ OR } [10<9]$$

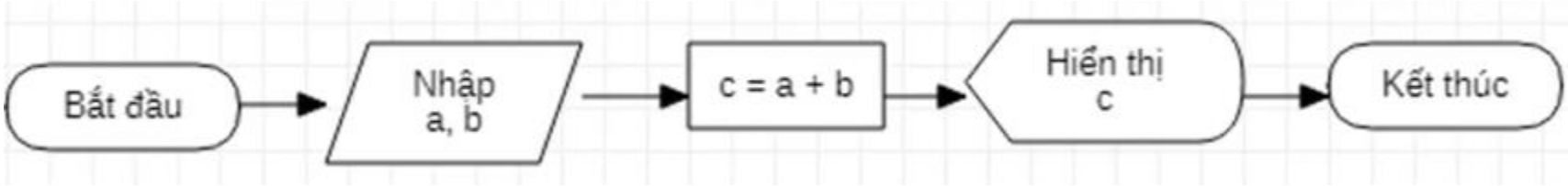
2.5. chiến lược giải quyết vấn đề trong lập trình c & gỡ lỗi

2.5.1. Cách phân tích và giải quyết bài toán

- Để giải quyết một bài toán bằng lập trình, chúng ta cần phải phân tích kỹ và xác định các bước



- Ví dụ: xây dựng phương trình tính tổng hai số



2.5.1. Cách phân tích và giải quyết bài toán

- Sau khi đã phân tích bài toán, chúng ta cần biểu diễn cách giải bài toán dưới các dạng
  - Ngôn ngữ tự nhiên (natural languages)
  - Mã giả (pseudocode)
  - Lưu đồ thuật toán (flowcharts)

2.5.2. Mã giả (Pseudo code)

- Ví dụ: Chương trình hiển thị “Hello World”

```
Begin
        display “Hello Worlk”
End
```

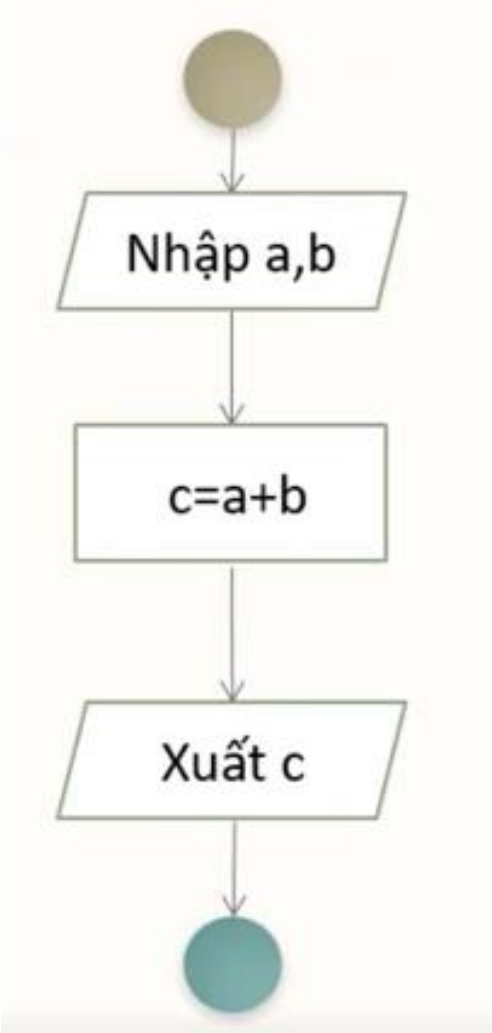
2.5.3. Lưu đồ thuật toán (flowchart)

- Lưu đồ thuật toán là công cụ dùng để **biểu diễn thuật toán**, mô tả nhập dữ liệu (**input**), xuất dữ liệu (**output**) và luồng xử lý thông qua các **ký hiệu hình học**

STT	Ký hiệu	Ý nghĩa
1		Bắt đầu/ kết thúc
2		Điều kiện rẽ nhánh (lựa chọn)
3		Nhập hoặc xuất
4		Tính toán
5		Luồng xử lý

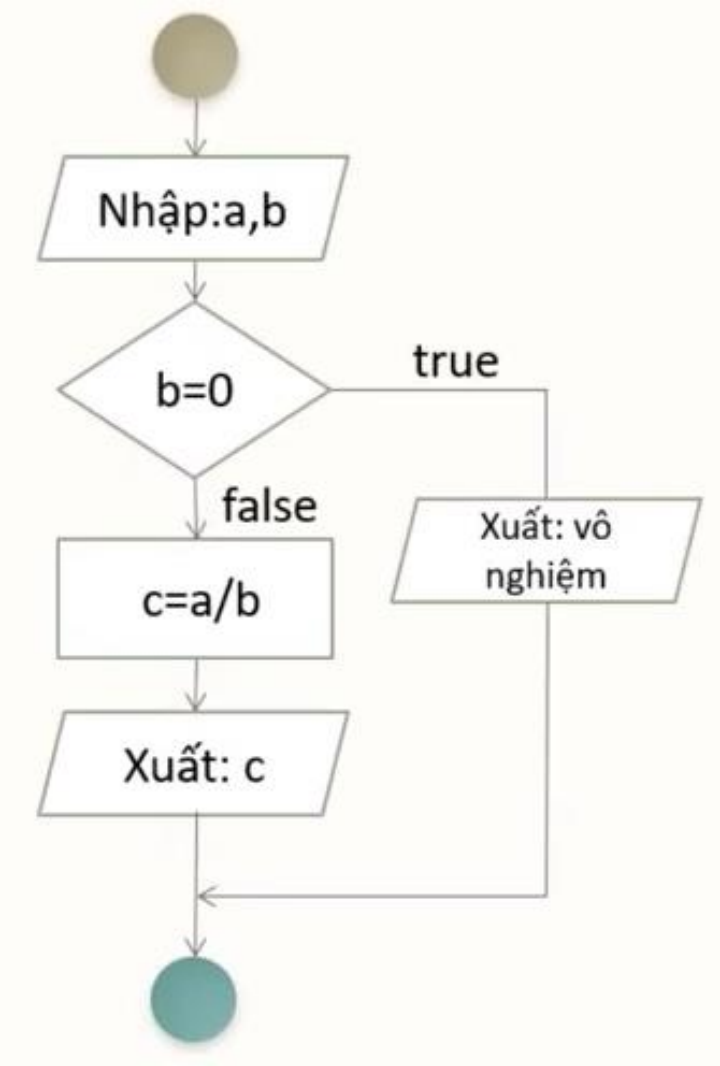
2.5.3. Lưu đồ thuật toán (flowchart)

