TRƯỜNG ĐẠI HỌC QUỐC TẾ HỒNG BÀNG
KHOA CÔNG NGHỆ THÔNG TIN
oOo

LẬP TRÌNH C CĂN BẢN

(TÀI LIỆU THAM KHẢO)

LÊ VĂN HẠNH

MỤC LỤC

HỆ THỐNG SỐ (*Numeral System*)

1.1. GIỚI THIỆU

- Hệ thống số là một tập các ký hiệu (bảng chữ số) để biểu diễn các số và xác định giá trị của các biểu diễn số
- Ví dụ 1: Hệ thống số La mã
  - Bảng chữ (ký hiệu) và giá trị tương ứng:

Ký hiệu	I	V	X	L	C	D	M
Giá trị đại diện	1	5	10	50	100	500	1000

  - *Quy tắc biểu diễn số*: là viết các chữ số cạnh nhau.
  - *Quy tắc tính giá trị*:giá trị của cặp số được tính:
    - Nếu số có giá trị lớn hơn đứng bên trái thì giá trị chung bằng tổng hai giá trị
    - Nếu số có giá trị nhỏ hơn đứng bên phải thì giá trị chung bằng tổng hai giá trị

```
MLVI = 1000 + 50 + 5 + 1 =1056
MLIV = 1000 + 50 + 5 - 1 = 1054
```

- Ví dụ 2: Hệ đếm thập phân
  - Bảng chữ số: {0,1,2,3,4,5,6,7,8,9}
  - Quy tắc biểu diễn: ghép các chữ số
    - Quy tắc tính giá trị: mỗi chữ số x đứng ở hàng thứ i tính từ bên phải có giá trị là $x.10^{i-1}$. Như vậy một đơn vị ở một hàng sẽ có giá trị gấp 10 lần một đơn vị ở hàng kế cận bên phải.
    - Giá trị của số: là tổng giá trị của các chữ số có tính tới vị trí của nó.

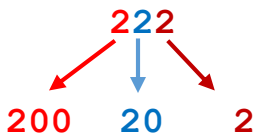
      Giá trị của 3294,5 là: $3.10^3 + 2.10^2 + 9.10^1 + 4.10^0 + 5.10^{-1}$

1.2. PHÂN LOẠI HỆ THỐNG SỐ THEO VỊ TRÍ XUẤT HIỆN

1.2.1. *Hệ đếm theo vị trí*

Là hệ đếm mà giá trị của mỗi chữ số phụ thuộc vào vị trí của nó trong con số (như hệ đếm thập phân)

Ví dụ:



1.2.2. *Hệ đếm KHÔNG theo vị trí*

Mỗi biểu tượng đại diện cho một giá trị giống nhau (như hệ đếm La Mã)

Ví dụ:

```
I đại diện cho 1
IIđại diện cho 2
...
IIIIII đại diện cho 6
```

1.3. HỆ THỐNG SỐ THEO VỊ TRÍ XUẤT HIỆN

1.3.1. *Hệ đếm*

- Là tập hợp các ký hiệu và quy tắc sử dụng tập ký hiệu đó để biểu diễn và xác định các giá trị các số. Giá tri của mỗi số phụ thuộc vào 3 yếu tố: giá trị số, vị trí và cơ số được sử dụng.
- Các hệ đếm:
  - Cơ số **2** (*Binary System*) sử dụng các chữ số {0, 1}
    ký hiệu sử dụng 110101_2 hay 110101_B
  - Cơ số **8** (*Octal System*) sử dụng các chữ số {0, 1, 2, 3, 4, 5, 6, 7}
    ký hiệu sử dụng 723_8 hay 723_O
  - Cơ số **10** (*Decimal System*) sử dụng các chữ số {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
    ký hiệu sử dụng 905_{10} hay 905_D
  - Cơ số **16** (*Hexadecimal System*) sử dụng các chữ số {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
    ký hiệu sử dụng 723_{16} hay 723_H
- Mỗi hệ đếm có một số ký số (*digits*) hữu hạn. Tổng số ký số của mỗi hệ đếm gọi là cơ số (*base* hay *radix*).

Numbering Systems		
System	*Base*	*Digits*
Binary	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

1.3.2. *Biểu diễn số trong các hệ đếm*

- Số K trong hệ đếm cơ số b được biểu diễn bởi:
  $K(b) = a_n.b_n + a_{n-1}.b_{n-1} + \ldots + a_1.b_1 + a_0.b_0 + a^{-1}.b^{-1} + a^{-2}.b^{-2} + \ldots + a^{-m}.b^{-m}$ Với n là vị trí xuất hiện của số theo quy ước:
  - *Đối với phần nguyên (bên trái dấu ngăn cách)*: tính từ 0 và đếm tăng dần từ phải sang trái

    Ví dụ 1: xét số 10110001_2

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	0	1	1	0	0	0	1

    $\Rightarrow 10110001_2 = 128 + 32 + 16 + 1 = 177_{10}$
  - *Đối với phần lẻ (bên phải dấu ngăn cách)*: tính từ -1 và đếm giảm dần từ trái sang phải

    Ví dụ 2: xét số 0.1101_2

2^{-1}	2^{-2}	2^{-3}	2^{-4}
1/2	1/4	1/8	1/16
1	0	1	1

    $\Rightarrow 0.1101_2 = 1/2 + 1/8 + 1/16 = (8+2+1)/16 = 11/16_{10}$

    Ví dụ 3: $345,2_8$

    $= 3 \times 8^2 + 4 \times 8^1 + 5 \times 8^0 + 2 \times 8^{-1}$

    $= [(3x64) + (4x8) + (5x1) + (2x1/8)]_8$

    $= [192 + 32 + 5 + 1/4]_8 = 229,25_{10}$

    Ví dụ 4: $1101,01_2$

    $= 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 + 0x2^{-1} + 1x2^{-2}$

    $= [(1x8) + (1x4) + (0x1) + (1x1) + (0x1/2) + (1x1/4)]_{10}$

    $= [8 + 4 + 0 + 1 + 0 + 0.25] = 13,25_{10}$

1.3.3. *Miền giá trị của số n-bit*

- Số nhị phân gồm n bit được gọi là số n-bit.
- Miền giá trị của số n-bit là từ 0 đến 2^n-1.
- Ví dụ:
  - Số 3-bit có $2^3 = 8$ ⇒ miền giá trị của số nhị phân 3-bit là từ 0 đến 7
  - Số 8-bit có $2^8 = 256$ ⇒ miền giá trị của số nhị phân 8-bit là từ 0 đến 255

1.4. CHUYỂN SỐ TỪ CƠ SỐ NÀY SANG CƠ SỐ KHÁC

1.4.1. *Chuyển số từ cơ số khác sang cơ số 10*

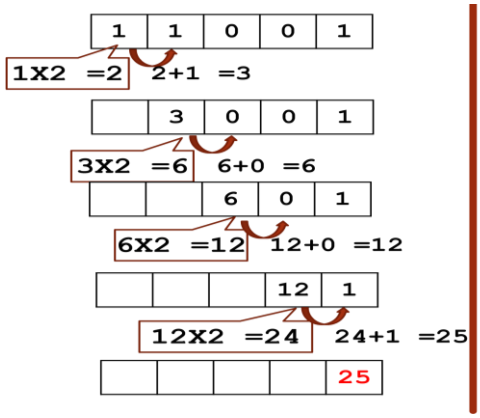
1.4.1.1. *Cách1*

- Các bước thực hiện:
  - **B1**: Xác định giá trị vị trí của mỗi ký số
  - **B2**: Nhân giá trị vị trí với ký số của cột tương ứng.
  - **B3**: Cộng kết quả của các phép tính nhân trong bước 2.
- Ví dụ
  - Ví dụ 1: $11001_2 = ?_{10}$
    $= 1x2^4 + 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0$
    $= 16 + 8 + 0 + 0 + 1 = 25_{10}$
  - Ví dụ 2: $4706.2_{(8)} = ?_{(10)}$
    $= 4x8^3 + 7x8^2 + 0x8^1 + 6x8^0 + 2x8^{-1}$
    $= (4x512) + (7x64) + (0x8) + (6x1) + (2x\ 1/8)$
    $= 2048 + 448 + 0 + 6 + 0.25$
    $= 2502.25_{10}$
- Thực hành: chuyển sang cơ số 10
  - $FE1AC_{16}$
  - 405.42_8
  - 100101.101_2
  - $1DB7_{16}$
  - 1101_{16}
  - 752.24_8
  - 111001101_2
  - 1011_8

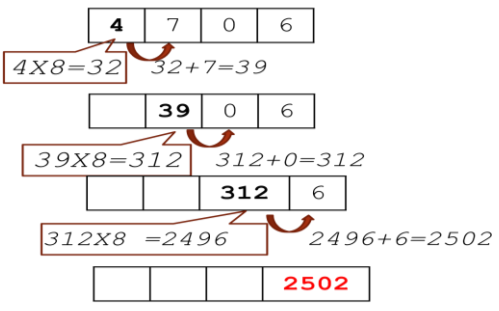
1.4.1.2. *Cách2: chỉ nên dùng khi số không có giá trị lẻ*

- Thực hiện lần lượt từ trái sang phải:
  - **B1**: Nhân giá trị ký số tại vị trí đang xét với cơ số hiện tại để có giá trị X.
  - **B2**: Cộng X vào số kế tiếp bên phải.
  - **B3**: lặp lại B1 nếu vẫn còn số chưa tính

Ví dụ 1: $11001_2 = ?_{10}$



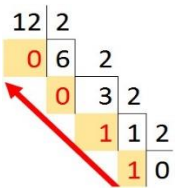
Ví dụ 2: $4706_8 = ?_{10}$



1.4.2. *Chuyển từ cơ số 10 sang cơ số khác (cơ số d)*

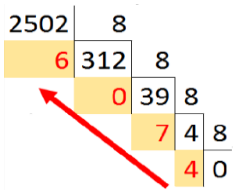
1.4.2.1. Chuyển phần nguyên

- Tổng quát:
  - Lần lượt chia cho cơ số d cho đến khi thương số bằng 0.
  - Kết quả là các dư số trong phép chia viết ra theo thứ tự ngược lại.
- Ví dụ:
  - Ví dụ 1: Số $12_{10} = ?_2$



Kết quả $12_{10} = 1100_2$

  - Ví dụ 2: Số $2502_{10} = ?_8$



Kết quả $2502_{10} = 4706_8$

Bài tập: $79023_{10} = ?_{16}$

1.4.2.2. Chuyển phần thập phân

- Tổng quát:
  - Lấy phần thập phân lần lượt nhân với cơ số cần đổi cho đến khi phần thập phân của tích số bằng 0.
  - Kết quả là các số phần nguyên trong phép nhân viết ra theo thứ tự tính toán.
- Ví dụ:
  - **Ví dụ 1:** $0.6875_{10} = ?_2$

```
0.6875 * 2 = 1.375
0.375  * 2 = 0.75
0.75   * 2 = 1.5
0.5    * 2 = 1.0
```

Kết quả: $0.6875_{10} = 0.1011_2$

  - **Ví dụ 2:** $0.6875_{10} = ?_8$

```
0.6875 * 8 = 5.500
0.5    * 8 = 4.00
```

Kết quả: $0.6875_{10} = 0.54_8$

- Bài tập:
  - $456.375_{10} = ?_8$
  - $1.023_{10} = ?_{16}$

1.4.3. *Chuyển từ cơ số Octal hoặc Hexa sang Binary*

- Tổng quát:
  - *Bước 1*: Chuyển mỗi ký số trong số hệ Octal (hoặc Hexadecimal) thành 3 (hoặc 4) số nhị phân tương ứng.
  - *Bước 2*: Kết nối tất cả các nhóm nhị phân (mỗi nhóm có 3 hoặc 4 số) thành một số nhị phân.
- Ví dụ
  - Ví dụ 1: $762_8 = ?_2$
    Kết quả: $762_8 = 111110010_2$

	7	6	2
	111	110	010
Kết quả:	111 110 010		

  - Ví dụ 2: $B6E_{16} = ?_2$
    Kết quả: $B6E_{16} = 101001101110_2$

	B	6	E
	1010	0110	1110
Kết quả:	1010 0110 1110		

- Bài tập:
  - $6751_8 = ?_2$
  - $ABC_{16} = ?_2$

1.4.4. *Chuyển đổi giữa 2 cơ số Octal và HexaDecimal*

- *Tổng quát*:
  - *Bước 1*: Chuyển số cần đổi sang hệ nhị phân.
  - *Bước 2*: Nhóm các số nhị phân vừa có (từ phải sang trái) theo số lượng bit của cơ số cần chuyển đổi đến.
  - *Bước 3*: chuyển giá trị các nhóm về giá trị của cơ số cần chuyển đổi đến
- *Ví dụ*
  - Ví dụ 1: $762_8 = ?_{16}$
    Kết quả $762_8 = 1F2_{16}$

7	6	2
111	110	010
111110010		
1 1111 0010		
1	F	2

  - Ví dụ 2: $E9A_{16} = ?_8$

    Kết quả $E9A_{16} = 7232_8$

E	9	A
1110	1001	1010
111010011010		
111 010 011 010		
7	2	3	2

1.5. CÁC PHÉP TÍNH TRONG HỆ NHỊ PHÂN

1.5.1. *Phép Cộng*

- *Quy ước*

```
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 (nhớ 1 lên cột bên
           trái của cột đang tính)
```

- *Ví dụ*

```
    1   1   1   1   1        (nhớ)
        0   1   1   0   1
+
        1   0   1   1   1
  -------------------------
    1   0   0   1   0   0
```

1.5.2. *Phép Trừ*

- *Quy ước*

```
0 - 0 = 0
0 - 1 = 1 (mượn)
1 - 0 = 1
1 - 1 = 0
```

- *Ví dụ*

```
        1       1            (mượn)
    1   1   0   1   0   1
-
            1   0   1   1
  -------------------------
    1   0   0   0   0   0
```

1.5.3. *Phép Nhân*

- *Quy ước*

```
0 * 0 = 0
0 * 1 = 0
1 * 0 = 0
1 * 1 = 1
```

- *Ví dụ*

```
                1   0   1   1
            X   1   0   1   0
  -------------------------
                0   0   0   0
            1   0   1   1
        0   0   0   0
    1   0   1   1
  -------------------------
    1   1   0   1   1   1   0
```

1.5.4. *Phép Chia*

- *Quy ước*: thực hiện như đối với số thập phân
- *Ví dụ*: thực hiện phép chia $11011_2(=27_{10})$ với 101 $(=5_{10})$

```
    1  1  0  1  1 | 1  0  1
-   1  0  1        | 1  0  1
    0  0  1  1  1
       -  1  0  1
          0  1  0
```

1.5.5. *Dịch trái/phải*

- Dịch trái (*Shift Left*) ≈ Nhân đôi giá trị

0	0	1	1	0	1	1	1	$(=55_{10})$

Shift left lần 1 (<<1) ⇩

0	1	1	0	1	1	1	0	$(=110_{10})$

Shift left lần 2 (<<1) ⇩

1	1	0	1	1	1	0	0	$(=220_{10})$

Shift left lần 3 (<<1) ⇩

1	1	0	1	1	1	0	0	0	$(=184_{10})$

(tràn mất số 1)

- Dịch phải (*Shift Right*) ≈ Chia đôi giá trị (chỉ lấy phần nguyên)

0	0	1	1	0	1	1	1	$(=55_{10})$

Shift right lần 1 (>>1) ⇩

0	0	0	1	1	0	1	1	$(=27_{10})$

Shift right lần 2 (>>1) ⇩

0	0	0	0	1	1	0	1	$(=13_{10})$

Shift right lần 3 (>>1) ⇩

0	0	0	0	0	1	1	0	$(=6_{10})$

1.6. Toán tử logic

A	B	A and B	A or B	A xor B	not A
0	0	0	0	1	1
1	0	0	1	0	0
0	1	0	1	0	1
1	1	1	1	1	0

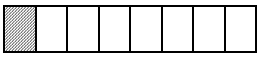
1.7. Biểu diễn số âm trong máy tính

1.7.1. *Giới thiệu*

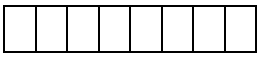
- Có nhiều phương pháp được sử dụng để biểu diễn số âm trong máy tính như:
  - Phương pháp dấu lượng (*sign-and-magnitude*)
  - Phương pháp bù 1(*one's complement*)
  - Phương pháp bù 2 (*two's complement*)
  - Phương pháp số quá *N* (*excess-N*).
- Các máy tính hiện nay hầu hết đều sử dụng phương pháp biểu diễn số bù 2. Tuy nhiên, trong vài tình huống, các phương pháp khác vẫn có thể được sử dụng.

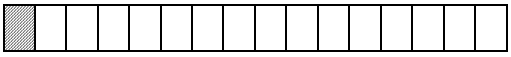
1.7.2. *Phương pháp dấu lượng (sign-and-magnitude)*

- Dùng bit cực trái làm bit dấu (*sign bit*), theo quy ước:
  - Nếu bit dấu là 1 thì số là số âm (≈ dấu "-")
  - Nếu bit dấu là 0 thì số là số dương (≈ với "+").
  - Các bit còn lại được dùng để biểu diễn độ lớn của số.
- Theo phương pháp này, một byte (8 bit) sẽ có 7 bit (trừ đi bit dấu) được dùng để biểu diễn cho các số có giá trị từ 0000000 (0_{10}) đến 1111111 (127_{10}). Khi sử dụng bit dấu, ý nghĩa của 7 bit trên sẽ thay đổi, và ta có thể biểu diễn các số từ -128_{10} đến $+127_{10}$.
- Trong phương pháp *dấu lượng*, số 0 có thể được biểu diễn ở hai dạng, đó là 00000000 (+0) và 10000000 (−0).
- Liên hệ đến ngôn ngữ lập trình C: khai báo kiểu dữ liệu
  - Kiểu char: (1 byte)
    - ***char***:
      VD char x; // x ∈ [-128,127]

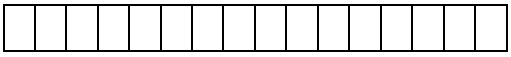
      

      ⇧ sử dụng bit bên trái làm dấu
    - ***unsigned char***:
      VD unsigned char y; // y∈[0,255]

      
  - Kiểu int: (2 bytes)
    - ***int***:
      VD int x; // x ∈[-32768,32767]

      ⇧ sử dụng bit bên trái làm dấu
    - ***unsigned int***:
      VD unsigned y; // y∈[0,65535]

      
  - Cách thực hiện tương tự đối với kiểu dữ liệu số nguyên 4 bytes (long, unsigned long).

1.7.3. *Phương pháp số bù 1 (one's complement)*

- Phương pháp *bù 1* biểu diễn số âm theo cách sau:
  - Bit dấu là 0 nếu số là số dương, và 1 nếu số là số âm.
  - Sử dụng toán tử NOT để đảo tất cả các bit của số nhị phân dương (dĩ nhiên không tính bit dấu) để biểu diễn số âm tương ứng.

    ⇒ phương pháp *bù 1* hoàn toàn giống như phương pháp *dấu lượng*, duy chỉ khác ở cách biểu diễn độ lớn của số.
- Ví dụ: biểu diễn số −5 (sử dụng 8 bit) trong máy tính theo phương pháp bù 1:
  - Biểu diễn số 5 trong máy tính: 0000 0101

- Đảo tất cả các bit có trong số 5: 1111 1010

  Vì là biểu diễn số âm nên bit bên trái cùng luôn giữ là 1.

- Giống phương pháp *Dấu Lượng*, một byte (8 bit) áp dụng phương pháp *bù 1* cũng có thể biểu diễn các số từ -127_{10} đến $+127_{10}$ (đã dùng 1 bit làm bit dấu).
- Phương pháp *Bù 1* cũng có hai dạng biểu diễn cho số 0, bao gồm: 00000000 (+0) và 11111111 (−0) (mẫu 8 bit).

1.7.4. *Phương pháp số bù 2 (two's complement)*

- Phương pháp *số bù 2* ra đời khi người ta gặp vấn đề với hai phương pháp *Dấu Lượng* và *bù 1* là:
  - Có hai cách biểu diễn cho số 0.
  - Bit nhớ phát sinh sau khi đã thực hiện phép tính phải được cộng tiếp vào kết quả.
- Trong phương pháp *Bù 2*, các số âm được biểu diễn giống như phương pháp *bù 1*, tuy nhiên, phải cộng thêm 1 vào kết quả (ở hệ nhị phân).
- Ví dụ: sử dụng 8 bite để biểu diễn số -5_{10}:

Biểu diễn nhị phân của số +5	0	0	0	0	0	1	0	1
Bù 1	**1**	**1**	**1**	**1**	**1**	0	**1**	0
Bù 2	1	1	1	1	1	0	1	**1**

- Với phương pháp *bù 2*, số 0 chỉ có một cách biểu diễn duy nhất là 00000000 (mẫu 8 bit).
- Việc đổi dấu một số – kể cả từ âm sang dương hay từ dương sang âm – đều được thực hiện theo cùng một cách, là: đảo tất cả các bit rồi cộng thêm một vào kết quả. Việc thực hiện phép cộng với số biểu diễn theo phương pháp *bù 2* được thực hiện hoàn toàn giống như cộng hai số nhị phân bình thường.
- Với mẫu 8 bit, phương pháp *bù 2* có thể biểu diễn tốt các số nguyên có giá trị từ -128_{10} đến $+127_{10}$ (so với từ -127_{10} đến $+127_{10}$ theo phương pháp *dấu lượng* và *bù 1*, do tiết kiệm được một cách biểu diễn số 0 (không phân biệt giữa −0 và +0).

1.7.5. *Phương pháp số quá N (excess-N)*

- Phương pháp biểu diễn số quá *N* – còn được gọi là biểu diễn số dịch (*biased representation*) – sử dụng một số nguyên *N* cho trước làm giá trị dịch ("dịch" hiểu nôm na theo nghĩa "sự dịch chuyển" hay "sự thiên lệch"). Theo phương pháp này, một giá trị thập phân (tức giá trị cần biểu diễn) sẽ được biểu diễn bằng dạng nhị phân của một số dương nào đó sao cho, giá trị của số dương này lớn hơn giá trị cần biểu diễn *N* đơn vị.
- Ví dụ: giả sử cần biểu diễn giá trị 2_{10} theo số quá 5 (mẫu 8 bit):
  - Bước 1: ta có:
    - Giá trị cần biểu diễn: 2.
    - N = 5.
  - Bước 2: xác định số dương lớn hơn 2_{10} năm đơn vị, đó là số 7.

    Vậy 2_{10} sẽ được biểu diễn bằng dạng nhị phân của 7: 00000111.

Theo ví dụ trên, ta sẽ có bảng sau:

Số thập phân cần biểu diễn	Giá trị thập phân của số quá 5	Số thập phân sẽ được biểu diễn
−5	0	00000000
−4	1	00000001
−3	2	00000010
−2	3	00000011
−1	4	00000100
0	5	00000101
1	6	00000110
2	7	00000111
3	8	00001000
4	9	00001001
5	10	00001010
6	11	00001011
7	12	00001100
8	13	00001101
9	14	00001110
10	15	00001111

- Ta thấy, 0 được biểu diễn bằng nhị phân của 5, và −5 được biểu diễn bằng nhị phân của 0. Tổng quát, 0 được biểu diễn bằng nhị phân của N, còn −N được biểu diễn bằng mẫu có tất cả các bit đều là 0.
- PP này được sử dụng rộng rãi để biểu diễn các số chấm động (floating point number), tiêu biểu là chuẩn số chấm động IEEE. Theo chuẩn này, các số chấm động có độ chính xác đơn (single-precision) 32 bit (như kiểu float của Java) có phần mũ (chính là số lượng ký số của phần nằm sau dấu chấm thập phân) được biểu diễn bằng số quá 127 với mẫu 8 bit, và các số chấm động có độ chính xác đôi (double-precision) 64 bit (như kiểu double của Java) có phần mũ biểu diễn bằng số quá 1023 với mẫu 11 bit.

GIẢI THUẬT

2.1. MỘT SỐ KHÁI NIỆM

2.1.1. *Thuật toán (Algorithm)*

Là một dãy các bước chặt chẽ và rõ ràng, xác định một trình tự các thao tác trên một số đối tượng nào đó sao cho một số hữu hạn lần thực hiện ta thu được kết quả như mong đợi.

2.1.2. *Các đặc trưng của thuật toán*

- Tính dừng (tính kết thúc): Một thuật toán bao giờ cũng phải dừng sau một số hữu hạn các bước thực hiện.
- Tính phổ dụng (tính chính xác): thuật toán có thể giải bất kỳ bài toán nào trong một lớp các bài toán.
- Tính duy nhất: chạy chương trình nhiều lần trên cùng một tập dữ liệu đầu vào phải cho ra cùng một kết quả.
- Tính rõ ràng: giải thuật phải được thể hiện bằng các câu lệnh minh bạch; các câu lệnh được sắp xếp theo thứ tự nhất định.
- Tính khách quan: Một giải thuật dù được viết bởi nhiều người trên nhiều máy tính vẫn phải cho kết quả như nhau.

2.2. GIẢI THUẬT

- Trong thực tiễn, người ta chấp nhận các cách giải thường cho kết quả tốt nhưng ít phức tạp, hiệu quả và khả thi. Do đó, khái niệm thuật toán được mở rộng bằng một khái niệm mới là giải thuật. Giải thuật là những cách giải không hoàn toàn đáp ứng đầy đủ các tính chất của một thuật toán nhưng vẫn cho kết quả gần đúng.

  Để thuận tiện, trong tài liệu này sẽ sử dụng khái niệm giải thuật để chỉ chung cho thuật toán và giải thuật.
- Tóm lại, *Giải thuật* là một tập hợp hữu hạn của các chỉ thị hay phương cách được định nghĩa rõ ràng cho việc hoàn tất một số sự việc từ một trạng thái ban đầu cho trước; khi các chỉ thị này được áp dụng triệt để thì sẽ dẫn đến kết quả sau cùng như đã dự đoán.
- Mô tả thuật toán: có thể mô tả thuật toán bởi một trong ba cách:
  - Mã tự nhiên
  - Mã giả (*pseudocode*).
  - Dùng các biểu tượng được quy định để biểu diễn giải thuật (*flowchart*).

2.2.1. *Mã tự nhiên*

2.2.1.1. Khái niệm

Là viết nội dung chương trình bằng ngôn ngữ tự nhiên giản lược.

2.2.1.2. Quy ước

Phải dễ hiểu, dễ nhớ, nhất quán và không có sự hiểu lầm.

2.2.1.3. Ví dụ 1.1

Viết chương trình cho người dùng nhập một số nguyên dương (n). Tính tổng các số từ 1 đến n.

B1: Nhập số n
B2: Kiểm tra giá trị của n
- Nếu n<=0: quay lại B1
- Ngược lại (n>0) chuyển sang B3

B3: Gán giá trị cho biến Tổng=0 và SoDangXet=1
B4: Thực hiện khi SoDangXet <=n
- Cộng dồn SoDangXet vào biến Tổng
- Tăng SoDangXet lên 1 đơn vị.

Quay lại đầu B4
B5: Xuất giá trị của Tổng

2.2.2. *Mã giả (pseudocode)*

2.2.2.1. Khái niệm

Dùng ngôn ngữ đã được quy ước trước để diễn đạt thuật toán.

2.2.2.2. Quy ước

- Phải dễ hiểu, không cần mô tả chi tiết đến các kỹ thuật dùng trong lập trình.
- Có thể dùng các từ khóa của ngôn ngữ đang dùng, hoặc có thể sử dụng một số từ khóa đã được quy ước trước như:
  - **IF** <Điều kiện> **THEN** … **ENDIF**
  - **IF** <Điều kiện> **THEN. .. ELSE. .. ENDIF**
  - **WHILE** <Điều kiện> **DO** … **ENDWHILE**
  - **DO** … **UNTIL** <Điều kiện>
  - **WRITE** …
  - **READ** …
  - **RETURN** …

2.2.2.3. Ví dụ 1.2

Viết chương trình cho người dùng nhập 1 số nguyên dương (n). Tính tổng các số từ 1 đến n.

Input: ∅
Process:

```
DO
    READ n
UNTIL (n>0)
Tong=0
SoDangXet=1
WHILE (SoDangXet <=n) DO
    Tong = Tong + SoDangXet
    SoDangXet = SoDangXet +1.
ENDWHILE
```

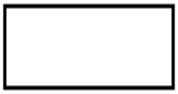
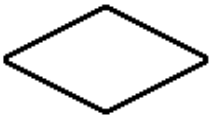
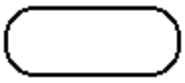
Output: *Tong*

2.2.3. *Lưu đồ*

2.2.3.1. Khái niệm

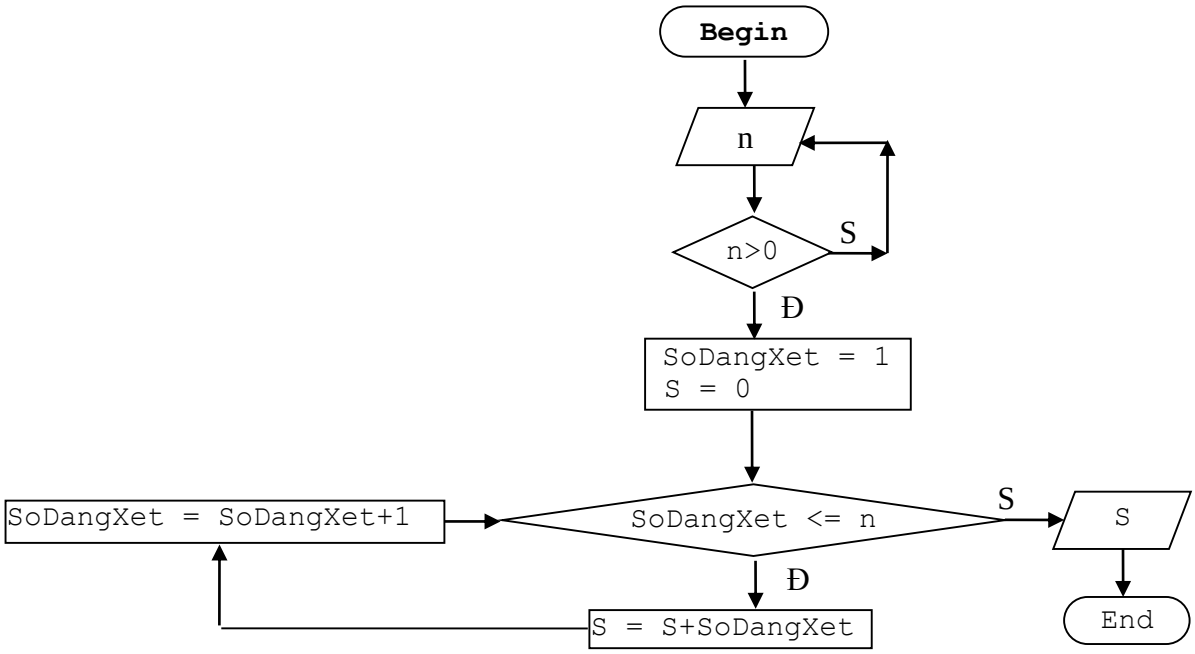
Còn được gọi là sơ đồ khối. Là sơ đồ thể hiện các bước của giải thuật liên quan đến một vấn đề nào đó được đưa vào giải quyết bằng máy tính.

2.2.3.2. Các ký hiệu dùng trong lưu đồ

KÝ HIỆU		Ý NGHĨA
	nhập	Nhập dữ liệu
	Xuất	Xuất dữ liệu
	xử lý	Nhóm lệnh để thực hiện một chức năng nào đó của chương trình (như gán giá trị cho biến, tính toán, …)
	Gọi hàm	Gọi hàm đã định nghĩa.
	Kiểm tra điều kiện	Quyết định rẽ nhánh căn cứ vào điều kiện chỉ định được ghi trong khối.
	Kiểm tra điều kiện	Quyết định rẽ nhánh căn cứ vào điều kiện chỉ định được ghi trong khối. Dùng cho trường hợp có nhiều đường ra của lệnh switch
		Hướng xử lý của lưu đồ.
	điểm đầu/cuối	Điểm đầu hay điểm cuối của lưu đồ.

2.2.3.3. Ví dụ 1.3

Viết chương trình cho người dùng nhập 1 số nguyên dương (n). Tính tổng các số từ 1 đến n.



2.2.4. *Một số ví dụ minh họa*

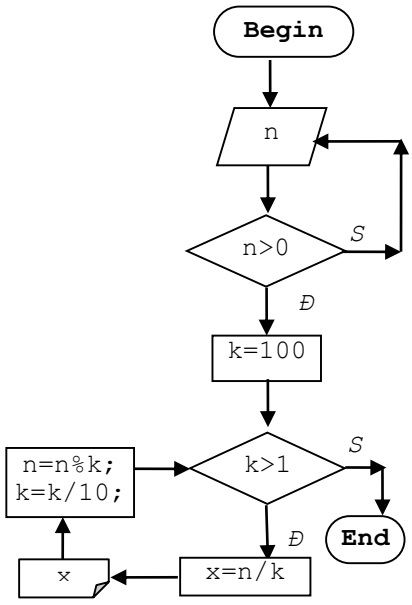
2.2.4.1. Ví dụ 1.4

Thực hiện cả 3 cách mô tả giải thuật cho ví dụ sau: Viết chương trình cho người dùng nhập 1 số nguyên dương (n) có giá trị trong khoảng từ 100 đến 999. In lần lượt từng số hạng (từ trái sang phải) có trong n ra màn hình. Vd, với n=123, sẽ lần lượt in ra từng số theo thứ tự từ trái sang phải là 1 2 3.

(i). Mã tự nhiên	*(ii). Mã giả*

B1: Nhập số *n*
B2: Kiểm tra giá trị của *n*
• Nếu *n*<=0: quay lại B1
• Ngược lại (*n*>0) chuyển sang B3
B3: Gán giá trị cho biến *i*=1; k=100; S=0
B4: Thực hiện khi *k* >1
• Gán x= n/k
• Xuất x
• Loại bỏ số bên trái của n vừa xét (n=n%k)
• Giảm k đi 10 lần (k=k/10)
• Quay lại đầu B4
	INPUT: ∅
	PROCESS:
	DO
	READ *n*
	UNTIL (n>0)
	i=1; k=100; S=0;
	WHILE (*k*>1) DO
	x= n/k
	WRITE x
	n=n%k
	k=k/10
	ENDWHILE
	OUTPUT: đã được lồng trong quá trình xử lý

(iii). Lưu đồ



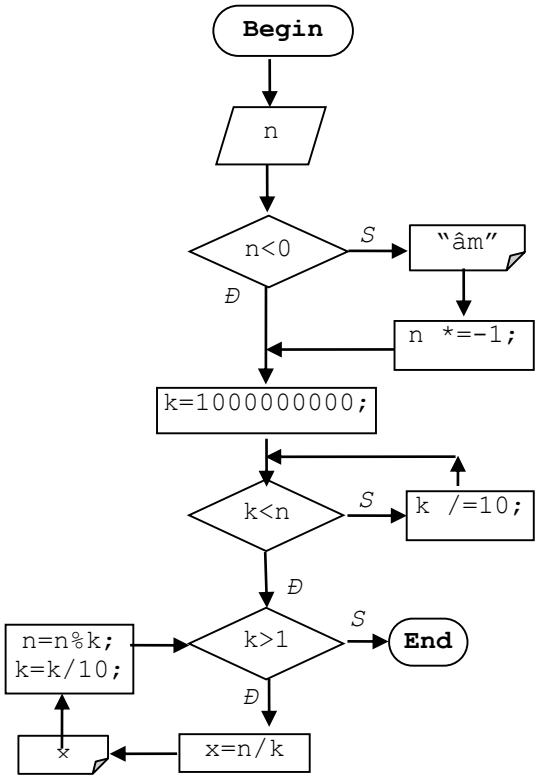
2.2.4.2. Ví dụ 1.5

Mở rộng ví dụ 1.4 để người dùng có thể nhập số nguyên dương (n) có giá trị trong khoảng ± 1 tỷ.

(i). Mã tự nhiên	*(ii). Mã giả*

B1: Nhập số *n*
B2: Kiểm tra giá trị của *n*
• Nếu *n*<0:
▫ In chuỗi "âm" ra màn hình
▫ Gán n=n*-1
• Sang B3
B3: Gán biến k=1000000000;
B4: (Giảm giá trị của k cho phù hợp với giá trị của n)
Trong khi (k>n)
k=k/10
B5: Thực hiện khi *k* >1
• Gán x= n/k
• Xuất x
• Loại bỏ số bên trái của n vừa xét (n=n%k)
• Giảm k đi 10 lần (k=k/10)
• Quay lại đầu B5
	INPUT: ∅
	PROCESS:
	DO
	READ *n*
	UNTIL (n>0)
	IF (n<0) THEN
	WRITE "âm";
	n*=-1;
	ENDIF
	k=1000000000;
	WHILE (*k*>n) DO
	k/=10;
	ENDWHILE
	WHILE (*k*>1) DO
	x= n/k
	WRITE x
	n=n%k
	k=k/10
	ENDWHILE
	OUTPUT: đã được lồng trong quá trình xử lý

(iii). Lưu đồ



2.3. BÀI TẬP

Thực hiện cả 3 cách mô tả giải thuật cho các bài tập sau. Biến n (nếu có) sẽ do người dùng nhập khi chương trình thực hiện:

1.3.1. Cho nhập số n, số bắt đầu a và công sai d (tất cả là số nguyên dương). In lên màn hình n số hạng đầu tiên của cấp số cộng.

1.3.2. Cho nhập số n, số bắt đầu a và công bội q (tất cả là số nguyên dương). In lên màn hình n số hạng đầu tiên của cấp số nhân.

1.3.3. Nhập số nguyên dương n (n>0). Tìm X sao cho tổng các số từ 1 đến X <=n. Ví dụ: n=16, in ra màn hình 1 + 2 + 3 + 4 + 5 <= 16

1.3.4. Tính S(n)= $\frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \ldots + \frac{n}{n+1}$

1.3.5. Tính S(n)= $\frac{1}{2} + \frac{3}{4} + \frac{5}{6} + \ldots + \frac{2n+1}{2n+2}$

1.3.6. Tính S(n)= $1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \ldots + \frac{1}{1+2+3+\ldots.+n}$

1.3.7. Viết chương trình In ra tất cả các số lẻ nhỏ hơn 100 trừ các số 5,7,93.

1.3.8. Cho hai số nguyên dương a và b. Tìm ước số chung lớn nhất và bội số chung nhỏ nhất của a, b.

1.3.9. Cho nhập số nguyên dương n. Kiểm tra xem n có phải là số nguyên tố hay không?

1.3.10. Cho nhập số nguyên dương n. Liệt kê các số nguyên tố < n.

1.3.11. Cho nhập số nguyên dương n. Đếm các số nguyên tố <n.

1.3.12. Cho nhập số nguyên dương n Tìm số nguyên tố đầu tiên <n và có giá trị gần với n nhất.

1.3.13. Cho nhập số nguyên dương n. Tìm số nguyên tố đầu tiên >n và có giá trị gần với n nhất.

1.3.14. Cho nhập số nguyên dương n, liệt kê các ước số của n là số nguyên tố.
Ví dụ: Nhập n=36. Các ước số của 36 gồm 1, 2, 3, 4, 6, 9, 12, 18
Nhưng chỉ in ra: các số vừa là ước số của 36, vừa là số nguyên tố: 2∪ 3.

NGÔN NGỮ C

3.1. CÁC LỆNH CƠ BẢN TRONG C

3.1.1. *Từ khóa riêng của ngôn ngữ C*

- Gồm: *break, char, continue, case, do, double, default, else, float, for, goto, int, if, long, return, struct, switch, unsigned, while, typedef, union, void, ...*
- Trong chương trình C tên biến, tên hằng, tên hàm, … (do người dùng tự đặt) không được trùng với từ khóa riêng của ngôn ngữ C.

3.1.2. *Cấu trúc đơn giản của chương trình C*

```
// Các chỉ thị tiền xử lý
#include <tên tập tin.h>
// Các khai báo hằng số (nếu có)
#define MAX 100
// Khai báo biến toàn cục (nếu có, thường ít được dùng)
// Hàm chính của chương trình
void main()
{ //các lệnh của chương trình
}
```

3.1.2.1. Các chỉ thị tiền xử lý

3.1.2.1.1. Chỉ thị #include <tên tập tin. h>

- Dùng để chèn nội dung của một file khác vào chương trình nguồn tại đúng vị trí mà chỉ thị ***include*** xuất hiện. Khi lập trình trên môi trường. NET, một số tập tin loại này đã được đưa vào thư viện của. NET nên không có phần. ***h*** đi sau. Ví dụ: `#include <iostream>`
- Cách sử dụng:
  - *Khai báo dạng 1*: cho phép C ngầm định tìm tập tin. ***h*** tại thư mục định sẵn (khai báo thông qua menu *Options\Directories*) thường là các tệp nguyên mẫu của thư viện C. Ví dụ: `#include <stidio.h>`
  - *Khai báo dạng 2*: cho phép tìm tập tin. ***h*** theo đường dẫn được chỉ định. Các tập tin. h dạng này thường được tạo bởi lập trình viên và được đặt trong cùng thư mục chứa chương trình. Dạng này cho phép lập trình viên chia một chương trình thành nhiều module đặt trên một số tệp khác nhau để dễ quản lý. Ví dụ: `#include <C:\mylib.h>`

3.1.2.1.2. Chỉ thị #define

- Dùng để định nghĩa các tên hằng số (constant) và các macro có phạm vi sử dụng trong toàn chương trình hoặc cho đến khi được định nghĩa lại sau chỉ thị `#undef`.
- Trước khi dịch bộ tiền xử lý sẽ tìm trong chương trình và thay thế bất kỳ vị trí xuất hiện nào của tên_macro bằng giá trị đã được khai báo trong `#define`.
- ví dụ:

```
#define MAX 100   // thay MAX bằng 100
#define TRUE 1    // thay TRUE bằng 1
#define PI 3.1415     // thay thế PI bằng 3.1415
```

3.1.2.1.3. Chỉ thị #if <dãy lệnh ……> #endif

Các chỉ thị này giống như câu lệnh if, mục đích là báo cho chương trình dịch biết đoạn lệnh giữa #if (điều kiện) và #endif chỉ được dịch nếu điều kiện đúng.

3.1.2.1.4. Chỉ thị #ifdef và #ifndef

Chỉ thị này báo cho chương trình dịch biết đoạn lệnh có được dịch hay không khi một tên gọi đã được định nghĩa hay chưa. #ifdef được hiểu là nếu tên đã được định nghĩa thì dịch, còn #ifndef được hiểu là nếu tên chưa được định nghĩa thì dịch.

3.1.2.1.5. Hàm main()

- Các dạng của hàm main:

Dạng 1:
```
void main()
{ /* Các khai báo*/
  /* Các lệnh*/
}
```

Dạng 2:
```
Kiểu_trả_về main(void)
{ /* Các khai báo*/
  /* Các lệnh*/
  return
  Giá_trị_trả_về;
}
```

Dạng 3:
```
Kiểu_trả_về main(int i,
                 char *s[])
{ /* Các khai báo*/
  /* Các lệnh*/
  return
  Giá_trị_trả_về;
}
```

- Ví dụ:

Dạng 1:
```
void   main()
{ printf ("Hello");
}
```

Dạng 2:
```
int main()
{ printf ("Hello");
  return 0;
}
```

Dạng 3:
```
int main(int k, char *s[])
{ printf ("Hello");
  return 0;
}
```

3.1.3. *Biến và khai báo biến*

3.1.3.1. Khai báo biến

- Phải khai báo biến trước khi sử dụng.
- Tên biến gồm chữ cái, ký số, dấu gạch dưới (_) và phải bắt đầu bằng một ký tự.
- Không được trùng với các từ khoá của ngôn ngữ C/C++
- Tên biến có phân biệt ký tự hoa và thường
- Tên biến nên dễ hiểu, súc tích và gợi nhớ.
- Biến khai báo trong một khối được gọi là biến cục bộ, biến không thuộc khối nào gọi là biến toàn cục.
- Biến có tác dụng trong toàn khối kể từ lúc được khai báo.

3.1.3.2. Cú pháp

```
<data type>   variable_name1, variable_name2;
```

Ví dụ:
```
int    sum;
long int  datenum=5;
foat  a,b,c;
```

3.1.4. *Xuất/ nhập dữ liệu*

3.1.4.1. Cú pháp

```
printf ("chuỗi định dạng"[, đối mục 1, đối mục 2,…]);
```

3.1.4.2. Chuỗi định dạng

- Định dạng dữ liệu cần xuất

Ký tự	*Ý nghĩa*
%c	Ký tự đơn
%d	Số nguyên thập phân có dấu
%e	Số chấm động (ký hiệu có số mũ)
%f, %g	Số chấm động (ký hiệu thập phân)

%i	Số nguyên
%o	Số nguyên bát phân (Octal)
%p	Con trỏ (pointer)
%s	Chuỗi
%u	Số nguyên không dấu
%x	Số nguyên thập lục (Hexa)
%%	Xuất ra ký hiệu phần trăm (%)

- Các ký tự điều khiển và ký tự đặc biệt

Ký tự	*Y nghĩa*	
\a	Alert (bell)	Tiếng kêu bip
\b	Backspace	Tương tự như phím Backspace
\f	Next page	
\n	Newline	Xuống dòng và chuyển con nháy về đầu dòng
\r	Carriage return	Nhảy về đầu dòng (không xuống dòng)
\t	Horizontal tab	Canh cột tab ngang
\'	Single quotation mark	In ra dấu '
\"	Double quotation mark	In ra dấu "
\\	Backslash	In ra dấu \
\0	Null character	
%%	percent	In ra dấu %

3.1.5. *Nhập dữ liệu*

3.1.5.1. Cú pháp

***scanf** ("chuỗi định dạng"[, đối mục 1, đối mục 2,…]);*

3.1.5.2. Giải thích

- Sử dụng chuỗi định dạng tương tự khi xuất dữ liệu.