- Ví dụ: Chương trình cộng 2 số bất kỳ a+b



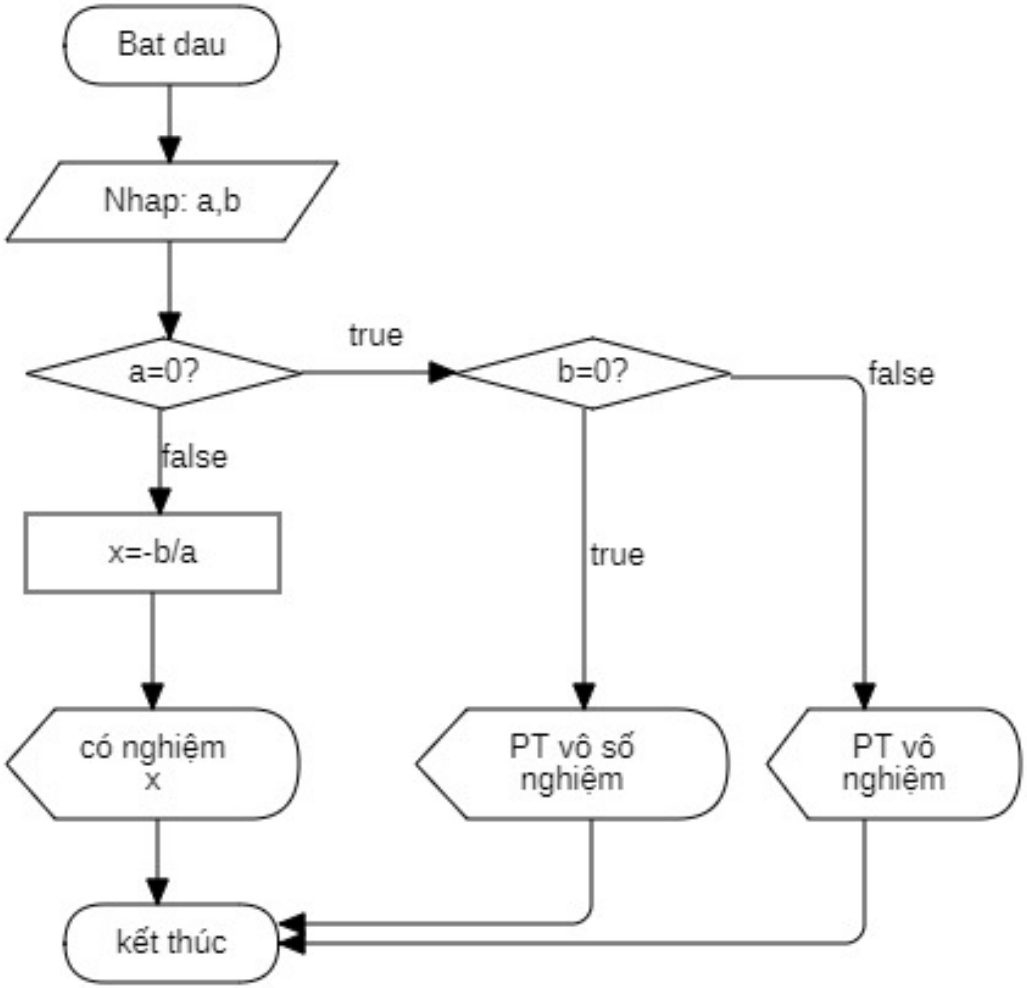
2.5.3. Lưu đồ thuật toán (flowchart)

- Ví dụ: Xây dựng lưu đồ giải phương trình: ax + b =0



2.5.3. Lưu đồ thuật toán (flowchart)

- Biểu diễn Flow chart bằng phần mềm Star UML



2.5.4. Gỡ lỗi chương trình (Debug)

- Khi lập trình, lỗi không thể tránh khỏi
- Khi lỗi xảy ra, lập trình viên cần phải tìm cách gỡ lỗi (Debug)
- Các bước cần thực hiện khi gặp lỗi:
  - Đọc thông báo lỗi để xác định nội dung, nguyên nhân, vị trí gây ra lỗi
  - Rà soát nội dung của chương trình
  - Chèn vào các lệnh hiển thị giá trị để truy vết quá trình chạy
  - Dùng chức năng chạy từng bước của IDE để truy vết kết quả

2.5.4. Gỡ lỗi chương trình (Debug)

CHƯƠNG TRÌNH DEMO GỠ LỖI

debug.c


THANK
YOU