3.1.6. *Các kiểu dữ liệu*

3.1.6.1. Số nguyên

Tên kiểu	*K/Thước (byte)*	*Miền giá trị*	*Nhỏ nhất*	*Lớn nhất*
char	1	–128 to 127	CHAR_MIN	CHAR_MAX
unsigned char	1	0 đến 255	0	UCHAR_MAX
short	2	–32,768 đến 32,767	SHRT_MIN	SHRT_MAX
unsigned short	2	0 đến 65535	0	USHRT_MAX
int	2	–32,768 đến 32,767	INT_MIN	INT_MAX
unsigned int	2	0 đến 65535	0	UINT_MAX
long	4	–2,147,483,648 đến 2,147,483,647	LONG_MIN	LONG_MAX
unsigned long	4	0 đến 4,394,967,295	0	ULONG_MAX

3.1.6.2. Số thực

Tên kiểu	*Kích thước (byte)*	*Miền giá trị*	*Nhỏ nhất*	*Lớn nhất*
float	4	3.4E +/- 38 (7 digits)	FLT_MIN	FLT_MAX
double	8	1.7E +/- 308 (15 digits)	DBL_MIN	DBL_MAX
long double	10	1.2E +/- 4932 (19 digits)	LDBL_MIN	LDBL_MAX

3.1.6.3. Ký tự

- Kiểu ký tự bao gồm 256 ký tự trong đó bao gồm các chữ cái (chữ thường và chữ hoa), chữ số, các dấu, một số các ký hiệu.
- Phân loại kiểu dữ liệu ký tự trong C:

Tên kiểu	*Bytes*	*Tên gọi khác*	*Miền giá trị*
`char`	1	signed char	–128 to 127
`unsigned char`	1	Không có	0 to 255

- Để trình bày một hằng ký tự, ta đặt vào dấu nháy đơn: 'a', 'D' , …

3.1.6.4. Phép biến đổi kiểu (cast operator)

Cú pháp: `datatype(expression)`

Trong đó, `datatype` là kiểu dữ liệu mà ta muốn áp đặt cho cho biểu thức.

3.1.7. *Toán tử trong C*

3.1.7.1. Toán tử số học

- Mỗi phép toán số học sẽ kết hợp 2 toán hạng.

Phép toán	*Ý nghĩa*	*Ghi chú*
✱	Phép nhân	
/	Phép chia	Tùy thuộc vào kiểu dữ liệu
%	Phép chia lấy phần dư	Chỉ áp dụng cho hai số nguyên
+	Phép cộng	
-	Phép trừ	

- Xác định kiểu dữ liệu của kết quả trong phép toán số học:
  - Nếu cả hai toán hạng là số nguyên thì kết quả thuộc kiểu nguyên.
  - Nếu có một toán hạng nào là kiểu số thực thì kết quả thuộc kiểu thực.
- Sự kết hợp và độ ưu tiên của các toán tử:
  - Phần biểu thức được đặt trong ngoặc sẽ được ưu tiên tính toán trước.
  - Có thể có nhiều cặp dấu ngoặc được sử dụng lồng vào nhau, khi đó biểu thức đặt ở ngoặc trong cùng có ưu tiên cao nhất.
  - Độ ưu tiên của các phép toán theo thứ tự liệt kê ở bảng trên. Nếu có 2 phép toán giống nhau trong cùng biểu thức thì thứ tự xét độ ưu tiên là từ trái sang phải.

3.1.7.1.1. Các toán tử quan hệ (relational operators)

- Các biểu thức sử dụng toán tử quan hệ gọi là biểu thức quan hệ (relation expression).
- Các toán tử quan hệ trong C:

Toán tử	*Công dụng*	*Ví dụ*
<	So sánh nhỏ hơn	*tuoi<30*
>	So sánh lớn hơn	*chieucao >1.7*
<=	So sánh nhỏ hơn hoặc bằng	*trongluong<=80*
>=	So sánh nhỏ hơn hoặc bằng	*soluong>=100*
==	So sánh bằng	*loai=='a'*
!=	So sánh khác	*loai!='b'*

- Một biểu thức quan hệ (điều kiện) sẽ có một trong hai kết quả đúng hoặc sai. Nếu đúng thì biểu thức có giá trị 1, ngược lại có giá trị 0.

3.1.7.1.2. Toán tử logic

- Các toán tử logic:
  - `AND (&&)`
  - `OR (||)`
  - `NOT (!)`

- Khi cần tạo ra các điều kiện phức hợp chúng ta sẽ sử dụng kết hợp giữa các toán tử quan hệ và các toán tử logic. Khi đó các biểu thức quan hệ đơn giản (như +, -, ✶, /,.. .) nên đặt trong cặp dấu ngoặc đơn ( ).
- Giá trị của biểu thức ghép được cho trong bảng sau. Trong đó `true`=1 và `false`=0:

A	B	A && B	A || B	! A
False	*False*	*False*	*False*	True
True	*False*	*False*	True	*False*
False	True	*False*	True	True
True	True	True	True	*False*

3.1.7.2. Toán tử một ngôi

Giúp tăng hoặc giảm giá trị của biến đếm đi 1 đơn vị.

i++(hay ++i) hoặc i-- hay(--i)

3.1.7.3. Toán tử sizeof

Dùng để xác định kích thước của một loại dữ liệu hoặc một biến.

Ví dụ:

```
printf( "kich thuoc kieu int la: %d", sizeof(int);
```

3.1.7.4. Biểu thức chọn theo điều kiện

- Cú pháp

(điều kiện) ? BT1: BT2

- Biểu thức nhận giá trị BT1 nếu điều kiện khác 0 (ĐÚNG), các trường hợp khác nhận giá trị BT2
- Ví dụ: Có thể định nghĩa sẵn một macro để tìm số lớn:

```
#define max(x, y) ((x>y) ? x: y)
```

3.1.7.5. Toán tử trên bit

- Toán tử trên bit chỉ có tác dụng trên các kiểu số nguyên.

Tên toán tử	*Ý nghĩa*	*Ghi chú*	
&	And	1 & 1 = 1	0 & 1 = 0
		1 & 0 = 0	0 & 0 = 0
|	Or	1 | 1 =1	0 | 1 =1
		1 | 0 = 1	0 | 0 = 0
^	XOr	1 ^ 1 = 0	0 ^ 1 = 1
		1 ^ 0 = 1	0 ^ 0 = 0
>>	Shift phải	A >> n = A/(2^n)	
<<	Shift trái	A << n = A*(2^n)	
~	Lấy phần bù	~1 = 0	~0 = 1

3.1.7.6. Độ ưu tiên của các phép toán

Toán tử	*Độ ưu tiên*	*Trình tự kết hợp*
() [] ->	1	Từ trái qua phải
! ~ ++ -- - + * & sizeof	2	Từ phải qua trái
* / %	3	Từ trái qua phải
+ -	4	Từ trái qua phải
<< >>	5	Từ trái qua phải
< <= >= >	6	Từ trái qua phải
== !=	7	Từ trái qua phải
&	8	Từ trái qua phải
|	9	Từ trái qua phải

^	10	Từ trái qua phải
&&	11	Từ trái qua phải
||	12	Từ trái qua phải
?:	13	Từ phải qua trái
= += -= *= /= %= ...	14	Từ phải qua trái

3.1.8. *Lệnh gán dữ liệu cho một biến*

3.1.8.1. Cú pháp

```
Tên_biến= biểu thức;
```

3.1.8.2. Lưu ý

- Biểu thức có thể là một hằng, một biến hoặc một biểu thức
- Trong một biểu thức gán có thể có nhiều toán tử gán, khi đó biểu thức sẽ được thực hiện từ phải qua trái. Ví dụ `a=b=c=5;`

3.1.9. *Hằng (const)*

- Sử dụng khi không cho phép dữ liệu thay đổi trong chương trình.
- Cú pháp: `const <data type> <name_const> = <value>;`
- Ví dụ: `const float PI=3.1416;`

3.2. CẤU TRÚC ĐIỀU KHIỂN

3.2.1. *Cấu trúc `if – else`*

3.2.1.1. Công dụng

Chỉ cho máy tính chọn thực hiện một trong hai dãy lệnh dựa vào kết quả của một Biểu thức Điều Kiện (btđk).

3.2.1.2. Cú pháp

```
if (btđk)
{
    Nhóm Lệnh 1;
}
else
{
    Nhóm Lệnh 2;
}
```

3.2.1.3. Cấu trúc if không else

- Đây là dạng biến đổi của cấu trúc trên, ở dạng này không có phần else.
- Cú pháp:

```
if (btđk)
{
    Nhóm Lệnh
}
```

Giải thích: khi `btđk` có giá trị khác 0 thì lệnh sẽ được thực hiện. Ngược lại sẽ không làm gì và thực hiện các lệnh sau đó (nếu có).

3.2.1.4. Cấu trúc if lồng nhau

Cho phép trong mỗi phần của if hoặc else có thể chứa một cấu trúc if – else khác. Toàn bộ một cấu trúc if – else lồng nhau được xem như một câu lệnh đơn.

3.2.2. *Cấu trúc switch*

3.2.2.1. Công dụng

Khi sử dụng cấu trúc *if – else* để lần lượt so sánh giá trị của một biến/biểu thức với nhiều giá trị, ta nên sử dụng cấu trúc ***switch.***

3.2.2.2. Cú pháp

```
switch  (biểu thức)
{
    case  giá_trị_1:
          nhóm lệnh 1;
          break;
    case   giá_trị _2:
          nhóm lệnh 2;
          break;
    ...
    case   giá_trị_n:
          nhóm lệnh n;
          break;
    default:
          nhóm lệnh default;
}
```

Giải thích:

- *Biểu thức*: có thể là biến hoặc biểu thức.
- *case*: Nếu kết quả của biểu thức sau *switch* bằng với giá trị đi sau *case* nào thì thực hiện các lệnh thuộc về *case* đó (bắt đầu từ dòng lệnh ngay dưới *case* cho đến khi gặp lệnh *break*).
- *Default:* Nếu như giá trị của biểu thức không bằng một giá trị nào trong các giá trị có trong cấu trúc *switch* thì các lệnh thuộc *default* sẽ được thực hiện.:
- *Lệnh break*: có công dụng thoát khỏi *switch* sau khi đã thực hiện một nhóm lệnh thuộc về một nhãn nào đó. Nếu không có lệnh này thì sau khi thực hiện xong các lệnh cần thiết nó sẽ thực hiện tiếp các lệnh của các trường hợp bên dưới.

3.2.2.3. Ví dụ

Chương trình nhập vào một số nguyên (0<=so<=9), sau đó in lên màn hình tên gọi của số này:

```
#include <stdio.h>
void main()
{ int so;
  printf("Nhap   so   nguyen(0-
  9):");
  scanf("%d", &so);
  switch (so)
  {
    case 0: printf("Khong");
            break;
    case 1: printf("Mot");
            break;
    case 2: printf("Hai");
            break;
    case 3: printf("Ba");
            break;
    case 4: printf("Bon");
            break;
    case 5: printf("Nam");
            break;
    case 6: printf("Sau");
            break;
    case 7: printf("Bay");
            break;
    case 8: printf("Tam");
            break;
    case 9: printf("Chin");
            break;
    default:
        printf("Nhap sai");
  }//switch
}//main
```

3.2.3. *Các cấu trúc lặp*

3.2.3.1. Cấu trúc while

3.2.3.1.1. Công dụng:

Cấu trúc while thường được dùng khi cần kiểm tra một số điều kiện trước khi cho nhóm lệnh thực hiện.

3.2.3.1.2. Cú pháp

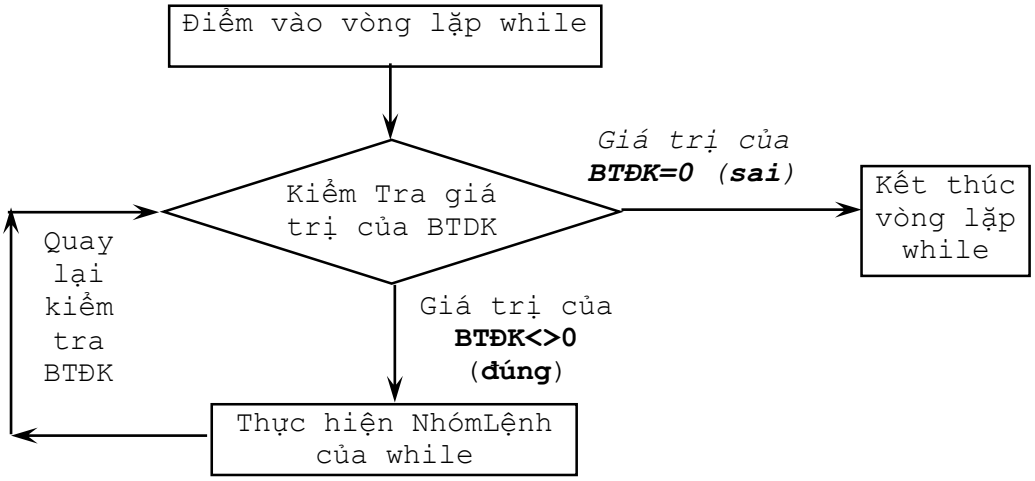
```
while (btđk)
{
    NhómLệnh;
}
```

3.2.3.1.3. Giải thích

- ***btđk*** phải được đặt trong cặp ngoặc đơn.
- *NhómLệnh* bên dưới ***btđk*** sẽ được thực hiện lặp đi lặp lại khi mà ***btđk*** vẫn có giá trị khác không (điều kiện còn đúng). Như vậy tất nhiên ở đâu đó trong đoạn lệnh của `while` phải có một lệnh làm thay đổi giá trị của ***btđk***.
- Câu lệnh bên dưới ***btđk*** phải là một lệnh duy nhất hoặc là một lệnh ghép (nhiều lệnh đặt vào trong {})

3.2.3.1.4. Hoạt động của cấu trúc while

- Kiểm tra giá trị của ***btđk***:
  - Nếu ***btđk*** có giá trị khác không.
    - Thực hiện lệnh bên dưới.
    - Quay lại kiểm tra ***btđk***.
  - Ngược lại: kết thúc cấu trúc while.
- Hình minh họa tiến trình hoạt động của ***while***.



3.2.3.2. Cấu trúc for

3.2.3.2.1. Công dụng

Cấu trúc for thường được dùng trong các trường hợp số lần lặp của vòng lặp được biết trước (giá trị cố định).

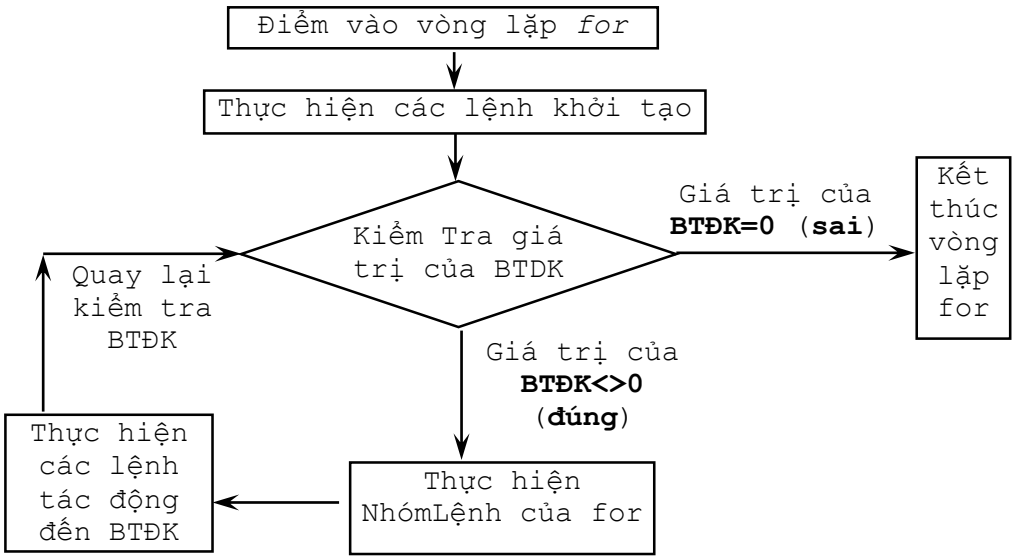
3.2.3.2.2. Cú pháp

```
for (lệnh khởi tạo; btđk; lệnh tác động đến điều kiện lặp)
{   NhómLệnh;
```

}

Các thành phần trong ngoặc của for đều có thể vắng mặt tuy nhiên phải để đầy đủ các dấu chấm phẩy (;).

3.2.3.2.3. Hoạt động của cấu trúc for



3.2.3.3. Cấu trúc do-while

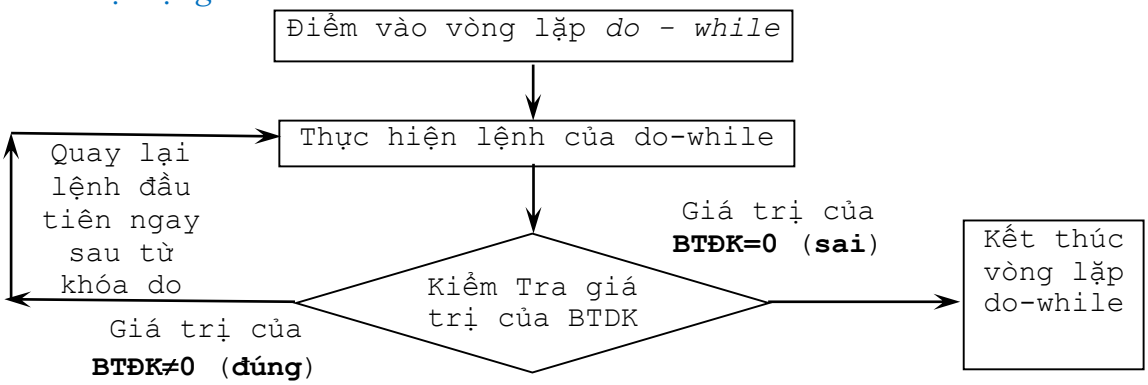
3.2.3.3.1. Công dụng

Cấu trúc do-while thường được dùng khi cho thực hiện trước một số lệnh, sau đó mới thực hiện kiểm tra một số điều kiện.

3.2.3.3.2. Cú pháp

```
do
{
    NhómLệnh;
}while (btđk);
```

3.2.3.3.3. Hoạt động của do – while



3.2.3.4. Các cấu trúc lồng nhau

- Toàn bộ một cấu trúc while, for, do-while được xem như một câu lệnh vì vậy ta có thể lồng ghép các cấu trúc này vào nhau.
- Ví dụ: in các bảng cửu chương 2,3,4 lên màn hình
  - Cách 1

```
void main()
{ int i, bcc;
```

```
        for (i=1;i<=9;i++)
        {   for (bcc =2; bcc <=4; bcc ++)
                printf("%3d * %2d = %3d", bcc, i, i*bcc);
            printf("\n");
        }
    }
```

- Cách 2:

```
void main()
{ int i=1,j;
  while (i<=10)
  {   for (j=2;j<=4;j++)
          printf("%3d * %2d = %3d", bcc, i, i*bcc);
      printf("\n");
      i++;
  }
}
```

3.2.3.5. Lệnh break và continue

3.2.3.5.1. break

Công dụng: Thoát khỏi các cấu trúc switch, while, for, do-while tại thời điểm break được gọi thi hành mà không cần kiểm tra kết quả của BTĐK.

Khi có nhiều vòng lặp lồng nhau, câu lệnh break sẽ thoát khỏi vòng lặp chứa nó bên trong khối lệnh lặp.

3.2.3.5.2. continue

Công dụng: Khi lệnh continue được gọi thì chương trình sẽ quay trở về đầu vòng lặp để bắt đầu lần lặp mới. Nếu có các lệnh còn lại (cùng trong vòng lặp) đặt sau continue sẽ không được thực hiện.

3.3. MỘT SỐ THƯ VIỆN THƯỜNG DÙNG TRONG C

- Hàm thư viện là những hàm đã được định nghĩa sẵn trong một thư viện nào đó, muốn sử dụng các hàm thư viện thì phải khai báo thư viện trước khi sử dụng bằng lệnh `#include <tên thư viện.h>`
- Ý nghĩa của một số thư viện thường dùng:

(i) **`stdio.h`**: Thư viện chứa các hàm vào/ ra chuẩn (standard input/output). Gồm các hàm `printf()`, `scanf()`, `getc()`, `putc()`, `gets()`, `puts()`, `fflush()`, `fopen()`, `fclose()`, `fread()`, `fwrite()`, `getchar()`, `putchar()`, `getw()`, `putw()`,...

(ii) **`conio.h`**: Thư viện chứa các hàm vào ra trong chế độ DOS (DOS console). Gồm các hàm `clrscr()`, `getch()`, `getche()`, `getpass()`, `cgets()`, `cputs()`, `putch()`, `clreol()`,...

(iii) **`math.h`**: Thư viện chứa các hàm tính toán gồm các hàm `abs()`, `sqrt()`, `log()`. `log10()`, `sin()`, `cos()`, `tan()`, `acos()`, `asin()`, `atan()`, `pow()`, `exp()`,...

(iv) **`alloc.h`**: Thư viện chứa các hàm liên quan đến việc quản lý bộ nhơ. Gồm các hàm `calloc()`, `realloc()`, `malloc()`, `free()`, `farmalloc()`, `farcalloc()`, `farfree()`,...

(v) **io.h**: Thư viện chứa các hàm vào ra cấp thấp. Gồm các hàm `open(), _open(), read(), _read(), close(), _close(), creat(), _creat(), creatnew(), eof(), filelength(), lock(),...`

(vi) **graphics.h**: Thư viện chứa các hàm liên quan đến đồ họa. Gồm `initgraph(), line(), circle(), putpixel(), getpixel(), setcolor(),...`

3.4. BÀI TẬP

3.4.1. *Phần cơ bản*

3.4.1.1. Cho biết các định danh sau đây có hợp lệ hay không?

a) Đối với các định danh hợp lệ, định danh nào có tên gợi nhớ (Tên gợi nhớ là một tên mà bản thân nó phần nào nói lên mục đích sử dụng).

b) Đối với các định danh không hợp lệ, cho biết tại sao không hợp lệ.

So_lan	do	234m1	NewDate	x007
dem_so_nguyen_to	Do	char	AB12	a9b8c7e6
Temp	dO	@b	gia tri	Thanh-tien

3.4.1.2. Đặt tên và xác định kiểu dữ liệu cần dùng của các biến trong các trường hợp thực hiện các công việc sau:

a) Tìm giá trị lớn nhất trong một tập hợp các số.
b) Xác định xem một số có phải là số nguyên tố hay không.
c) Lưu họ và tên của học sinh.
d) Xác định xem một số có phải là là số lẻ hay không
e) Đếm số lượng các số chia chẵn cho 3 trong một tập hợp các số.
f) Điểm trung bình của ba môn học.
g) Số ngày trong một tháng.
h) Khoảng cách từ điểm A đến điểm B.
i) Chiều dài của một cây cầu.

3.4.1.3. Viết lại các số dưới dạng số thập phân chuẩn.

a) 6.35e5 b) 1.51926e2 c) 2.95e-3 d) 4.623e-6 e) 1.001e-5

3.4.1.4. Viết lại các số thập phân dưới đây dưới dạng số mũ.

a) 123 b) 789.456 c) 0.0312 d) 0.1357 e) 0.000008

3.4.1.5. Viết lại các biểu thức đại số sau đây thành biểu thức thích hợp trong C:

a) (4)(5)-(7)(3) *b)* (2+1.96)(4.5-2.1) *c)* 5.7(2+15.96)(-2) *d)* $3^{y} \times 4^{m}$

e) $\frac{mn}{m^2+n^2}$ f) $\frac{m-n}{\frac{6x-y}{mx+2y}}$ g) $\frac{8(k+1)^2}{k^3}$ h) $\frac{mx^2+nx+r}{ex^4+fx^2+g}$

3.4.1.6. Cho x=4.5, y=6.2, z=-1.8. Tính giá trị của các biểu thức sau:

a) int(x) *b)* int(x)+y+z *c)* float(int(x))+y *d)* int(x)+float(fabs(z))
e) int(y) *f)* int(x+y)+z *g)* float(int(x+y)) *h)* (fabs(x)*int(z))/int(y)
i) int(x+y) *j)* int(x+y+z) *k)* sqrt(fabs(x)+fabs(z)) *l)* float(0*x)+(z*x)

3.4.1.7. Cho x=4, y=2, z=-3, w=4. Tính giá trị của các biểu thức sau:

a) x>y *b)* x!=y *c)* w%y==z%y *d)* x*y!=z*w
e) x/y==w/z *f)* x*y *g)* x%y*z *h)* x%y==y%x

3.4.1.8. Viết lại các biểu thức bằng cách thêm các dấu ngoặc để cho biết thứ tự sẽ thực hiện các phép toán. Sau đó hãy tính giá trị của các biểu thức với x=5, y=2 và z=4:

Ví dụ: x%y*z && z%y*x được viết lại là ((x%y)*z) && ((z%y)*x)

a) y%z*x || x%z/x

b) x+y%z*x && x%z==z/x
c) z%y/x*z && z%x || x==y || x=z%y

3.4.1.9. Viết các biểu thức quan hệ để biểu diễn các điều kiện sau (tên biến do học viên tự đặt):
a) Chiều cao của căn nhà phải lớn hơn 2m.
b) Thân nhiệt phải lớn hơn 37^{O}C.
c) Tuổi của một người trong khoảng từ 18 đến 25.
d) Điểm trung bình của học viên phải lớn hơn 5 và điểm môn toán không được nhỏ hơn 4.
e) Chiều dài phải nằm trong khoảng từ 5 đến 8 mét, chiều rộng phải nằm trong khoảng từ 2 đến 6 mét và diện tích không được vượt quá $45m^2$.

3.4.1.10. Xác định giá trị của các biểu thức sau, biết x=5, y=2, z=4 và w=5:
a) x == 5
b) y* z == z*z
c) w%y*z > 5 || z%y*w < 7
d) y == z/2 && x == w && 2*x*y >= z*w

3.4.1.11. C cho phép thực hiện phép cộng và trừ giữa số nguyên và ký tự. Điều này có thể thực hiện được do C luôn chuyển ký tự thành một số nguyên tương ứng (đó chính là mã ASCII của ký tự) khi ta sử dụng một ký tự trong một biểu thức số học. Dựa vào các thông tin trên, hãy xác định kết quả của các biểu thức sau:

a) char('c'-5) *b)* char('c'+5) *c)* char('C'-5) *d)* char('C'+5)
e) char('b'-'a') *f)* char('d'-('A'+1)) *g)* char('b'*3 -'B'*2) *h)* char('b'-(3*'E'))

3.4.1.12. Trình bày lại các chương trình sau cho rõ ràng hơn:

```
a) #include <stdio.h>
   int main(
   ){
   printf(
   "hello world"
   );return 0;}

b) void main
   (     ){ printf(    "Khu du lich Suoi Tien"); printf(
   "Thanh pho Ho Chi Minh \n"); printf(
   "VIET NAM"); printf
   ("chao don quy khach
   ");}

c) void main
   (     ){ printf   ( "Ha
   Noi"); printf(
   "\n
   VIET NAM"); ;}
```

```
d) void
   main(     ){ printf ("        *"); printf("\n"); printf("   *
   *"); printf("\n"); printf("  *             *"); printf("\n");
   printf( "* * * * *\n";}
```

3.4.1.13. Trình bày lại các chương trình dưới đây, sau đó cho biết kết quả thực hiện của các chương trình này:

```
a) #include <stdio.h>
   int main(
   ){  int x=5, y=7
   printf( "y/x =%f"
   ,y/x) ; printf("y*x= %d ", y*x); }

b) #include <stdio.h>
   int main(
   ){  int m, n, tam;
   printf("Nhap m:") ; scanf("%d", &m); printf("Nhap n:") ;
   scanf("%d", &m); tam=n; n=m; m=tam; printf( "Sau khi thuc hien
   chuong trinh, m=%d, n= %d", m, n); }

c) #include <stdio.h>
   int main(
   ){  int x=100, y=100;
   printf("x+y = %d", x+y); }
```

3.4.1.14. Tìm và sửa lỗi các chương trình sau:

```
a) #include <stdio.h>
   void main()
   {
     rong=10;
     printf("Dien tich hinh chu nhat là: %d" , dai * rong);
   }

b) #include <stdio.h>
   void main()
   { int dai, rong, dientich;
     dientich=dai*rong;
     rong=5;
     dai=12;
     printf("Dien tich hinh chu nhat là:", dientich);
   }

c) #include <stdio.h>
   void main()
   { int dai=14, rong=6, dientich;
     dai*rong=dientich;
     printf("Dien tich hinh chu nhat là: %d",dientich);
   }
```

3.4.1.15. Tìm các biểu thức hợp lệ, giải thích lý do:

a) `(x-y)=5`	*b)* `x-(y=5)`	*c)* `(x+y)-(m==n)`	*d)* `(m==n)*k`	*e)* `(x=y)==m`

3.4.1.16. Viết chương trình in ra màn hình dòng chữ :

"SAI GON - THANH PHO HO CHI MINH"

3.4.1.17. Viết chương trình in ra màn hình các hình sau:

a.-

```
          *
        * * *
      * * * * *
    * * * * * * *
```

b.-

```
* * * * *
* * * * *
* * * * *
* * * * *
```

c.-

```
          *
        *   *
      *       *
    * * * * * * *
```

d.-

```
* * * * *
*       *
*       *
* * * * *
```

3.4.1.18. Viết chương trình nhập vào tên của bạn, sau đó chương trình in ra dòng chữ: *Chao<tên bạn>*.

3.4.1.19. Cho a=10, b=6, hãy viết chương trình in ra màn hình tổng của a và b.

3.4.1.20. Cho hình chữ nhật có chiều dài là d=5.82, chiều rộng là r=9.15, hãy viết chương trình in ra màn hình diện tích và chu vi của hình chữ nhật trên.

3.4.1.21. Cho hằng số PI = 3.14 và bán kính r=6.25, viết chương trình tính chu vi và diện tích của hình tròn trên.

3.4.1.22. Viết chương trình tính chu vi và diện tích hình chữ nhật. Biết rằng chiều dài và chiều rộng là các số nguyên được nhập vào từ bàn phím.

3.4.1.23. Viết chương trình chuyển đổi từ độ Fahrenheit sang Celsius. Biết công thức chuyển đổi là C=5/9(F-32).

3.4.1.24. Viết chương trình nhập vào hai số nguyên, sau đó in ra tổng bình phương của chúng.

3.4.1.25. Viết chương trình tính diện tích và chu vi của hình chữ nhật với chiều rộng được nhập từ bàn phím. Biết rằng chiều dài= 1.5 chiều rộng.

3.4.1.26. Viết chương trình cho phép nhập số R. Sau đó tính rồi in ra chu vi, diện tích hình tròn có đường kính là R.

3.4.1.27. Viết chương trình cho người dùng nhập một số thực *R* vào rồi in ra:

a. Trị tuyệt đối của *R*.
b. Căn bậc hai của *R*.
c. Số đối của *R* (*VDụ: nhập 5.15 sẽ in ra -5.15; hoặc nhập vào -98.7 sẽ in ra +98.7*).

3.4.1.28. Viết chương trình cho phép nhập 2 số thực A và B. Sau đó thực hiện hoán vị giá trị của 2 số đó. Học viên lần lượt thực hiện theo 2 cách sau:

a. Cho phép dùng biến trung gian.
b. Không được dùng biến trung gian.

3.4.1.29. Nhập vào 2 số nguyên p, q và tính biểu thức sau:

$$(-q/2+(p^3/27+q^2/4)^{1/2})^{1/3} + (-q/2 - (p^3/27+q^2/4)^{1/2})^{1/3}$$

3.4.1.30. Giả sử có các loại tờ tiền 1đ, 2đ, 5đ,10đ và 20đ. Viết chương trình cho người dùng nhập vào số tiền, tìm số lượng mỗi loại tiền là bao nhiêu để có số tờ giấy bạc là ít nhất.

3.4.2. *Cấu trúc rẽ nhánh*

3.4.2.1. Nhập vào số nguyên n, kiểm tra xem n chẵn hay lẻ và xuất ra màn hình.

3.4.2.2. Nhập vào số nguyên n. Xuất ra n màn hình (Nếu n chẵn thì gấp đôi giá trị).

3.4.2.3. Nhập vào số nguyên n. Nếu n>5 thì tăng n lên 2 đơn vị và in ra giá trị n, ngược lại in ra giá trị 0.

3.4.2.4. Nhập vào 2 số x, y. Xuất ra màn hình tổng, hiệu, tích, thương của 2 số trên.

3.4.2.5. Nhập vào 2 số x, y. Xuất ra màn hình giá trị lớn nhất.

3.4.2.6. Viết chương trình nhập vào một số nguyên chỉ số đo độ của một góc và cho biết nó thuộc góc vuông thứ mấy của vòng tròn lượng giác.

3.4.2.7. Viết chương trình cho nhập một ký tự. Xác định ký tự vừa nhập có phải là chữ cái thường hay không? Nếu đúng in lên màn hình chữ cái hoa tương ứng, ngược lại in mã ASCII của ký tự vừa nhập.

3.4.2.8. Nhập vào hai số nguyên a, b. In ra màn hình giá trị lớn nhất.

3.4.2.9. Cho ba số a, b, c được nhập vào từ bàn phím. Hãy tìm giá trị lớn nhất của ba số trên và in ra kết quả.

3.4.2.10. Nhập vào 3 số nguyên dương a, b, c. Kiểm tra xem 3 số đó có lập thành 3 cạnh của một tam giác hay không? Nếu có hãy cho biết:

- Chu vi, diện tích của tam giác
- Chiều dài mỗi đường cao của tam giác.
- Tam giác đó thuộc loại nào? (vuông cân, cân, vuông, đều, …).

Nhắc lại:

- Công thức tính diện tích s = sqrt(p*(p-a)*(p-b)*(p-c))
- Công thức tính các đường cao: $h_a = 2s/a$, $h_b=2s/b$, $h_c=2s/c$.

(Với p là nữa chu vi của tam giác).

3.4.2.11. Nhập vào 6 số thực a, b, c, d, e, f . Giải hệ phương trình sau: $\begin{cases} ax+by=c \\ dx+ey=f \end{cases}$

Hướng dẫn:

$$D=\begin{vmatrix} a & b \\ d & e \end{vmatrix} = ae - bd$$

$$Dx=\begin{vmatrix} c & b \\ f & e \end{vmatrix} = ce - bf$$

$$Dy=\begin{vmatrix} a & c \\ d & f \end{vmatrix} = af - dc$$

Nếu D!=0 $\Rightarrow$ x=Dx/D; y=Dy/D

Ngược lại nếu Dx!=0 hoặc Dy !=0 $\Rightarrow$ PT vô nghiệm

Ngược lại, $\Rightarrow$ PT vô định

3.4.2.12. Giải và biện luận phương trình bậc 1 ax+b=0.

3.4.2.13. Giải và biện luận phương trình bậc 2: $ax^2+bx+c=0$

3.4.2.14. Giải và biện luận phương trình trùng phương: $ax^4 + bx^2 + c = 0$.

3.4.2.15. Cho nhập vào:

a. Hai (2) số. Sau đó hỏi người dùng muốn chọn phép tính nào trong 4 phép tính +, - , *, /. Xong ta cho in ra kết quả tương ứng. (*Hướng dẫn: dùng 1 biến kiểu char*).

b. Đường kính R của hình tròn. Sau đó hỏi người dùng muốn tính diện tích hay chu vi. Dựa vào chọn lựa của người dùng chương trình cho in ra kết quả tương ứng.

3.4.2.16. Viết một chương trình trắc nghiệm đơn giản.

a. Chương trình chỉ gồm 1 câu hỏi và 4 lựa chọn a, b, c, d (trong đó có một câu đúng).Sau khi người dùng chọn bằng cách nhấn 1 trong 4 phím (a, b, c, d). Chương trình thông báo kết quả đúng sai.

b. Mở rộng bài tập trên bằng cách tạo ra từ 3 đến 5 câu hỏi.

3.4.2.17. Viết chương trình nhập vào một số nguyên n gồm ba chữ số. Xuất ra màn hình chữ số lớn nhất?

Ví dụ: n=291. Chữ số lớn là 9.

3.4.2.18. Cho nhập điểm 3 môn toán, lý, hóa. Tính điểm trung bình khi hệ số của môn toán là 3, các môn còn lại hệ số 2. Dựa vào điểm trung bình (ĐTB) vừa tính được, in ra xếp loại, biết rằng :

- ĐTB>=9 : Xuất sắc
- 7<=ĐTB<8 : Khá
- 5<=ĐTB<6 : Trung bình
- 8<=ĐTB<9 : Giỏi
- 6<=ĐTB<7 : TB Khá
- ĐTB<5 : Yếu

3.4.2.19. Cho nhập số tiền gởi và thời hạn gởi tiền tiết kiệm (năm). Tính và in ra số tiền được nhận sau khi hết thời hạn gởi:

Thời gian gởi (năm)	Tỷ lệ (%)
Lớn hơn hay bằng 5 năm	0.95
4	0.90
3	0.85
2	0.80
1	0.75

3.4.2.20. Viết chương trình cho nhập số KM, sau đó tính và in ra số tiền cước TAXI phải trả.Biết rằng:

- KM đầu tiên là 5000^{đ}, mỗi 200m tiếp theo là 1000^{đ}.
- Nếu lớn hơn 30KM thì mỗi KM thêm ra là 3000^{đ}.
- Hãy nhập số KM sau đó in ra số tiền phải trả.

3.4.2.21. Viết chương trình tính tiền thuê máy dịch vụ Internet và in ra màn hình kết quả. Với dữ liệu nhập vào là giờ bắt đầu thuê (GBD), giờ kết thúc thuê (GKT), số máy thuê (SoMay).

- Điều kiện cho dữ liệu nhập: 6<=GBD<GKT<=21. Giờ là số nguyên.
- Đơn giá: 2500đ cho mỗi giờ máy trước 17 và 3000đ cho mỗi giờ máy sau 17.

3.4.2.22. Viết chương trình tính tiền lương ngày cho công nhân, cho biết trước giờ vào ca, giờ ra ca của mỗi người.

Giả sử rằng:

- Tiền trả cho mỗi giờ trước 12 giờ là 6000đ và sau 12 giờ là 7500đ.
- Giờ vào ca sớm nhất là 6 giờ sáng và giờ ra ca trễ nhất là 18 giờ *(Giả sử giờ nhập vào nguyên).*

3.4.2.23. Viết chương trình cho nhập vào tổng thu nhập và số người phụ thuộc của 1 người, tính và in ra màn hình số tiền thực lãnh sau khi trừ thuế. Biết rằng:

- Mỗi người phụ thuộc sẽ được giảm trừ 1.600.000đ/tháng
- Số thu nhập còn lại sau khi giảm trừ sẽ được tính thuế như sau:

Bậc chịu thuế	*Số tiền (triệu)*		*Tỷ lệ (%) chịu thuế*
	Từ	*Đến*	
1		4	0
2	>4	6	5
3	>6	9	10
4	>9	14	15
5	>14	24	20
6	>24	44	25
7	>44	84	30
8	>84		35

Ví dụ: tổng thu nhập 13.200.000đ/tháng, có 2 người phụ thuộc số tiền thuế được tính như sau:

Số tiền chịu thuế (T)=13.200.000-(2 ngườiX 1.600.000)=10.000.000đ

Số tiền thuế= (6.000.000đ X 5%)+(3.000.000 X 10%)+(1.000.000 X 15%)

= 300.000 đ + 300.000đ + 150.000 đ =750.000 đ

Thu nhập sau khi trừ thuế= 13.200.000 – 750.000đ = 12.450.000 đ

NGÀY, THÁNG, NĂM – GIỜ, PHÚT, GIÂY

3.4.2.24. Nhập vào giờ, phút, giây. Kiểm tra xem giờ, phút, giây đó có hợp lệ hay không? In kết quả ra màn hình.

3.4.2.25. Viết chương trình nhập vào hai giá trị chỉ giờ, phút, giây sau đó tính và in ra tổng của chúng.

3.4.2.26. Nhập vào ngày, tháng, năm. Kiểm tra xem ngày, tháng, năm đó có hợp lệ hay không? In kết quả ra màn hình.

3.4.2.27. Viết chương trình nhập vào ngày, tháng, năm hợp lệ. Cho biết năm này có phải là năm nhuận hay không? In kết quả ra màn hình.

3.4.2.28. Viết chương trình nhập một ngày (ngày, tháng, năm). Tìm ngày trước ngày vừa nhập (ngày, tháng, năm).

3.4.2.29. Viết chương trình nhập ngày, tháng, năm . Tính xem ngày đó là ngày thứ bao nhiêu trong năm.

3.4.3. *Cấu trúc lặp*

3.4.3.1. Bài tập căn bản

3.4.3.1.1. Cho đoạn chương trình sau:

```
void main()
{   int so=0;
    while (x<30)
    {   x++;                // (1)
        printf("%5d",x);    // (2)
    }
}
```

- Cho biết số nguyên đầu tiên và số nguyên cuối cùng được in ra màn hình.
- Nếu đảo ngược 2 phát biểu (1) và (2). Cho biết số nguyên đầu tiên và số nguyên cuối cùng được in ra màn hình.

3.4.3.1.2. Cho nhập chuỗi ký tự (S) và số lần cần in (n). In ra màn hình n lần chuỗi S.

3.4.3.1.3. Nhập số nguyên dương n (n>0). Liệt kê tất cả các số nhỏ hơn n (tính từ 1)

3.4.3.1.4. Nhập số nguyên dương n (n>0). Liệt kê tất cả các số ***lẻ*** nhỏ hơn n (tính từ 1)

3.4.3.1.5. Viết chương trình cho nhập nhiều, mỗi lần là một số nguyên. In ra tổng các số vừa nhập. Quá trình nhập kết thúc khi số nhập vào là số 0.

3.4.3.1.6. Viết chương trình nhập vào một số nguyên dương n. Tính tổng các số nguyên nhỏ hơn hoặc bằng n thỏa ít nhất một trong hai điều kiện:

- Cùng chia hết cho 3 và 5.
- Chia cho 3 thì dư 2 và chia cho 5 thì dư 3.

3.4.3.1.7. Nhập số nguyên dương n (n>0). Tìm X sao cho tổng các số từ 1 đến X <=n.

```
Ví dụ:  n=16, in ra màn hình 1 + 2 + 3 + 4 + 5 <= 16
```

3.4.3.1.8. Cho nhập số n, số bắt đầu a và công sai d (tất cả là số nguyên dương). In lên màn hình n số hạng đầu tiên của cấp số cộng.

3.4.3.1.9. Cho nhập số n, số bắt đầu a và công bội q (tất cả là số nguyên dương). In lên màn hình n số hạng đầu tiên của cấp số nhân.

3.4.3.2. Bảng cửu chương

3.4.3.2.1. Cho nhập 1 số nguyên dương n (n > 0). In ra bảng cửu chương của n.

3.4.3.2.2. In ra các bảng cửu chương từ 2 đến 9.

3.4.3.2.3. Cho nhập 2 số nguyên dương n và m (n < m). In ra lần lượt các bảng cửu chương từ n đến m.

3.4.3.3. Mã ASCII

3.4.3.3.1. In ra bảng mã ASCII theo dạng: số thứ tự mã : ký tự tương ứng . .

3.4.3.3.2. Tương tự câu a, nhưng in bảng mã trên nhiều dòng, trên mỗi dòng sẽ gồm 8 cặp sau : số thứ tự mã : ký tự tương ứng.

3.4.3.3.3. Viết chương trình cho nhập một ký tự. Xác định ký tự vừa nhập có phải là chữ cái thường hay không? Nếu đúng cho biết thêm mã ASCII và vị trí của ký tự này trong bảng 26 chữ cái.

3.4.3.3.4. Nhập vào 1 số nguyên n (0 <= n <= 255). In ra ký tự tương ứng của n trong bảng mã ASCII. Chương trình cho thực hiện nhiều lần và chỉ kết thúc khi n < 0 hoặc n > 255.

3.4.3.4. Kiểm tra số do người dùng nhập vào

3.4.3.4.1. Cho người dùng nhập vào 1 số nguyên dương ***n*** thỏa lần lượt các yêu cầu sau đây. Nếu người dùng nhập vào số không đúng yêu cầu thì chương trình cho nhập lại

a. Điều kiện: ***n*>0**. Nếu nhập đúng, chương trình in ra các số chẵn nhỏ hơn ***n*.**

b. Điều kiện: **0 <= *n* <= 9**. Nếu nhập đúng, chương trình in ra cách đọc của số vừa nhập.

c. Điều kiện: ***n* <10 hoặc *n* > 50**. Nếu nhập đúng, chương trình in ra ***n*** số ngẫu nhiên.

d. Điều kiện: ***n* bội số của 5**. Nếu nhập đúng, chương trình in ra bảng cửu chương 5.

e. Điều kiện: ***n* nằm trong các số (2, 4, 6, 8, 10)**. Nếu nhập đúng, chương trình cho nhập tiếp 2 số nguyên i, j. Hãy tính và in ra cấp số cộng với số hạng đầu là i, công sai là j và số phần tử của dãy là ***n* .**

3.4.3.4.2. Viết chương trình cho nhập 2 số nguyên dương ***n*** và ***m*** sao cho số nhập sau (***m***) phải luôn lớn hơn số nhập trước (***n***).

3.4.3.4.3. Cho nhập 2 số a, b sao cho: số lớn nhất trong 2 số phải là một số dương và chia hết cho 7. Nếu nhập sai phải yêu cầu nhập lại cho đến khi đúng.

3.4.3.4.4. Cho nhập 2 số nguyên dương a và b sao cho số nhập sau (b) thỏa 2 điều kiện : là ước số của a và b bội số của 5. Nếu nhập sai, cho nhập lại cả 2 số a và b.

3.4.3.4.5. Viết chương trình cho thực hiện nhiều lần các công việc sau. CT kết thúc khi số nhập vào là 0:

a. Nhập một số thực vào rồi in ra căn bậc hai của nó.

b. Nhập một số thực vào rồi in ra trị tuyệt đối của nó.

c. Nhập một số nguyên dương n, in ra n dấu sao (*).

d. Đọc một số nguyên n vào rồi in ra tổng các số từ 1 đến n. Yêu cầu dùng phát biểu WHILE để thực hiện.

e. Cho nhập một ký tự vào rồi in ra mã ASCII của nó.

3.4.3.4.6. Nhập vào số nguyên dương là lũy thừa của 2.

3.4.3.5. Tách 1 số nguyên gồm nhiều chữ số

3.4.3.5.1. Viết chương trình nhập vào một số nguyên (n) 3 chữ số (từ 100-999), sau đó in ra các chữ số thuộc hàng trăm, hàng chục, hàng đơn vị.

```
Ví dụ: n nhập vào là 128. Chương trình in ra:
                Số hàng trăm:  1
                Số hàng chục: 2
                Số hàng đơn vị: 8
```

3.4.3.5.2. In ra cách đọc của một số dương:

a. Nhập số nguyên dương n gồm 3 chữ số (n > 0), in ra cách đọc của số n.
Ví dụ: Nhập n = 105. In ra màn hình: Mot khong nam.

b. Tương tự như bài tập trên, nhưng cho phép giá trị của n nằm trong khoảng (0 <= n <= 4 tỷ). In ra cách đọc của n.

3.4.3.5.3. Cho nhập số nguyên dương n. Đếm số lượng chữ số chẵn, số chữ số lẻ có trong n.

3.4.3.5.4. Hãy tính tổng và tích các chữ số của số nguyên dương n.

3.4.3.5.5. Hãy tính tổng và tích của các chữ số chẵn, các chữ số lẻ của số nguyên dương n.

3.4.3.5.6. Hãy tìm chữ số lớn nhất của số nguyên dương n.

3.4.3.5.7. Hãy tìm chữ số nhỏ nhất của số nguyên dương n.

3.4.3.5.8. Hãy đếm số lượng chữ số lớn nhất của số nguyên dương n.

3.4.3.5.9. Hãy đếm số lượng chữ số nhỏ nhất của số nguyên dương n.

3.4.3.5.10. Hãy đếm số lượng chữ số của số nguyên dương n giống như chữ số đầu tiên của số nguyên dương n.

```
Ví dụ: Nhập n = 10151. In ra màn hình: Có ba chữ số 1.
```

3.4.3.5.11. Cho nhập số nguyên dương n. Kiểm tra xem các chữ số của n có toàn lẻ (hay toàn chẵn) không.

3.4.3.5.12. Cho nhập số nguyên dương n. Liệt kê các số <n có dạng 2^k.

3.4.3.5.13. Viết chương trình nhập số nguyên dương n gồm k chữ số (2<k<=5), kiểm tra xem các chữ số của n có phải là số đối xứng hay không?

```
Ví dụ:  Đối xứng:  12321.   Không đối xứng: 12345
```

3.4.3.5.14. Viết chương trình nhập vào số nguyên n gồm ba chữ số. Xuất ra màn hình theo thứ tự đảo ngược của các chữ số.

```
Ví dụ: n=291. Xuất ra 192.
```

3.4.3.5.15. Viết chương trình nhập vào số nguyên n gồm ba chữ số. Xuất ra màn hình theo thứ tự tăng dần của các chữ số.

```
Ví dụ: n=291. Xuất ra 129.
```

3.4.3.5.16. Hãy kiểm tra các chữ của số nguyên dương n có tăng dần từ trái sang phải hay không?

3.4.3.5.17. Hãy kiểm tra các chữ của số nguyên dương n có giảm dần từ trái sang phải hay không?

3.4.3.5.18. Cho ba số a, b, c được nhập vào từ bàn phím. Hãy in ra vị trí chứa số lớn nhất. Ví dụ : nhập 579, in ra : *« số lớn nhất ở hàng đơn vị »*

3.4.3.5.19. **(*)** Viết chương trình nhập số nguyên dương n gồm k chữ số $(0 < k \le 5)$, sắp xếp các chữ số của n theo thứ tự tăng dần.

```
Ví dụ: Nhập n=1536
       Kết  quả sau khi sắp xếp: 1356.
```

3.4.3.5.20. **(*)** Viết chương trình nhập số nguyên dương n gồm 5 chữ số, kiểm tra xem các chữ số n có phải là số đối xứng hay không.

```
Ví dụ:  Đối xứng:      13531
        Không đối xứng:13921
```

3.4.3.6. Ước/bội số chung

3.4.3.6.1. Cho nhập số nguyên dương n, liệt kê tất cả các ước số của n.

3.4.3.6.2. Cho nhập số nguyên dương n, đếm số lượng các ước số của n.

3.4.3.6.3. Cho nhập số nguyên dương n, tính tổng và tích các ước số của n.

3.4.3.6.4. Cho nhập số nguyên dương n, tính tổng các ước số chẵn của n.

3.4.3.6.5. Cho nhập số nguyên dương n, tìm ước số lẻ lớn nhất của n (nhỏ hơn n).

```
Ví dụ: Ước số lẻ lớn nhất của 27 là 9.
```

3.4.3.6.6. Cho nhập 2 số nguyên dương n và m, liệt kê các ước số của n có giá trị nhỏ hơn m.

```
Ví dụ: Nhập n=16,  m=7;
    in ra  các ước số của 16 nhỏ hơn 7 là 1, 2, 4.
```

3.4.3.6.7. Nhập vào số nguyên dương là lũy thừa của 2 (không cần viết chương trình kiểm tra; khi nhập vào, người dùng sẽ chủ động nhập 1 số sao cho số đó có thể phân tich thành dạng 2^k). In ra cách biểu diễn của n dưới dạng số mũ của 2.

```
Ví dụ : nhập n = 8 in ra ước số 8 = 2^3
```

3.4.3.6.8. Cho nhập số nguyên dương n (n>=2). Hãy phân tích n thành tích các thừa số nguyên tố.

```
Ví dụ : nhập n = 1350. Phân tích 1350 = 2*3*3*3*5*5
```

3.4.3.6.9. Cho nhập 2 số nguyên dương a, b. Tìm USCLN của a và b.

3.4.3.6.10. Viết chương trình cho người dùng nhập vào tử và mẫu số, thực hiện đơn giản phân số.

3.4.3.6.11. Cho nhập số nguyên dương n. In ra màn hình cách phân tích n thành thừa số nguyên tố.

```
Ví dụ : nhập n = 5000  in ra 5000 = 2^3  X  5^4.
```

3.4.3.6.12. Cho cho nhập 2 số nguyên dương a, b. Tìm BSCNN của a và b.

3.4.3.6.13. Cho nhập 2 số nguyên dương n và m, liệt kê các bội số của n có giá trị nhỏ hơn m.

```
Ví dụ: Nhập n=6, m=27;
    in ra  các bội số của 6 nhỏ hơn 27 là 12, 24
```

3.4.3.6.14. Tìm ước chung lớn nhất và bội chung nhỏ nhất của 2 số nguyên dương n và m nhập từ bàn phím. (Sử dụng thuật toán Euclide : USCLN(A,B) = USCLN(B, A mod B) với A>B)

3.4.3.6.15. Có 3 tuyến xe bus cùng khởi hành tại bến vào lúc 5h và chạy theo những tuyến đường (hướng đi) khác nhau. Mỗi lượt chạy, xe thứ 1 chạy và trở lại bến sau 1h05', nghỉ 10 phút rồi tiếp tục chạy lượt kế tiếp; xe thứ 2 chạy và trở lại bến sau 55', nghỉ 5 phút rồi tiếp tục chạy lượt kế tiếp; xe thứ 3 chạy và trở lại bến sau 48', nghỉ 2 phút rồi tiếp tục chạy lượt kế tiếp. Hỏi sau bao lâu nữa 3 xe sẽ lại cùng xuất phát.

Tổng quát, cho người dùng nhập thời gian chạy 1 lượt, thời gian nghỉ của mỗi loại xe. Cho biết sau bao lâu 3 xe sẽ lại cùng xuất phát.

3.4.3.7. Tổng/tích của 1 dãy số

Viết chương trình cho người dùng nhập số nguyên dương n (và k nếu có), rồi in ra trị của S (hoặc P). Biết rằng:

3.4.3.7.1. $S = 1 + 2 + 3 + \ldots + n$

3.4.3.7.2. $S = 1 + 3 + 5 + \ldots + (2n+1)$

3.4.3.7.3. $S = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$

3.4.3.7.4. $S= \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n+1}$

3.4.3.7.5. $S= \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \ldots + \frac{1}{2n}$

3.4.3.7.6. $S= 1 + \frac{1}{3} + \frac{1}{5} + \ldots + \frac{1}{2n-1}$

3.4.3.7.7. $S= \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \ldots + \frac{n}{n+1}$

3.4.3.7.8. $S= \frac{1}{2} + \frac{3}{4} + \frac{5}{6} + \ldots + \frac{2n+1}{2n+2}$

3.4.3.7.9. P = 1 x 2 x 3 x . . . x n

3.4.3.7.10. P = 1 x 3 x 5 x . . . x (2n-1)

3.4.3.7.11. P = k^n (không dùng hàm pow, chỉ dùng các lệnh lặp để tính)

3.4.3.7.12. S = (1) + (1+2)+(1+2+3)+(1+2+3+4)+ +(1+2+3+4+5)+ . . . +(1+2+3+... +n)

3.4.3.7.13. S = (1) + (1x2) +(1x2x3)+ . . . +(1x2x3x. . . xn)

3.4.3.7.14. $S= 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \ldots + \frac{1}{1+2+3+\ldots+n}$

3.4.3.7.15. $S = 1 - \frac{1}{1+2} + \frac{1}{1+2+3} - \ldots + (-1)^{n+1} \frac{1}{1+2+3+\ldots+n}$

3.4.3.7.16. $S= \frac{1}{1*2} + \frac{2}{2*3} + \frac{3}{3*4} + \ldots + \frac{n}{n*(n+1)}$

3.4.3.7.17. $S= 1 + \frac{1+2}{2} + \frac{1+2+3}{3} + \ldots + \frac{1+2+3+\ldots+n}{n}$

3.4.3.7.18. $S = \frac{1^2 + 2^2 + 3^2 + \ldots + n^2}{1*2 + 2*3 + 3*4 + \ldots + n(n+1)}$

3.4.3.7.19. $S = 1^2 + 2^2 + 3^2 + \ldots + n^2$

3.4.3.7.20. $S = x + x^2 + x^3 + \ldots + x^n$

3.4.3.7.21. $S = x^2 + x^4 + \ldots + x^{2n}$

3.4.3.7.22. $S = x + x^3 + x^5 + \ldots + x^{2n+1}$

3.4.3.7.23. $S= x + \frac{x^2}{1+2} + \frac{x^3}{1+2+3} + \ldots + \frac{x^n}{1+2+3+\ldots+n}$

3.4.3.7.24. $S = 1 - 2 + 3 - 4 + (-1)^{n+1}n$

3.4.3.7.25. $S = x - x^2 + x^3 - \ldots + (-1)^{n+1}x^n$

3.4.3.7.26. $S = -x^2 + x^4 - \ldots + (-1)^n x^{2n}$

3.4.3.7.27. $S = x - x^3 + x^5 + \ldots + (-1)^n x^{2n+1}$

3.4.3.7.28. $S = -x + \frac{x^2}{1+2} - \frac{x^3}{1+2+3} + \ldots + (-1)^n \frac{x^n}{1+2+3+\ldots+n}$

3.4.3.7.29. Lập chương trình nhập và tính một đa thức dạng:

$$P(x) = a_0x^n + a_1x^{n-1} + \ldots\ldots + a_{n-1}x + a_n.$$

3.4.3.7.30. Lập chương trình cho nhập x và y. Tính một đa thức dạng:

$$P = ((-x/2+y^3/27+x^2/4)^{1/2})^{1/3} + (-x/2-(y^3/27+x^2/4)^{1/2})^{1/3}$$

3.4.3.7.31. $S = \sqrt{2+\sqrt{2+\sqrt{2+.....\sqrt{2+\sqrt{2}}}}}$ có n dấu căn.

3.4.3.7.32. $S = \sqrt{n+\sqrt{n-1+\sqrt{n-2+.....\sqrt{2+\sqrt{1}}}}}$ có n dấu căn.

3.4.3.7.33. $S = \sqrt{1+\sqrt{2+\sqrt{3+.....\sqrt{n-1+\sqrt{n}}}}}$ có n dấu căn và $n > 0$

3.4.3.7.34. $S = \sqrt{n!+\sqrt{(n-1)!+\sqrt{(n-2)!+.....\sqrt{2!+\sqrt{1!}}}}}$ có n dấu căn.

3.4.3.7.35. $S = \sqrt[n]{n+\sqrt[n-1]{n-1+\sqrt[3]{3+\sqrt{2}}}}$ có n -1 dấu căn.

3.4.3.7.36. $S = \sqrt[n+1]{n+\sqrt[n]{n-1+\sqrt[3]{2+\sqrt{1}}}}$

3.4.3.7.37. $S = \sqrt[n+1]{n!+\sqrt[n]{(n-1)!+\sqrt[3]{2!+\sqrt{1!}}}}$

3.4.3.7.38. $S = \sqrt{x^n+\sqrt{x^{n-1}+\sqrt{x^{n-2}+.....\sqrt{x^2+\sqrt{x}}}}}$

3.4.3.7.39. $S = \cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{\ldots 1}}}}$

$$1+\frac{1}{1+1}$$

3.4.3.7.40. Cho n là số nguyên dương. Hãy tìm giá trị nguyên dương k lớn nhất sao cho S(k) < n. Trong đó chuỗi S(k) được định nghĩa như sau: S(k)=1+ 2+3+ . . . +k.

3.4.3.8. GIAI THỪA

3.4.3.8.1. Cho nhập một số nguyên dương (n). In ra giai thừa của n. Với giai thừa của số n được định nghĩa như sau : 0!=1

n!=1*2*3*...*n

3.4.3.8.2. Viết chương trình tính n!!=1.3.5…n nếu n lẻ, n!!=2.4.6…n nếu n chẵn.

3.4.3.8.3. Cho nhập một số nguyên K. Nếu:

- K >=0 thì in ra giai thừa của nó.
- K < 0 thì cho nhập lại .

3.4.3.8.4. Cho nhập số nguyên dương n, rồi in ra trị của S (hoặc e^x). Biết rằng:

a. S = 1! + 2! + 3! + . . . + n!

b. $S = 1/1! + ½! + 1/3! + \ldots + 1/n!$

c. $S = 1 + \frac{1+2}{2!} + \frac{1+2+3}{3!} + \ldots + \frac{1+2+3+\ldots+n}{n!}$

d. $S = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}$

e. $S = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots + \frac{x^{2n}}{(2n)!}$

f. $S = 1 + x + \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots + \frac{x^{2n+1}}{(2n+1)!}$

g. $S = -x + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots + (-1)^n \frac{x^n}{n!}$

h. $S = -1 + \frac{x^2}{2!} - \frac{x^4}{4!} + \ldots + (-1)^{n+1} \frac{x^{2n}}{(2n)!}$

i. $S = 1 - x + \frac{x^3}{3!} - \frac{x^5}{5!} + \ldots + (-1)^{n+1} \frac{x^{2n+1}}{(2n+1)!}$

3.4.3.8.5. Lập chương trình tính sin(x) với độ chính xác 0.0001 theo công thức:

$$\sin(x) = x - \frac{x3}{3!} + \frac{x5}{5!} - \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

3.4.3.8.6. Cho nhập ε (số thực), n (số nguyên). Tính e^x:

$$e^x = 1 + \frac{\varepsilon}{1!} + \frac{\varepsilon^x}{2!} + \ldots + \frac{\varepsilon^n}{n!}$$

với ε nhập từ bàn phím.

3.4.3.9. Số nguyên tố

3.4.3.9.1. Cho nhập số nguyên dương n. Kiểm tra xem n có phải là số nguyên tố hay không?

3.4.3.9.2. Cho nhập số nguyên dương n. Liệt kê các số nguyên tố < n.

3.4.3.9.3. Cho nhập số nguyên dương n. Đếm các số nguyên tố <n.

3.4.3.9.4. Cho nhập số nguyên dương n Tìm số nguyên tố đầu tiên <n và có giá trị gần với n nhất.

3.4.3.9.5. Cho nhập số nguyên dương n. Tìm số nguyên tố đầu tiên >n và có giá trị gần với n nhất.

3.4.3.9.6. Cho nhập số nguyên dương n, liệt kê các ước số của n là số nguyên tố.

```
Ví dụ: Nhập n=36. Các ước số của 36 gồm 1, 2, 3, 4, 6,
    9, 12, 18
Nhưng chỉ in ra:  các số vừa là ước số của 36, vừa là số
    nguyên tố:    2∪ 3
```

3.4.3.9.7. Nếu người dùng nhập vào số n>1 thì in n số nguyên tố đầu tiên (tính từ 2), sau đó cho chương trình lặp lại. Chương trình kết thúc khi n<=1.

3.4.3.9.8. Trung bình cộng các số nguyên tố > 10 và < n (3 <= n <= 100).

3.4.3.9.9. N bắt đầu bằng chữ số là số nguyên tố?

3.4.3.9.10. Tổng các chữ số có giá trị nguyên tố của N.

3.4.3.9.11. Cho nhập vào hai số n và m, in ra n số nguyên tố mà giá trị của chúng phải lớn hơn m.

3.4.3.9.12. Viết chương trình nhập số nguyên dương n gồm k chữ số $(0 < k \le 5)$, đếm xem n có bao nhiêu chữ số là số nguyên tố.

3.4.3.9.13. Viết chương trình chứng minh bài toán Gobach (một số nguyên tố bất kỳ lớn hơn 5 đều có thể khai triển thành tổng của 3 số nguyên tố khác. Minh họa cho những số nguyên tố<100.

3.4.3.10. CÁC BÀI TOÁN VỀ SỐ KHÁC

📖 *Số hoàn thiện*: là số (***n***) có tổng các ước số không kể n bằng chính n.

Ví dụ: 6 là số hoàn thiện vì 1+2+3=6.

📖 *Số chính phương*: là số có căn bậc hai là một số nguyên.

Ví dụ 9 là số chính phương vì căn bậc hai của 9 là 3.

📖 *Số Armstrong*: là số có đặc điểm sau: số đó gồm n ký số, tổng các lũy thừa bậc n của các ký số bằng chính số đó.

Ví dụ 153 là một số có 3 ký số, và $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$.

3.4.3.10.1. Cho số nguyên dương x. Kiểm tra xem x có phải là số hoàn thiện hay không?

3.4.3.10.2. Tìm các số hoàn thiện nhỏ hơn 5000.

3.4.3.10.3. Cho số nguyên dương x. In ra các số hoàn thiện nhỏ hơn x?

3.4.3.10.4. Cho số nguyên dương x. In ra tổng các số hoàn thiện nhỏ hơn x?

3.4.3.10.5. Cho số nguyên dương x. Kiểm tra xem x có phải là số chính phương hay không?

3.4.3.10.6. Nhập số nguyên dương n (n>0). Liệt kê n số chính phương đầu tiên.

3.4.3.10.7. Liệt kê tất cả các hoán vị của một tập gồm n phần tử.

3.4.3.10.8. Viết chương trình nhập số nguyên dương N (N <= 2 tỷ), kiểm tra xem N có phải là số đối xứng hay không. (Số đối xứng là số có giá trị không đổi nếu đọc các chữ số từ phải qua trái, ví dụ: 34543).

3.4.3.10.9. Viết chương trình nhập vào một số nguyên dương không dấu, rồi in ra số tương ứng thuộc các cơ số Bibanry, Octal, Hexa.

3.4.3.10.10. Viết chương trình nhập một số thập phân n. Tạo menu lựa chọn các công việc in giá trị tương ứng với n theo các cơ số: 2, 8, 16.

3.4.3.10.11. Viết chương trình nhập vào một số nguyên thuộc hệ m rồi in giá trị tương ứng ở hệ n.Trong đó m và n là hai giá trị nhập từ bàn phím (n, m là một trong các giá trị 2, 8, 10, 16).

3.4.3.10.12. Viết chương trình nhập vào 1 số nguyên dương n (hệ thập phân). In ra màn hình giá trị nhị phân tương ứng của số vừa nhập.

3.4.3.10.13. Tìm các số Armstrong nhỏ hơn 1000.

3.4.3.10.14. Dãy Fibonaci {Fn} bậc 2 được định nghĩa:

$f_0 = f_1 = 1$

$f_n = f_{n-1} + f_{n-2}$ (n>=2)

Viết chương trình nhập số nguyên dương n và in ra dãy Fibonaci n phần tử. Nhập số nguyên k < n, tính phần tử F_k

3.4.3.10.15. Tìm những giá trị nguyên x, y, z thỏa mãn công thức Pithagore $x^2+y^2=z^2$. Với 0<x, y, z <1000

3.4.3.10.16. Bài toán trăm trâu. Nội dung bài toán như sau:

Trăm trâu trăm cỏ
Trâu đứng ăn năm
Trâu nằm ăn ba
Trâu già ăn một

Viết chương trình tìm tất cả các trường hợp bài toán có nghiệm (vét cạn). Trong mỗi trường hợp, cho biết số lượng của mỗi loại trâu (đứng, nằm, già) có bao nhiêu con?

3.4.3.10.17. Bài toán gà chó. Nội dung bài toán như sau:

Vừa gà vừa chó
Bó lại cho tròn
Đúng ba sáu con
Một trăm chân chẵn.

Viết chương trình tìm tất cả các trường hợp bài toán có nghiệm (vét cạn). Trong mỗi trường hợp, cho biết số lượng gà, số lượng chó?

3.8.1.1. Viết chương trình minh họa bài toán An Casi. Biết bài toán An Casi được mô tả như sau: với mọi số n nguyên dương thì ta luôn có đẳng thức sau:

$1 + 2^4 + \ldots + n^4 = 6n^5 + 15n^4 + 10n^3 - n$ (đẳng thức Ansi)

3.8.1.2. Tính tổ hợp chập k của n.

$$C_n^k = \frac{n!}{k!(n-k)!}$$

3.8.1.3. Viết chương trình in tam giác Pascal.

C_0^0	1
$C_1^0\ C_1^1$	1 1
$C_2^0\ C_2^1\ C_2^2$	1 2 1
$C_3^0\ C_3^1\ C_3^2\ C_3^3$	1 3 3 1
.....	.....
$C_n^0\ C_n^1\ C_n^2 \ldots C_n^n$	1 9 36 84 126 126 84 36 9 1

Với $C_n^k = \frac{n!}{k!(n-k)!}$

3.8.1.4. ° Viết chương trình hiển thị tháp PASCAL :

```
                1
               121
              12321
             1234321
            123454321
           12345654321
          1234567654321
         123456787654321
        12345678987654321
```

° Tương tự, nhưng viết lại chương trình hiển thị tháp đảo ngược.

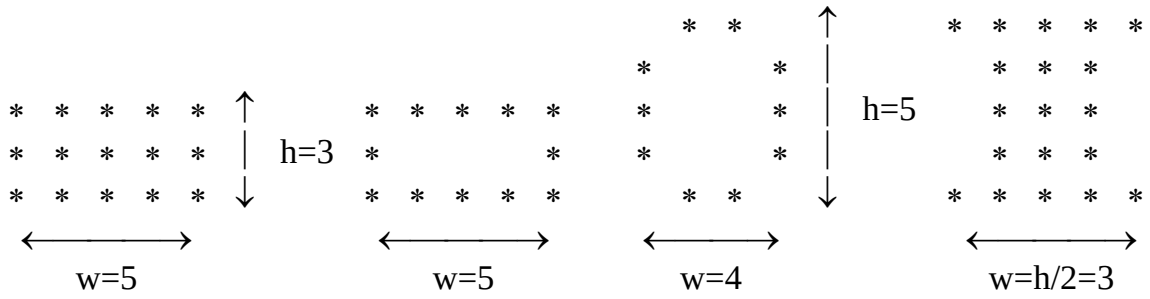
3.4.3.11. Vẽ hình

3.4.3.11.1. Viết chương trình cho người dùng nhập:

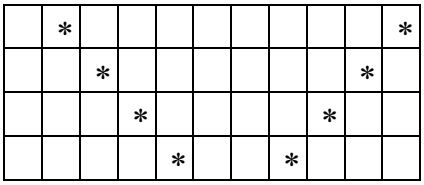
a. Cho người dùng nhập chiều rộng (w) và chiều cao (h), vẽ hình chữ nhật đặc và hình chữ nhật rỗng.

b. Cho người dùng nhập chiều rộng (w) và chiều cao (h), vẽ hình chữ O.

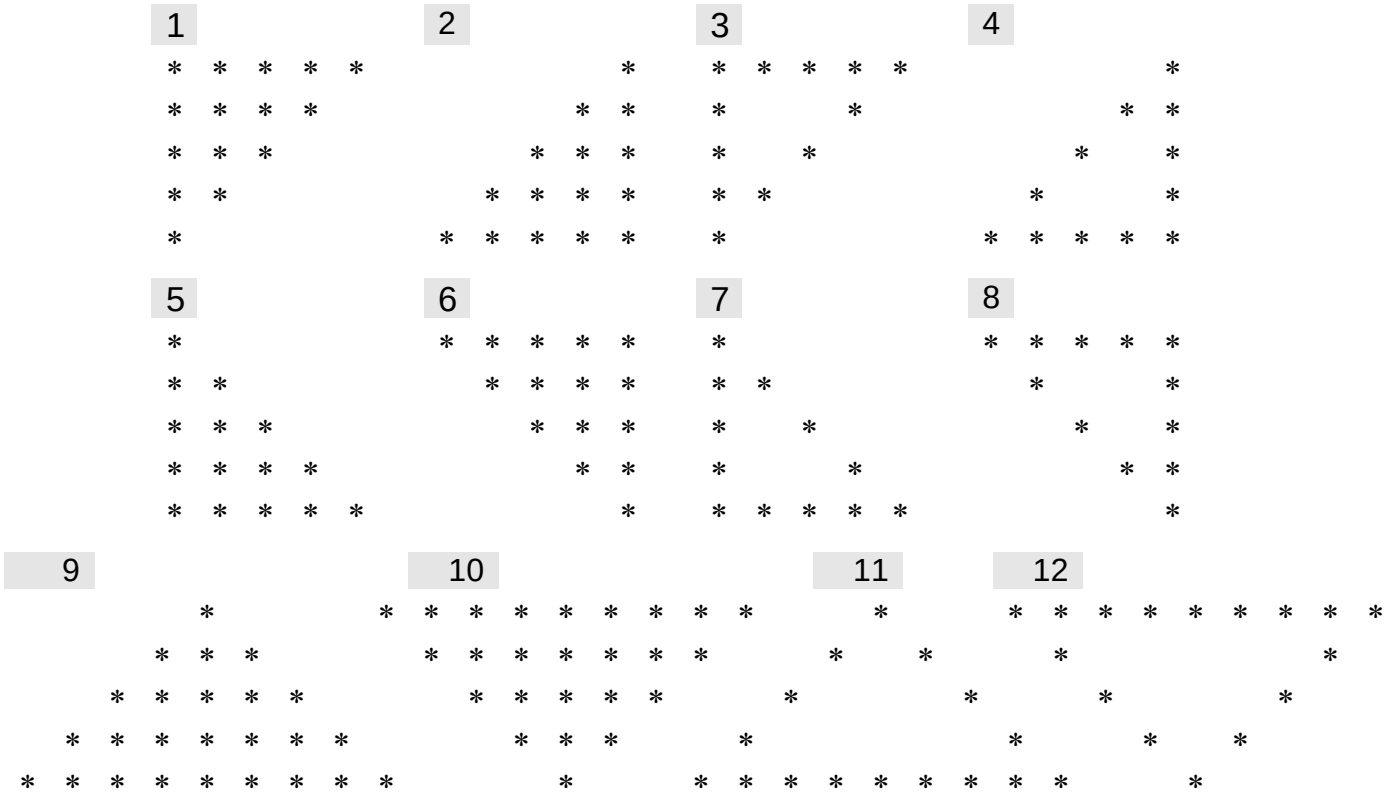
c. Cho người dùng nhập chiều chiều cao (h), vẽ hình chữ I. Biết chiều rộng của chữ I=h/2. Nếu I là số lẻ sẽ được làm tròn.



d. Các đường chéo của hình vuông có cạnh là h Ví dụ : h =4 thì lần lượt in ra từng đường chéo như hình bên:



3.4.3.11.2. Viết chương trình cho người dùng nhập cạnh của tam giác vuông. Lần lượt in ra các hình sau:



13

```
*
* *
*   *
*     *
*       *
*     *
*   *
* *
*
```

14

```
        *
      * *
    *   *
  *     *
*       *
  *     *
    *   *
      * *
        *
```

15

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

16

```
        *
      * *
    * * *
  * * * *
* * * * *
  * * * *
    * * *
      * *
        *
```

17

```
1
22
333
4444
55555
```

18

```
55555
 4444
  333
   22
    1
```

19

```
55555
 4444
  333
   22
    1
```

20

```
    1
   22
  333
 4444
55555
```

21

```
1
12
123
1234
12345
```

22

```
12345
 1234
  123
   12
    1
```

23

```
12345
 1234
  123
   12
    1
```

24

```
    1
   12
  123
 1234
12345
```

25

```
    X
   X X
  X X X
 X X X X
X X X X X
```

26

```
X X X X X
 X X X X
  X X X
   X X
    X
```

27

```
    1
   2 2
  3 3 3
 4 4 4 4
5 5 5 5 5
```

28

```
5 5 5 5 5
 4 4 4 4
  3 3 3
   2 2
    1
```

29

```
    1
   1 2
  1 2 3
 1 2 3 4
1 2 3 4 5
```

30

```
1 2 3 4 5
 1 2 3 4
  1 2 3
   1 2
    1
```

31

```
    X
   X X
  X X X
 X X X X
X X X X X
X X X X X
 X X X X
  X X X
   X X
    X
```

32

```
    1
   2 2
  3 3 3
 4 4 4 4
5 5 5 5 5
5 5 5 5 5
 4 4 4 4
  3 3 3
   2 2
    1
```

33

```
    1
   1 2
  1 2 3
 1 2 3 4
1 2 3 4 5
1 2 3 4 5
 1 2 3 4
  1 2 3
   1 2
    1
```

34

```
1
2 2
3   3
4     4
5       5
4     4
3   3
2 2
1
```

35

```
        1
      2 2
    3   3
  4     4
5       5
  4     4
    3   3
      2 2
        1
```

36

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

3.4.3.11.3. Viết chương trình cho người dùng nhập Cho nhập cạnh (n). Vẽ hình vuông có dạng

n=8

*	*	*	*	*	*	*	*
*							*
*		*	*	*	*		*
*		*			*		*
*		*			*		*
*		*	*	*	*		*
*							*
*	*	*	*	*	*	*	*

n=9

*	*	*	*	*	*	*	*	*
*								*
*		*	*	*	*	*		*
*		*				*		*
*		*				*		*
*		*		*		*		*
*		*				*		*
*		*	*	*	*	*		*
*								*
*	*	*	*	*	*	*	*	*

Hình cánh quạt

```
* * * * *             *
  * * * *         * *
    * * *     * * *
      * * * * * * *
* * * * * * * * * *
* * * * * *
* * *   * * *
* *     * * * *
*       * * * * *
```

3.4.3.11.4. *Hình cánh quạt*: Sử dụng các hàm `cprintf()`, `textcolor()`, `delay()`, `kbhit()`, … thay đổi màu để tạo cảm giác cho cánh quạt xoay cho đến khi nhấn một phím bất kỳ.

3.4.3.12. Hình học giải tích

3.4.3.12.1. Viết chương trình nhập tọa độ của 2 điểm A(x_A,y_A), B(x_B,y_B). Tính chiều dài đoạn AB.

3.4.3.12.2. Viết chương trình nhập tọa độ trực chuẩn của 3 điểm A(x_A,y_A), B(x_B,y_B), C(x_C,y_C) sau đó thực hiện kiểm tra: nếu ABC là tam giác thì in ra tọa độ của điểm D(x_D,y_D) tạo thành hình bình hành ABCD.

3.4.3.12.3. Viết chương trình nhập tọa độ trực chuẩn của 4 điểm A(xa,ya), B(xb,yb), C(xc,yc), M(xm,ym) sau đó thực hiện kiểm tra và in ra kết quả vị trí tương đối của M với ABC: nếu ABC thẳng hàng thì M nằm trên hay nằm ngoài ABC? Hoặc nếu ABC là tam giác thì M nằm trong, nằm trên cạnh hoặc nằm ngoài ABC? Biết Phương trình đường thẳng qua 2 điểm A và B là :

$$(x-x_A) / (x_A-x_B) = (y-y_A) / (y_A-y_B)$$

3.4.3.12.4. Viết chương trình nhập tọa độ trực chuẩn của 3 điểm A(xa,ya), B(xb,yb), C(xc,yc) sau đó thực hiện kiểm tra 3 điểm ABC có tạo thành tam giác không? Nếu 3 điểm ABC tạo thành tam giác thì in ra trị số của chu vi, diện tích và tọa độ trọng tâm của tam giác này.

3.4.3.12.5. Viết chương trình nhập tọa độ trực chuẩn của 3 điểm A(x_A,y_A), B(x_B,y_B), C(x_C,y_C) sau đó thực hiện kiểm tra và in ra kết qủa ABC là hình gì? (đường thẳng, tam giác vuông, tam giác cân, tam giác vuông cân, tam giác đều, tam giác thường)

3.4.3.12.6. Viết chương trình nhập tọa độ trực chuẩn của 2 điểm khác nhau A(x_A,y_A), B(x_B,y_B), sau đó in ra tọa độ 2 điểm C(x_C,y_C), D(x_D,y_D) sao cho ABCD là một hình vuông.

3.4.3.12.7. Viết chương trình nhập tọa độ trực chuẩn của 2 điểm khác nhau A(x_A,y_A), B(x_B,y_B). Sau đó in ra vị trí tương đối của A và B so với đường tròn tâm O(0,0) bán kính OM (M là trung điểm của AB và M <>0).

3.4.3.12.8. Viết chương trình nhập tọa độ trực chuẩn của điểm A(x_A,y_A) và một số dương R sau đó in ra tọa độ giao điểm cuả đường tròn tâm A bán kính R với 2 trục tọa độ.

3.4.3.12.9. Cho biết trong hệ tọa độ trực chuẩn xOy:

- Phương trình đường tròn tâm O(0,0) bán kính R là: $x^2 + y^2 = R^2$.
- Phương trình đường trẳng D // Oy đi qua điểm M(k,0) là: x=k.

3.4.3.12.10. Viết chương trình nhập vào bán kính R của đường tròn O và hoành độ k∈[-R,+R] của điểm M sau đó in ra:

- Tọa độ giao điểm của đường tròn O với đường thẳng D.
- Diện tích tam giác tạo bởi gốc tọa độ O với 2 giao điểm trên.

3.4.3.12.11. Cho biết trong hệ tọa độ trực chuẩn xOy phương trình ellipse có tâm O và trục AB = 2a ∈ xOx', trục CD = 2b ∈ yOy' là: $x^2/a^2 + y^2/b^2 = 1$ với a, b, x, y là các số thực và a>0, b>0. Viết chương trình nhập vào các hệ số a, b của một ellipse dạng nêu trên sau đó in ra tọa độ giao điểm của ellipse với đường phân giác y=x.

3.4.3.12.12. Cho biết trong hệ tọa độ trực chuẩn xOy phương trình Parapol có trục song song với Oy là: $y = ax^2 + bx + c$, với a, b, c, x, y là các số thực và a<>0; Viết chương trình nhập các hệ số a, b, c của một Parapol có dạng trên sau đó in ra tọa độ: đỉnh của Parapol và giao điểm của Parapol với trục hoành nếu có.

3.4.3.13. Chạy chữ

3.4.3.13.1. Vẽ một hình chữ nhật (có kích thước tùy ý) bằng chữ O với yêu cầu:

- Các ký tự O tạo nên hình chữ nhật sẽ đổi màu liên tục và chạy theo chiều quay của kim đồng hồ.
- Chương trình chỉ kết thúc khi người dùng nhấn 1 phím bất kỳ.

3.4.3.13.2. Viết chương trình cho người dùng nhập một chuỗi. Sau đó cho chuỗi đó chạy:

a. Từ trái qua phải.

b. Từ phải qua trái.

c. Từ trên xuống dưới.

d. Từ dưới lên trên.

e. Bổ sung yêu cầu: mỗi lần chuỗi di chuyển thì màu của chuỗi sẽ thay đổi.

3.4.3.13.3. Viết chương trình cho ký tự A chạy từ trái qua phải và ký tự O chạy từ phải qua trái. Hai ký tự này sẽ chạy trên cùng 1 dòng.

Khi 2 ký tự này gặp nhau sẽ tạo ra hàng trăm mảnh vỡ (có hình dáng khác nhau) nằm ngẫu nhiên trên màn hình.

HÀM (Function)

4.1. TỔ CHỨC CHƯƠNG TRÌNH CÓ NHIỀU HÀM

4.1.1. *Khái niệm*

- Sử dụng hàm trong chương trình là sự chia nhỏ của chương trình cần thực hiện.
- Mỗi hàm là một đoạn chương trình độc lập ***thực hiện trọn vẹn một công việc nhất định*** sau khi thực hiện xong, hàm có thể sẽ trả về giá trị cho chương trình gọi nó.

4.1.2. *Mục đích khi sử dụng các hàm con*

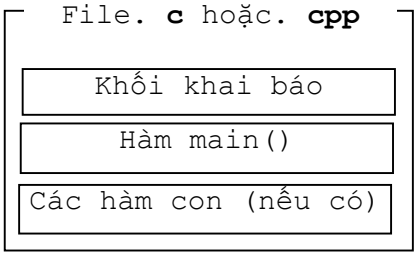
- Khi có một công việc giống nhau cần thực hiện ở nhiều vị trí.
- Chia chương trình có nhiều chức năng thành các chức năng con để chương trình được trong sáng, dễ hiểu, dễ quản lý. Trong mỗi chức năng con vừa có lại có thể phân chia thành những chức năng nhỏ/chi tiết hơn.

4.1.3. *Một số lưu ý khi xây dựng hàm trong C*

- Nội dung của hàm.
- Sự tương tác của nó với những hàm khác, bao gồm việc truyền dữ liệu chính xác vào trong hàm khi gọi hàm thực hiện và giá trị mà hàm sẽ trả về khi thực hiện xong.

4.1.4. *Cấu trúc một chương trình trong C*

4.1.4.1. *Cách 1: sử dụng 1 file cho toàn bộ chương trình*



- *Khối khai báo*: Bao gồm các khai báo về:
  - Khai báo tên các thư viện chuẩn của C được sử dụng trong chương trình.
  - Hằng số sẽ sử dụng trong chương trình.
  - Các kiểu dữ liệu tự định nghĩa.
  - Các biến toàn cục.
  - Hàm con (các nguyên mẫu hàm - prototype).
- *Hàm chính* (main()): Chứa các biến, các lệnh và các lời gọi hàm cần thiết trong chương trình.
- *Các hàm con*:
  - Được sắp xếp sao cho mỗi hàm nằm trên 1 đoạn riêng.
  - Không đặt nội dung của hàm này chứa trong hàm khác, hoặc nội dung của 2 hàm có phần giao nhau.
  - Không cần quan tâm thứ tự sắp xếp trước/sau của các hàm.

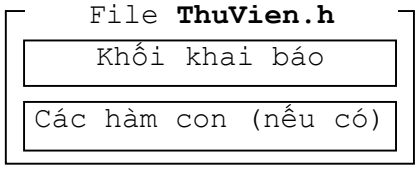
Ví dụ 3.1: toàn bộ chương trình sau được chứa trong file có tên *BaiTap.cphương pháp*

```
//Khối khai báo
  // Khai báo thư viện cần dùng
  #include<conio.h>
  #include<stdio.h>
  #include<string.h>
  #include<dos.h>
  #include<process.h>
  //Khai báo nguyên mẫu hàm
  void ThayThe(char * S, char *St );
  void Doc1Sector(int vt);
  void Ghi1Sector(int vt);
```

```
//Hàm main
void main()
{    unsigned char buf[512];
     char S[20], St[20];
     printf("Nhap chuoi ban dau: ");
     gets(S);
     printf("Nhap chuoi can thay the:");
     gets(St) ;
     printf("\nXin cho…";
     TimVaThayThe(S,St,buf);
     printf("\n Thanh cong. Chuoi ket qua la: %s", S);
     getch();
}
//Khối chứa các hàm con
 void ThayThe(char * S, char *St )
 {  int l=strlen(St);
    for(int i=0;i<l;i++)
           S[i]=St[i];
 }
 void Doc1Sector(int vt, char buf[512])
 {    if(absread(0,1,vt,buf))
      {   printf(""\n loi doc dia,  nhan enter thoat");
          getch();
          exit(1);
      }
 }
 void Ghi1Sector(int vt, char buf[512])
 {    if(abswrite(0,1,vt,buf))
      {   printf("\n loi ghi dia, nhan enter thoat");
          getch();
          exit(1);
      }
 }
 void TimVaThayThe(char * S, char *St, unsigned char buf[])
 {    for(int i=33;i<=500;i++)
      {   Doc1Sector(i, buf);
          char * p=strstr(buf, S);
          if(p)
          {   ThayThe(p, St);
              Ghi1Sector(i, buf);
          }
      }
 }
```

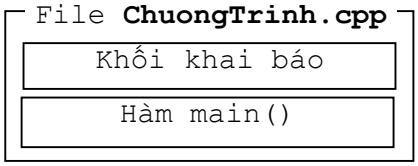
4.1.4.2. Cách 2: sử dụng 2 file cho toàn bộ chương trình

4.1.4.2.1. File thư viện chứa các hàm do người dùng tự tạo



- *Khối khai báo*: Bao gồm các khai báo về:
  - Khai báo tên thư viện chuẩn của ngôn ngữ C được sử dụng trong chương trình.
  - Các kiểu dữ liệu tự định nghĩa.
  - Hằng số sẽ sử dụng trong chương trình.
  - Các biến toàn cục.
  - Hàm con (các nguyên mẫu hàm - prototype).
- *Các hàm con*: (tương tự như cách 1)

4.1.4.2.2. File chương trình



- *Khối khai báo*: khai báo tên file thư viện chứa các hàm do người dùng tự tạo. Tên file (kể cả phần mở rộng) được đặt trong dấu nháy đôi.
- *Hàm chính* (main()): Chứa các biến, các lệnh và các lời gọi hàm cần thiết trong chương trình.

Ví dụ 3.2:

- Nội dung file ***ThuVien.h***

```
//Khối khai báo
  // Khai báo thư viện cần dùng
  #include<conio.h>
  #include<stdio.h>
  #include<string.h>
  #include<dos.h>
  #include<process.h>
  //Khai báo nguyên mẫu hàm
  void ThayThe(char * S, char *St );
  void Doc1Sector(int vt);
  void Ghi1Sector(int vt);
//Khối chứa các hàm con
 void ThayThe(char * S, char *St )
 {  int l=strlen(St);
    for(int i=0;i<l;i++)
          S[i]=St[i];
 }
 void Doc1Sector(int vt, char buf[512])
 {    if(absread(0,1,vt,buf))
      {   printf("\n loi doc dia,  nhan enter thoat");
          getch();
          exit(1);
      }
 }
 void Ghi1Sector(int vt, char buf[512])
 {    if(abswrite(0,1,vt,buf))
      {   printf("\n loi ghi dia, nhan enter thoat");
          getch();
          exit(1);
      }
 }
 void TimVaThayThe(char * S, char *St, unsigned char buf[])
 {    for(int i=33;i<=500;i++)
      {   Doc1Sector(i, buf);
          char * p=strstr(buf, S);
          if(p)
          {   ThayThe(p, St);
              Ghi1Sector(i, buf);
          }
      }
 }
```

- Nội dung file ***ChuongTrinh.cphương pháp***

```
//Khối khai báo
  #include "ThuVien.h"
//Hàm main
void main()
{   unsigned char buf[512];
    char S[20], St[20];
```

```
        printf("Nhap chuoi ban dau: ");
        gets(S);
        printf("Nhap chuoi can thay the:");
        gets(St) ;
        printf("\nXin cho…";
        TimVaThayThe(S,St,buf);
        printf("\n Thanh cong. Chuoi ket qua la: %s", S);
        getch();
}
```

4.2. PHÂN LOẠI THAM SỐ TRUYỀN CHO HÀM

- Các tham số đầu vào của một hàm được gọi là tham số hàm.
- Phân loại: có hai loại tham số là:
  - Tham trị : Giá trị của tham số *KHÔNG thay đổi* sau khi hàm thực hiện.
  - Tham biến: Giá trị của tham số *CÓ thay đổi* sau khi hàm thực hiện.

4.2.1. *Sử dụng tham số của hàm là tham trị*

- Tham trị là cách gọi tắt của "*tham số hình thức **trị***".
- Tham số dạng này chỉ mang ý nghĩa là *dữ liệu đầu vào*.
- Cơ chế truyền và xử lý dữ liệu của cách dùng tham trị như sau:
  - *Hàm gọi* (ví dụ như hàm main) có thể dùng hằng hoặc biến để truyền giá trị cho tham số của chương trình được gọi.
  - *Hàm được gọi* sẽ nhận giá trị truyền cho nó và lưu những giá trị này vào một vùng nhớ gọi là vùng tạm trong bộ nhớ. Khi đó mọi lệnh tác động lên tham số trong chương trình con này sẽ tác động lện vùng nhớ tạm. Vùng nhớ tạm sẽ bị xóa ngay sau khi hàm được gọi kết thúc.

4.2.2. *Sử dụng tham số của hàm là tham biến (hay tham chiếu)*

- Tham biến là cách gọi tắt của "*tham số hình thức **biến***".
- Theo cách này thì khi gọi thực hiện một hàm có tham số thì hàm gọi phải dùng biến để truyền dữ liệu cho các tham số của hàm được gọi. Lúc này tham số sẽ nhận địa chỉ trong bộ nhớ RAM của biến đã truyền cho nó, và mọi tác động lên tham số của hàm được gọi sẽ tác động trực tiếp lên biến tương ứng của chương trình gọi.
- Có thể sử dụng *tham biến* bằng 1 trong 2 cách sau:
  - *Tham biến* vừa là dữ liệu đầu vào vừa là dữ liệu đầu ra.
  - *Tham biến* chỉ đóng vai trò của dữ liệu đầu ra (kết quả thực hiện của hàm). Khi đó, dữ liệu đầu vào không ảnh hưởng đến kết quả thực hiện của hàm.
- Cú pháp để khai báo một tham số dạng tham biến:
  - data_type & //khai báo trên dòng prototype
  - data_type &reference_name //khai báo trên *function* header
- Ví dụ 3.3: chương trình sau sử dụng hàm `swap`, hàm `swap` có công dụng nhận giá trị của hai biến và đổi giá trị của hai biến cho nhau:

```
#include <stdio.h>
#include <conio.h>
void  swap(int&, int&);
void main()
{   int    i=5,j=10;
    printf("\n TRUOC khi goi ham swap,\ni=%d;j=%d\n",i,j);
    swap(i,j);
    printf("\n SAU khi goi ham swap,\n i=%d;j=%d\n",i,j);
    getch();
}
```

```
void  swap(int &x, int &y)
{   int temp=x;
        x=y;
        y=temp;
}
```

4.3. TRIỂN KHAI XÂY DỰNG HÀM

Gồm 3 bước chính:

- Khai báo tiêu đề của hàm
- Viết nội dung cho hàm
- Khai báo nguyên mẫu hàm

4.3.1. *Khai báo tiêu đề hàm (Function Header)*

4.3.1.1. Cú pháp khai báo tiêu đề của một hàm

```
<Kiểu dữ liệu trả về của hàm> Tên hàm  ([danh sách các tham số])
```

4.3.1.2. Đặt tên cho hàm

Mỗi hàm có một tên để phân biệt, tên hàm trong C được đặt theo quy tắc sau:

- Chỉ được dùng chữ cái, chữ số hoặc underscore (_) để đặt tên hàm.
- Ký tự đầu tiên phải là một chữ cái hoặc dấu underscore (_).
- Tên hàm không được trùng tên với từ khóa riêng của C.
- Có phân biệt chữ hoa, chữ thường.
- Số ký tự tối đa là 31.

Nên đặt tên hàm sao cho tên gọi *đúng với chức năng hay mục đích thực hiện của hàm và gợi nhớ.*

4.3.1.3. Kiểu dữ liệu trả về của hàm

Xác định dựa vào kết quả của bài toán (Output). Gồm 2 loại:

- **void:**
  - Hàm không trả về giá trị.
  - Thường dùng cho những chức năng*: Nhập/xuất dữ liệu, thống kê, sắp xếp, liệt kê.*

```
void Tên_hàm (danh sách các tham số)
{     Khai báo các biến cục bộ
     Các câu lệnh/khối lệnh hay lời gọi đến hàm khác
}
```

- **Kiểu dữ liệu cơ bản (rời rạc/ liên tục) hay kiểu dữ liệu có cấu trúc:**
  - Kiểu dữ liệu tùy theo mục đích của hàm cần trả về giá trị gì thông qua việc phân tích bài toán.
  - Thường dùng cho những chức năng: *Đếm, kiểm tra, tìm kiếm, tính trung bình, tổng, tích,.. .*

```
<Kiểu dữ liệu> Tên_hàm ([danh sách các tham số])
{
      <Kiểu dữ liệu> kq;
      Khai báo các biến cục bộ
      Các câu lệnh/khối lệnh hay lời gọi đến hàm khác
      return kq;
}
```

4.3.1.4. Danh sách tham số

- Cần quan tâm:
  - Số lượng tham số.
  - Thứ tự của các tham số. Khi có nhiều tham số, các tham số phải cách nhau bởi dấu phẩy (,).
  - Kiểu dữ liệu của từng tham số.
- Ngoại trừ hàm *main* tất cả các hàm khác (không phải hàm có sẵn của C) có trong chương trình đều phải khai báo nguyên mẫu.

4.3.2. *Nội dung của hàm (hay phần thân hàm – Function Body)*

- Bao gồm các lệnh, các phép toán sẽ tác động lên các giá trị được truyền cho hàm thông qua các tham số, để tạo ra kết quả.
- Mỗi hàm sẽ được định nghĩa một lần trong chương trình.
- Mỗi hàm có thể được gọi thực hiện một (hoặc nhiều lần) bởi một hàm khác có trong chương trình.
- Trong nội dung của hàm có thể gọi bất kỳ hàm nào khác có trong chương trình.

4.3.3. *Khai báo nguyên mẫu hàm (prototypes)*

- Nguyên mẫu hàm thực chất là dòng đầu của hàm thêm dấu chấm phẩy (**;**) vào cuối, tuy nhiên tham số trong nguyên mẫu hàm có thể chỉ có kiểu dữ liệu mà bỏ qua phần tên của tham số.
- Các *prototype* thường được khai báo ở đầu chương trình sau các dòng *#include*.

4.3.4. *Ví dụ 3.4 (về khai báo nguyên mẫu hàm)*

Chương trình in lên màn hình giá trị lớn nhất của hai số nguyên được nhập từ bàn phím, trong chương trình có một hàm *findmax* dùng để tìm giá trị lớn nhất trong hai số:

```
#include <stdio.h>
int TimMax(int, int);     //prototype declare
void main()
{
    int  so1, so2;
    printf("Nhap  so thu nhat:");
    scanf("%d", &so1");
    printf("Nhap  so thu hai:");
    scanf("%d", &so2");
    printf("Gia tri lon nhat trong hai so vua nhap la:%d\n",
                                          TimMax(so1,so2);
}
int TimMax(int x, int y)     // Function Header
{
   if (x>=y) return x;          // Function body
   else return y;               // Function body
}
```

4.3.5. *Lời gọi hàm*

- Lời gọi hàm giống như một lệnh, nó xuất hiện trong chương trình khi có yêu cầu gọi thực hiện một hàm nào đó thực hiện. Lời gọi hàm bao gồm tên hàm các dữ liệu truyền cho hàm được gọi. Nếu nguyên mẫu của hàm có tham số thì khi gọi hàm phải truyền giá trị.
- Số lượng dữ liệu truyền cho hàm phải bằng số lượng tham số khi khai báo nguyên mẫu và đúng thứ tự đã khai báo (cùng kiểu dữ liệu của tham số).

- Minh họa lời gọi hàm CÓ tham số

Nguyên mẫu hàm

```
int addition (int a, int b)
```

Lời gọi hàm

```
z = addition (  5  ,  3  );
```

- Minh họa lời gọi hàm KHÔNG có tham số

Nguyên mẫu hàm

```
void duplicate (int& a,int& b,int& c)
```

Lời gọi hàm

```
duplicate (  x  ,  y  ,  z  );
```

4.4. VÍ DỤ (về trình tự khi xây dựng hàm)

4.4.1. *Ví dụ 3.5*

Viết hàm nhận số nguyên dương n và in ra màn hình các ước số của n.

4.4.1.1. Phân tích bài toán

- **Input:** n (tham số)
  - Kiểu dữ liệu: số nguyên dương (*unsigned int*).
  - Giá trị n không bị thay đổi sau quá trình hàm thực hiện ⇨ Tham số của hàm là tham trị (không có ký hiệu & trước tên biến).
- **Output:** chỉ in ra các ước số của n mà không trả về giá trị nên kiểu dữ liệu của hàm là *void*.
- **Xác định tên hàm:** Hàm này dùng in ra các ước số của n nên có thể đặt là *LietKeUocSo.*

Ta có nguyên mẫu hàm:

```
void LietKeUocSo (unsigned int n);
```

4.4.1.2. Bổ sung nội dung hàm

```
#include<conio.h>
#include<stdio.h>
//Khai báo nguyên mẫu hàm
void LietKeUocSo (unsigned int n);
void main()
{   unsigned int n;
    printf("Nhap n: )";
    scanf ("%d",&n);
    printf("Cac uoc so cua n: )" ;
    LietKeUocSo(n); //hàm có kiểu dữ liệu hàm là void
                    //nên không cần gán giá trị vào biến
    getch();
}
void LietKeUocSo (unsigned int n)
{   for(int i=1; i<=n; i++)
        if (n%i==0)
          printf("%d \t",i);
}
```

4.4.2. *Ví dụ 3.6*

Viết chương trình nhập số nguyên dương n và tính tổng

$$S = 1+2+3+\cdots+n \quad \text{, với n>0}$$

4.4.2.1. Phân tích bài toán

- **Input:** n (tham số)
  - Kiểu dữ liệu: số nguyên dương (*unsigned int*).
  - Giá trị n không bị thay đổi sau quá trình hàm thực hiện ⇨ Tham số của hàm là tham trị (không có ký hiệu & trước tên biến).
- **Output**: Trả về giá trị tổng của các số từ 1 đến n. Do n là số nguyên dương nên S cũng là số nguyên dương ⇨ Kiểu trả về của hàm là *unsigned int* (hoặc *unsigned long* cho trường hợp giá trị của tổng lớn hơn 2 bytes).
- **Xác định tên hàm**: Hàm này dùng tính tổng S nên có thể đặt là *TongS.*

Ta có nguyên mẫu hàm:

```
unsigned long TongS ( unsigned int n );
```

4.4.2.2. Bổ sung nội dung hàm

```
//Khai báo nguyên mẫu hàm
unsigned long TongS (unsigned int n);
void main()
{   unsigned int n;
    unsigned long kq;
    printf("Nhap n: )";
    scanf ("%d",&n);
    /* hàm có kiểu dữ liệu hàm không phải là void nên cần gán
    giá trị vào biến ở vế trái của biểu thức */
    kq  = TongS(n);
    printf("Tong can tinh la: %d", kq);
    getch( );
}
unsigned long TongS (unsigned int n)
{   unsigned long S=0;
    int i=1;
    while(i<=n)
    {   S+=i;
        i++;
    }
    return S;
}
```

4.5. PHẠM VI CỦA BIẾN

4.5.1. *Phân loại*

- **Biến cục bộ:** Là các biến được khai báo trong thân của một hàm nào đó. Các biến này chỉ có giá trị trong phạm vi hàm khai báo nó.
- **Biến toàn cục:** Là các biến được khai báo bên ngoài các hàm. Những biến này có giá trị với tất cả các hàm được khai báo sau nó.

4.5.2. *Ví dụ 3.7 (về phạm vi của biến)*

```
int a;
void  func(int );
void main()
{
    int   b=2;
    a=1;
    printf("\nTRUOC khi goi ham func, trong ham main() a=%d,
                                                    b=%d\n",a,b);
    func();
    printf("\nSAU khi goi ham func, trong ham main() a=%d,
                                                    b=%d\n",a,b);
}
void func()
{
    int b=3;
    printf("\nTRONG ham func, a=%d, b=%d\n",a,b);
    a=4;
}
```

📖 *Giải thích*:

- Trong ví dụ trên a là biến toàn cục, và b là biến cục bộ có cả trong `main` và `func`. Lệnh printf đầu của hàm `main` sẽ in lên kết quả:
  `TRUOC khi goi ham func, trong ham main(), a=1, b=2`
- Khi hàm `func` thực hiện, do b là biến cục bộ của hàm nên hàm sẽ in ra kết quả:
  `TRONG ham func a= 1; b=3`
- Lệnh gán cuối cùng trong hàm `func` (`a=4;`) đã làm thay đổi giá trị của biến toàn cục a, nên lệnh printf cuối của hàm `main` sẽ in lên kết quả:
  `SAU khi goi ham func, trong ham main(), a=4, b=2`
- Trong ví dụ trên, nếu ta khai báo biến a trong hàm `main` thì hàm `func` sẽ không hiểu được biến này, do đó khi biên dịch sẽ gây ra lỗi.

4.5.3. *Trùng tên giữa 2 biến cục bộ và toàn cục*

- Khi biến toàn cục và biến cục bộ trùng tên thì các biến cục bộ sẽ ưu tiên được hiểu trong hàm khai báo nó.
- Ví dụ 3.8:

```
int a=2;
void  func(int );
void main()
{
    a=2;
    printf("\nTRUOC khi goi ham func, a=%d\n",a);
    func();
    printf("\nSAU khi goi ham func, a=%d\n",a);
}
void func()
{
    int a=3;
    printf("\nTRONG ham func,a=%d\n",a);
}
```

📖 *Giải thích*:

- Trong ví dụ 3.8, a là biến toàn cục, trong hàm `main` a được gán giá trị là 2. Lệnh printf đầu của hàm `main` sẽ in lên kết quả:
  ```
  TRUOC khi goi ham func, a=2
  ```
- Khi hàm `func` thực hiện, do biến a được khai báo là biến cục bộ của hàm nên hàm sẽ in ra kết quả:
  ```
  TRONG ham func a=3
  ```
- Lệnh thứ tư trong hàm `main` gán giá trị cho biến toàn cục a tang lên 1 đơn vị (`a++;`), nên lệnh printf cuối của hàm `main` sẽ in lên kết quả:
  ```
  SAU khi goi ham func, a=2
  ```

4.5.4. *Lưu ý khi sử dụng biến toàn cục*

- Sẽ dễ dẫn đến phá vỡ sự độc lập của các hàm.
- Làm mất đi sự cẩn thận cần thiết của người lập trình là:
  - Phải xác định rõ kiểu dữ liệu của tham số.
  - Quản lý biến cục bộ
  - Giá trị trả về của hàm.
- Do giá trị của biến toàn cục có thể thay đổi bởi bất cứ một hàm nào khai báo sau nó, vì vậy việc phát hiện ra những lỗi giải thuật có trong chương trình là rất khó khăn.

4.6. CÁC BƯỚC ĐỂ VIẾT MỘT CHƯƠNG TRÌNH CÓ SỬ DỤNG HÀM

Để viết các chương trình có sử dụng hàm ta theo các bước chung như sau:

- *Bước 1*: Xác định các bước cần làm để chia vấn đề muốn giải quyết thành các vấn đề nhỏ hơn.
- *Bước 2*: Với mỗi vấn đề nhỏ, chúng ta xác định các đối tượng có sẵn (còn gọi là "Các đối tượng nhập" hay " Dữ liệu vào"), các đối tượng cần phải tính (còn gọi là "Các đối tượng xuất" hay " Dữ liệu ra"), các đối tượng trung gian (các đối tượng cần sử dụng tạm trong quá trình tính toán).
- *Bước 3*: Chuyển mỗi vấn đề nhỏ thành một chương trình con (Hàm), các đối tượng nhập được chuyển thành tham trị, các đối tượng xuất được chuyển thành tham chiếu, các đối tượng trung gian được chuyển thành các biến riêng (còn được gọi là biến cục bộ) của hàm.
- *Bước 4*: Viết chương trình chính (hàm main): Khai báo các đối tượng được dùng trong chương trình chính, gọi các các chương trình con theo một trình tự thích hợp.

4.7. CÁC LỖI THƯỜNG GẶP KHI XÂY DỰNG VÀ SỬ DỤNG HÀM

- Lỗi thường gặp là truyền dữ liệu cho tham số khi gọi hàm thực hiện không chính xác chẳng hạn như truyền thiếu/thừa, dùng hằng để truyền cho tham chiếu.
- Cần lưu ý khi có hai biến có cùng tên cùng xuất hiện trong hàm gọi lẫn làm được gọi.
- Thiếu các nguyên mẫu của các hàm có trong chương trình.
- Thiếu dấu chấm phẩy ở cuối các dòng nguyên mẫu.
- Dư dấu chấm phẩy ở cuối dòng tiêu đề của mỗi hàm.
- Không ghi kiểu dữ liệu của tham số trong nguyên mẫu cũng như tiêu đề hàm.

4.8. BÀI TẬP

4.8.1. *Viết dòng mô tả của các hàm theo mô tả như sau:*

4.8.1.1. Hàm sẽ nhận một giá trị kiểu double truyền cho nó khi được gọi thực hiện. Hàm trả về trị tuyệt đối của giá trị được truyền.

4.8.1.2. Hàm sẽ nhận 2 tham số kiểu float. Hàm trả về tích của 2 tham số.

4.8.1.3. Hàm nhận hai tham số kiểu số nguyên a và b. Sau khi thực hiện xong, hàm trả về giá trị của a lũy thừa b.

4.8.1.4. Hàm nhận một tham số kiểu số thực. Hàm trả về giá trị bình phương của tham số đầu vào.

4.8.1.5. Hàm nhận một tham số kiểu số nguyên n. In ra bảng cửu chương của tham số nhận vào.

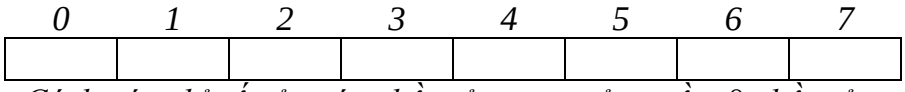
4.8.2. *Viết lại các bài tập trong các chương trước bằng kỹ thuật lập trình hàm.*

MẢNG MỘT CHIỀU (One Dimensional Array)

5.1. GIỚI THIỆU

5.1.1. *Khái niệm*

- Mảng thực chất là một biến được cấp phát bộ nhớ liên tục và bao gồm nhiều biến thành phần.
- Chỉ số mảng (chỉ mục): là số thứ tự của các phần tử trong mảng (tính từ 0).



Cách gán chỉ số của các phần tử trong mảng gồm 8 phần tử

5.1.2. *Khai báo mảng*

```
<Kiểu dữ liệu> <Tên mảng> [<Số phần tử tối đa của mảng>];
```

- Ví dụ:

```
int A[100];//Khai bao mang so nguyen a gom 100 phan tu
float b[50];//Khai bao mang so thuc b gom 50 phan tu
```

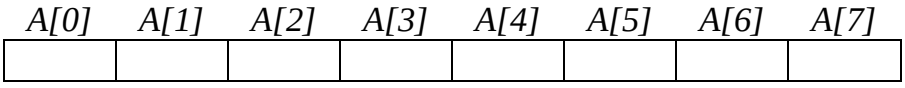
- Thông thường trong lập trình người ta thường định nghĩa trước một hằng chứa số phần tử của mảng. Điều này rất có lợi khi ta cần thay đổi số phần tử của mảng. Như vậy ta có thể khai báo một mảng như sau:

```
const int SIZE=500;
int diem[SIZE];
```

Hay

```
const int SIZE=8;
char A[SIZE];
```

- Khi khai báo như trên thì mỗi mảng sẽ được cung cấp đủ số ô nhớ cần thiết để lưu trữ các phần tử của mảng. Và các phần tử của một mảng được lưu trữ liên tục với nhau.



Hình ảnh của mảng A được lưu trong bộ nhớ

5.1.3. *Truy xuất các phần tử của mảng*

- Các thành phần của mảng là tập hợp các biến có cùng kiểu dữ liệu và cùng tên. Do đó để truy xuất các biến thành phần, ta dùng cơ chế chỉ mục.
- Cú pháp:

```
array_name[offset]
```

- Ví dụ: như A[2] (truy xuất đến phần từ thứ hai của mảng a và là phần tử thứ 3 tính từ đầu mảng).
- Mỗi một phần tử của mảng có thể sử dụng bất cứ ở vị trí nào mà một biến vô hướng hợp lệ xuất hiện. Ví dụ các vị trí xuất hiện của mảng A sau đây là hợp lệ:

```
A[0]= 'A';
A[i]=A[0]+32
```

- Chỉ số của một phần tử đặt trong [] là một hằng số nguyên hoặc bất kỳ một biểu thức nào mà giá trị của nó là một số nguyên.

```
Ví dụ:     A[i] , A[i+1], A[i*2]
```

- Có thuận lợi rất lớn khi ta sử dụng một biểu thức như là chỉ số của mảng, bởi khi đó nó cho phép ta có thể truy xuất từ đầu đến cuối mảng bằng một vòng lặp.

Ví dụ: có một mảng được khai báo như sau:

```
const int SIZE=5;
int A[SIZE];
```

giả sử như mảng đã có giá trị và ta cần tính tổng giá trị của mảng đưa vào biến sum, thay vì phải viết lệnh:

```
sum= A[0]+A[1]+A[2]+A[3]+A[4];
```

có thể viết lại như sau:

```
for (i=0;i<SIZE;i++)
     sum +=A[i];
```

Việc sử dụng vòng lặp để truy xuất một mảng rõ ràng rất thích hợp khi truy xuất một mảng lớn.

5.1.4. *Tham số của hàm là mảng*

- Khi một mảng được truyền cho một hàm thông qua tham số, địa chỉ bắt đầu của vùng nhớ dành cho mảng sẽ được truyền cho tham số vì vậy hàm được gọi sẽ tác động trực tiếp lên vùng nhớ của mảng truyền cho nó.
- Ví dụ: Xét chương trình sau:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
//Khai bao hang so SIZE
const int SIZE=100;
//Khai bao prototype
void TaoMangNgauNhienDuong(int A[],int n);
void XuatToanBoMang (int A[], int n);
// Ham chinh cua chuong trinh
int main()
{   int n, A[SIZE];
    srand((unsigned int) time (NULL));
    printf("Nhap so phan tu cua mang: ");
    scanf("%d",&n);
    TaoMangNgauNhienDuong(A,n);
    printf("Mang da nhap:");
    XuatToanBoMang (A,n);
    return 0;
}
// Phan chua cac ham
void TaoMangNgauNhienDuong(int Arr[],int n)
{   int min=50;
    int max=100;
    for (int i=0; i<n; i++)
        Arr[i]=(rand()%(max-min+1))+min;
}
void XuatToanBoMang (int Arr[], int n)
{   for (int i = 0; i < n; i ++)
        printf("%5d",Arr[i]);
}
```

5.2. MỘT SỐ KỸ THUẬT CƠ BẢN XỬ LÝ TRÊN MẢNG

5.2.1. *Nhập*

5.2.1.1. Nhập (gán) dữ liệu cho 1 phần tử

- Dùng lệnh gán, ví dụ: A[0]='A';
- Dùng *scanf*. Ví dụ: scanf("%d",&A[3]);

5.2.1.2. Khởi tạo giá trị cho mảng tại thời điểm khai báo:

- Khởi tạo bằng cách đặt các giá trị này vào cặp dấu ngoặc nhọn ({}), các giá trị cách nhau bởi dấu phẩy (,).
- Ví dụ:

```
const int SIZE=5;
int A[SIZE]={2,95,9,6,3};
```

các giá trị khởi tạo cho mảng cũng có thể đặt trên nhiều dòng

```
const int SIZE=10;
int  Arr[10]= {2,6,1,7,8,
               5,3,9,10,12};
```

- Nếu các giá trị liệt kê trong {} ít hơn số phần tử của mảng thì chỉ có các phần tử đầu được khởi tạo tương ứng với các giá trị trong {} các phần tử còn lại sẽ được khởi tạo với giá trị =0.
- Gán cùng 1 giá trị cho tất cả các phần tử của mảng (thường dùng khi khởi tạo giá trị ban đầu cho mảng):

VD: `int A[SIZE]={0};`

5.2.1.3. Nhập dữ liệu lần lượt cho nhiều phần tử của mảng bằng lệnh lặp

5.2.1.3.1. Người dùng tự nhập giá trị cho từng phần tử của mảng

Ví dụ:

```
void NhapTungPhantuMang(int A[],int n)
{
    for (int i=0; i<n; i++)
    {   printf("Nhap gia tri cho phan tu thu %d",i);
        scanf("%d",&A[i]);
    }
}
```

5.2.1.3.2. Phát sinh giá trị ngẫu nhiên cho từng phần tử của mảng

Ví dụ:

```
void TaoMangNgauNhienDuong(int A[],int n)
{
    int min=50;
    int max=100;
    for (int i=0; i<n; i++)
        A[i]=(rand()%(max-min+1))+min;
}
void TaoMangNgauNhienAmDuong(int A[],int n)
{   int min=50;
    int max=100;
    for (int i=0; i<SIZE; i++)
    {   //B1: tao dau +/-
        int dau,x;
        x=rand();
        if (x%2==1)
```

```
                dau=-1;
            else
                dau=1;
            //B2: random so X dau
            A[i]=dau*((rand()%(max-min+1))+min );
        }
    }
```

5.2.2. *Xuất (liệt kê) mảng một chiều*

Lưu ý: Hầu hết các lệnh của C khi thực hiện không kiểm tra chỉ số của mảng đang dùng có nằm trong phạm vi hợp lệ hay không (không báo lỗi khi biên dịch). Giả sử ta khai báo một mảng int A[10] , nhưng khi ta truy xuất ta lại truy xuất đến những phần tử có chỉ số >9, điều này có thể gây ra lỗi trong quá trình xử lý của chương trình.

5.2.2.1. *Xuất (liệt kê) toàn bộ các phần tử của mảng*

```
void XuatToanBoMang (int A[], int n)
{
    for (int i = 0; i < n; i ++)
        printf("%5d",A[i]);
}
```

5.2.2.2. *Xuất (liệt kê) các phần tử của mảng thỏa điều kiện cho trước*

5.2.2.2.1. *Điều kiện lựa chọn được gán trực tiếp trong biểu thức điều kiện của phát biểu if*

- *Điều kiện căn cứ trên **giá trị** của từng phần tử trong mảng:*
  - Xuất các số chẵn có trong mảng

```
void XuatSoChanCoTrongMang (int A[], int n)
{
    for (int i = 0; i < n; i ++)
        if(A[i]%2==0)
            printf("%5d",A[i]);
}
```

  - Xuất các số âm có trong mảng

```
void XuatSoAmCoTrongMang (int A[], int n)
{
    for (int i = 0; i < n; i ++)
        if(A[i]<0)
            printf("%5d",A[i]);
}
```

- *Điều kiện căn cứ vào **vị trí** của phần tử trong mảng:*
  - Xuất các số tại vị trí chẵn trong mảng

```
void LietKeSoTaiViTriChan (int A[], int n)
{
    for (int vitri = 0; vitri < n; vitri ++)
        if(vitri %2==0)
            printf("\nVi tri %d chua so %d", vitri,A[i]);
}
```

5.2.2.2.2. Điều kiện lựa chọn là kết quả trả về của một hàm trong biểu thức điều kiện của phát biểu if

```
void LietKeSoTaiViTriChan (int A[], int n)
{
    for (int i=0; i<n; i++)
      if(LaSNT(A[i]))
        printf("\nVi tri %d chua so %d", i, A[i]);
}

bool LaSNT(int k)
{   int dem=0;
    for(int i=1;i<=k;i++)
        if (k%i==0)
            dem++;
    if (dem==2)
        return true;
    else
        return false;
}
```

5.2.3. *Tính tổng – Tính trung bình có điều kiện*

5.2.3.1. Tính tổng

5.2.3.1.1. Tính tổng toàn bộ các phần tử trong mảng

VD: tính tổng tất cả các số có trong mảng

```
int  TinhTong (int A[], int n )
{
    int  tong = 0;
    for (int i = 0; i < n; i++ )
        tong = tong + A[i] ;
    return tong;
}
```

5.2.3.1.2. Tính tổng các phần tử trong mảng thỏa điều kiện cho trước

VD: tính tổng các số âm có trong mảng

```
int  TinhTong (int A[], int n )
{
    int  tong = 0;
    for (int i = 0; i < n; i++ )
        if (A[i]<0)
            tong = tong + A[i] ;
    return tong;
}
```

5.2.3.2. Tính trung bình

5.2.3.2.1. Tính giá trị trung bình toàn bộ các phần tử trong mảng

VD: tính tổng tất cả các số có trong mảng

```
double  TrungBinhAm (int A[], int n )
{
    long  tong = 0;
    for (int i = 0; i < n; i++ )
        if( A[i]<0 )
            tong = tong + A[i] ;
```

```
        return (double)tong/spt;
    }
```

5.2.3.2.2. *Tính tổng các phần tử trong mảng thỏa điều kiện cho trước*

Đối với hàm tính trung bình có điều kiện phải lưu ý khi chia giá trị (có thể mảng không có phần tử nào thoả điều kiện, nếu ta chia tức là chia cho 0).

VD: tính tổng các số âm có trong mảng

```
double  TrungBinhAm (int A[], int n )
{
    long  tong = 0;
    int spt=0;
    for (int i = 0; i < n; i++ )
        if( A[i]<0 )
        {   tong = tong + A[i] ;
            spt++;
        }
    if(spt==0)
                return 0;
    return (double)tong/spt;
}
```

5.2.4. *Kỹ thuật đặt cờ hiệu*

*Kỹ thuật này thường được áp dụng cho những bài toán **"kiểm tra"** (có hay không, đúng hay sai, ...) hay **"đánh dấu"**.*

5.2.4.1. *Dạng 1: (dạng ∀)*

Sử dụng khi cần kiểm tra điều kiện trên toàn bộ mảng (mọi phần tử đều phải thoả điều kiện)

5.2.4.1.1. *Kiểm tra giá trị từng phần tử riêng lẻ*

Thường dùng trong những trường hợp kiểm tra mảng toàn dương, toàn âm, đồng nhất, … Khi đó người ta thường dùng lượng từ tồn tại ∃ để kiểm tra. Ví dụ:

Yêu cầu cần kiểm tra	*Điều kiện cần kiểm tra*
Tất cả các phần tử trong mảng đều là số dương	Không tồn tại phần tử trong mảng là số âm
A[i]>=0; $\forall_i$	$\neg\exists$ (A[i]<0) $\Rightarrow$ là đúng ①

Lấy phủ định 2 vế của ① $\Leftrightarrow$ $\exists$ (A[i]<0) $\Rightarrow$ là sai ②

Hay nói cách khác điều kiện dùng trong phát biểu IF là điều kiện ngược với điều kiện muốn kiểm tra. Nếu điều kiện ngược này xuất hiện thì kết thúc (KHÔNG thành công).

VD: Viết hàm kiểm tra xem tất cả các phần tử có trong mảng đều là số dương hay không? (Trả về true nếu mảng toàn dương, ngược lại trả về false).

```
bool KiemTraMangToanDuong (int A[ ], int n)
{
    for (int i = 0; i < n; i ++ )
        if (A[i]<0) // ⇔ !(A[i]>=0)⇨Vi phạm điều kiện dương
            return  false;
    //Nếu đã tìm hết trên mảng nhưng vẫn không thấy số âm
    //⇒ mảng chứa toàn số dương
    return true;
}
```

5.2.4.1.2. Kiểm tra giá trị giữa 2 phần tử trong mảng

Thường dùng trong những trường hợp kiểm tra mảng tăng, mảng giảm, mảng đối xứng, …

VD: Viết hàm kiểm tra xem có phải là mảng tăng hay không? (Trả về true nếu mảng tăng dần, ngược lại trả về false).

```
bool KiemTraMangTangDan (int A[ ], int n)
{
    for (int i = 0; i < n-1; i++)
        if (A[i]<A[i+1]) // ⇔ !(A[i]>=A[i+1])⇨Vi phạm
            return  false;
    //Nếu đã duyệt hết trên mảng nhưng vẫn không thấy vi phạm
    //⇒ mảng tăng
    return true;
}
```

5.2.4.2. Dạng 2: (dạng ∃)

Sử dụng khi cần kiểm tra có xuất hiện (hay tồn tại) một giá trị x nào đó. Khi đó điều kiện dùng trong phát biểu IF là điều kiện đúng với điều kiện muốn kiểm tra. Nếu điều kiện này xuất hiện thì kết thúc (THÀNH CÔNG).

VD: Viết hàm kiểm tra xem trong mảng các số nguyên có tồn tại số lẻ?

```
bool KiemTraMangCoChuaSoLeLonHon100 (int A[ ], int n)
{
    for (int i = 0; i < n; i ++ )
        if (A[i]%2==1)//Gặp phần tử thoả
            return  true;
    /* Khi kết thúc vòng lặp for (đã duyệt hết mảng) nhưng
    không thấy phần tử thỏa điều kiện (chưa return true)⇒ mảng
    không chứa số lẻ*/
    return false;
}
```

5.2.5. *Kỹ thuật đặt lính canh*

*Kỹ thuật này thường được áp dụng cho những bài tập về **"tìm kiếm"**, **"liệt kê"** theo một điều kiện nhất định nào đó.*

5.2.5.1. Viết hàm tìm và trả về giá trị lớn nhất trong mảng một chiều các số nguyên.

```
int TimMax (int A[], int n)
{   int max;
    max = A[0];
    for (int i = 1; i < n ; i ++)
        if ( A[i] > max )
            max = A[i] ;
    return max;
}
```

5.2.5.2. Viết hàm tìm phần tử có giá trị x xuất hiện đầu tiên trong mảng một chiều. (Nếu tìm thấy trả về vị trí xuất hiện x, ngược lại trả về -1)

```
int TimX (int A[], int n, int x)
{   for (int i = 0; i < n ; i ++)
        if ( x==A[i] )
            return i;
    return -1;
}
```

5.2.6. *Đếm số lần xuất hiện của số trong mảng*

Có thể chia các bài toán về đếm thành 3 loại:

5.2.6.1. Đếm số lần xuất hiện của số thỏa điều kiện cho trước

Thường dùng trong những trường hợp đếm số lượng số có giá trị = x, đếm số âm, đếm số nguyên tố, …)

```
int DemSoChiaChanCho5 (int A[], int n )
{   int dem = 0;
    for (int i = 0; i < n ; i++ )
        if ( A[i] % 5 == 0 )
            dem++;
    return dem;
}
```

5.2.6.2. Đếm số lần xuất hiện của 1 gá trị nào đó

Thường dùng trong những trường hợp đếm số lượng số có giá trị bằng với giá trị x, hay nhỏ nhất, hoặc số trung bình, …) xuất hiện trong mảng. Tương tự như 4.2.6.1, nhưng trước khi thực hiện đếm, cần qua bước tìm số lớn nhất (số nhỏ nhất, số trung bình, …).

```
int DemSoLanXuatHienCuaSoLonNhat (int A[], int n )
{   //Sử dụng hàm TimMax trong các ví dụ ở các phần trước
    int max=TimMax(a,n);
    int dem = 0;
    for (int i = 0; i < n ; i++ )
        if ( A[i] == max )
            dem++;
    return dem;
}
```

5.2.6.3. Đếm số lần xuất hiện của từng số có trong mảng

Yêu cầu: đếm số lần xuất hiện của các số có trong mảng A (chỉ chứa số nguyên dương).

Sử dụng mảng phụ (B) chứa kết quả đếm, với chỉ số của mảng phụ đại diện cho số xuất hiện trong mảng chính (A).

```
int DemSoLanXuatHienCuaCacSoTrongMang (int A[], int n )
{   //Gọi hàm TimMax đã có trong các ví dụ trước)
    int max=TimMax(a,n);
    //mảng B gồm max+1 phần tử
    int B[max+1];
    for (int i = 0; i < max+1 ; i++ )
        B[i]=0;
    //đếm các số trên mảng A. Ket qua dem luu vao mang B
    for (int i = 0; i < n ; i++ )
        B[A++]++;
    //xuất kết quả đếm
    printf("Cac so xuat hien trong mang A:\n");
    for (int i = 0; i < max+1 ; i++ )
        if (B[i]>0)
            printf("So %d xuat hien %d lan",i,B[i]);
}
```

5.2.7. *Sắp xếp*

Viết hàm sắp xếp mảng theo thứ tự tăng dần.

```
void HoanVi (int &a, int &b)
{   int tam = a;
    a = b;
    b = tam;
}
void SapTang (int A[], int n)
{   for (int i = 0; i < n-1 ; i++)
        for (int j = i+1; j < n; j++)
            if (A[i] > a [j])
                HoanVi (A[i], A[j]);
}
```

5.2.8. *Xoá 1 phần tử khỏi mảng*

Các bước thực hiện:

B1: *Duyệt mảng từ trái sang phải, trong quá trình duyệt, sẽ tìm giá trị (hoặc vị trí) cần xóa.*
B2: *Xuất phát từ vị trí cần xoá tiến hành dời lần lượt các phần tử về phía trước cho đến khi kết thúc mảng.*
B3: *Giảm kích thước mảng.*

5.2.8.1. *Viết hàm xoá phần tử tại vị trí (vitri) cho trước trong mảng*

```
void XoaTaiViTri (int A[], int &n, int vitri)
{   //Dời sang tri từ vitri den n-1
    for (int i = vitri; i < n-1 ; i++)
        A[i] = A[i+1];
    //Giảm n di 1 đơn vị
    n--;
}
```

5.2.8.2. *Viết hàm xoá tất cả các phần tử trong mảng có giá trị =Z*

```
void XoaTatCaGiaTri (int A[], int &n, int SoCanXoa)
{ for (int i = 0; i < n ; )
    /*để trống biểu thức 3 nhằm tránh bỏ sót khi có 2 phần tử
    cần xóa nằm liền kề nhau*/
    if (A[i] == SoCanXoa)
        XoaTaiViTri (a, n, i);
    else
        i++;  // tăng biến chạy của vịng lặp
}
```

Hàm **XoaTatCaGiaTri** thường được viết lại bằng cách thay thế phát biểu FOR bằng phát biểu WHILE như sau:

```
void XoaTatCaGiaTri (int A[], int &n, int SoCanXoa)
{
    int i = 0;
    while (i<n)
        if (A[i] == SoCanXoa)
          XoaTaiViTri (A, n, i);
        else
          i++;
}
```

5.2.9. *Chèn (hay thêm) 1 phần tử vào mảng*

Các bước thực hiện:

> **B1:** *Duyệt mảng từ phải sang trái để tìm vị trí cần chèn.*
> **B2:** *Xuất phát từ cuối mảng tiến hành đẩy lần lượt các phần tử về phía sau cho đến vị trí cần chèn.*
> **B3:** *Chèn phần tử cần chèn vào vị trí chèn*
> **B4:** *Tăng kích thước mảng.*

5.2.9.1. *Thêm phần tử có giá trị X vào cuối mảng*

```
void ThemCuoi (int A[], int &n, int X)
{
    A[n]=X;
    n++;
}
```

5.2.9.2. *Chèn phần tử có giá trị X vào mảng tại vị trí cho trước*

```
void ChenXVaoViTri  (int A[], int &n, int X, int vitri)
{
    /* Xuất phát từ cuối mảng tiến hành đẩy lần lượt các phần tử
    về phía sau cho đến vị trí cần chèn*/
    for (int i = n; i >vitri ; i--)
        A[i] = A[i-1] ;
    // Đưa phần tử cần chèn vào vị trí chèn
    A[vitri] = X;
    // Tăng kích thước mảng
    n++;
}
```

5.2.9.3. *Chèn phần tử có giá trị X vào sau giá trị Y đầu tiên (tính từ trái sang phải) có trong mảng. Nếu trong mảng không tồn tại giá trị Y thì thực hiện thêm X vào cuối mảng.*

```
void ChenXVaoSauY  (int A[], int &n, int X, int Y)
{
    int i;
    for (i = 0; i < n ; i++)
        if (A[i] == Y)
        {   ChenXVaoViTri(A, n, X, i);
            break;
        }
    if (i==n)//Y không tồn tại ⇒ thêm X vào cuối mảng
        ThemCuoi (A, n, X);
}
```

5.2.10. *Tách mảng*

Cho mảng A kích thước n. Tách mảng A thành 2 mảng B và C sao cho: B có ½ phần tử đầu của mảng A, ½ phần tử còn lại đưa vào mảng C.

Vậy nếu n là số chẵn thì số lượng 2 mảng B và C bằng nhau; ngược lại nếu n lẻ mảng C sẽ nhiều hơn mảng B 1 phần tử.

```
void TachMang(int A[],int n,int B[],int &p,int C[],int &q)
{
    int k = n/2;
    p = q = 0;
    for(int i=0; i<k; i++)
    {
        B[p++]=A[i];
        C[q++]=A[k+i];
    }
}
```

5.2.11. *Ghép mảng*

5.2.11.1. Ghép tuần tự

Cho 2 mảng số nguyên A và B kích thước lần lượt là n và m. Viết chương trình nối mảng B vào cuối mảng A.

```
void NoiMang(int A[], int &n, int B[], int m)
{
    for(int i=0; i<m; i++)
          A[n+i]=B[i];
    n=n+m;
}
```

5.2.11.2. Ghép xen kẽ(đan xen)

Cho 2 mảng số nguyên A và B kích thước lần lượt là na và nb đều đã được sắp xếp tăng dần. Viết hàm nối 2 mảng A và B vào mảng C sao cho mảng C cũng được sắp xếp tăng dần.

Cách thực hiện:

B1: Đưa lần lượt từng phần tử của mảng A và mảng B vào mảng C, tăng chỉ số tương ứng.

B2: Nếu một trong hai mảng hết trước thì chép tất cả các phần tử còn lại của mảng chưa hết vào mảng C.

*Đặt **ia** là chỉ số của mảng A; **ib**: chỉ số của mảng B và **ic** là chỉ số của mảng c.*

```
void NoiMang(int A[],int na,int B[],int nb,int C[],int &nc)
{
    int ia=0, ib=0, ic=0;
    while(ia<na && ib<nb)
    {
        if (A[ia]<B[ib])
            C[ic++]=A[ia++];
        else
          C[ic++]=B[ib++];
    }
    //nếu mảng A còn thì đưa hết vào C
    while(ia<na)
        C[nc++]=A[ia++];
    //nếu mảng B còn thì đưa hết vào C
    while(ib<nb)
        C[nc++]=B[ib++];
}
```

5.3. BÀI TẬP

5.3.1. *THAO TÁC TRÊN MỘT MẢNG*

5.3.1.1. Nhập mảng

5.3.1.1.1. Nhập giá trị các phần tử của mảng từ bàn phím

1/- Tạo mảng A gồm n phần tử (n>0), với yêu cầu giá trị của các phần tử của mảng thỏa điều kiện đều là số dương:

✍ *Mở rộng*:

- Các trường hợp số âm, số chẵn, số vừa âm vừa lẻ, …
- Nằm trong khoảng từ 50 đến 100.
- Nhỏ hơn 10 hoặc lớn hơn hay bằng 50.

2/- Tạo mảng A gồm n phần tử (n>0), với yêu cầu những vị trí (hay chỉ số mảng) là số lẻ chỉ nhận giá trị nhập vào là số lẻ, và những vị trí (hay chỉ số mảng) là số chẵn chỉ nhận giá trị nhập vào là số chẵn.

✍ *Mở rộng* cho trường hợp ngược lại: vị trí lẻ chỉ nhận số chẵn; vị trí chẵn chỉ nhận số lẻ.

3/- Tạo mảng A gồm n phần tử (n>0), với yêu cầu giá trị của phần tử nhập sau trong mảng phải lớn hơn hoặc bằng phần tử liền trước (sau khi nhập hoàn tất, ta thu được mảng được sắp xếp tăng dần).

VD1: 2 4 7 7 8 11 ⇨Hợp lệ

VD2: 2 4 ***3*** ⇨không hợp lệ⇨yêu cầu nhập lại

4/- Tạo mảng A gồm n phần tử (n>0), với yêu cầu chỉ cho người dùng nhập giá trị của các phần tử của mảng là số nguyên tố

✍ *Mở rộng* cho các trường hợp số hoàn thiện, số chính phương, …

5.3.1.1.2. Tạo mảng với giá trị của các phần tử được phát sinh ngẫu nhiên

5/- Lần lượt viết các hàm phát sinh ngẫu nhiên mảng 1 chiều các số nguyên dương gồm n phần tử trong từng trường hợp sau (mỗi hàm xử lý cho 1 trường hợp):

a. Với 5<n<200, và giá trị của các phần tử trong khoảng từ 0 đến 99.

b. Với 5<n<50, và cho giá trị của các phần tử trong khoảng từ -100 đến 100.

c. Với 5<n<50, và các số nguyên sao cho giá trị phát sinh ngẫu nhiên phải toàn là số lẻ.

✍ *Mở rộng* cho các trường hợp số chẵn, số vừa âm vừa lẻ, số nguyên tố, số hoàn thiện, số hoàn hảo, …

6/- Viết chương trình phát sinh ngẫu nhiên mảng 1 chiều các số nguyên gồm n phần tử, sao cho số phát sinh sau phải lớn hơn hay bằng số phát sinh liền trước đó (các phần tử của mảng được sắp xếp tăng dần).

✍ *Mở rộng* cho trường hợp số giảm dần.

7/- Viết chương trình phát sinh ngẫu nhiên mảng 1 chiều các số nguyên gồm n phần tử, với yêu cầu những vị trí (hay chỉ số mảng) là số lẻ chỉ nhận giá trị nhập vào là số lẻ, và những vị trí (hay chỉ số mảng) là số chẵn chỉ nhận giá trị nhập vào là số chẵn.

✍ *Mở rộng* cho trường hợp ngược lại: vị trí lẻ chỉ nhận số chẵn; vị trí chẵn chỉ nhận số lẻ.

5.3.1.2. Xuất mảng

8/- Xuất toàn bộ các phần tử có trong mảng.

5.3.1.2.1. Xuất theo giá trị phần tử có trong mảng

9/- Liệt kê các phần tử trong mảng có giá trị lẻ.
✍ *Mở rộng*: giá trị chẵn, giá trị âm, giá trị dương, …

10/- Liệt kê các phần tử trong mảng có giá trị là số chẵn và nhỏ hơn 20.
✍ *Mở rộng*: giá trị chẵn và là số âm, giá trị lẻ và là số dương, …

11/- Viết hàm (ngoài các tham số cần thiết) nhận tham số là số x, và in ra tất cả các số có giá trị lớn hơn x.
✍ *Mở rộng*: khác X, nhỏ hơn hay bằng X, bội số của X, ước số của X, …

12/- Liệt kê các giá trị là số nguyên tố có trong mảng.
✍ *Mở rộng* cho các trường hợp số hoàn thiện, số hoàn hảo, …

13/- Liệt kê các số trong mảng có giá trị thuộc [x,y] cho trước (x và y là tham số của hàm)
✍ *Mở rộng*: liệt kê các số chẵn trong mảng nguyên thuộc [x,y].

5.3.1.2.2. Xuất theo vị trí (hay chỉ số) của mảng

14/- Liệt kê các phần tử nằm tại vị trí lẻ.
✍ *Mở rộng* cho trường hợp: vị trí chẵn, vị trí là số hoàn thiện, số hoàn hảo, …

15/- Liệt kê các phần tử trong mảng có vị trí là số chẵn và nhỏ hơn 20.
✍ *Mở rộng* cho trường hợp: vị trí từ p đến q (thay vì từ 0 đến n-1 như thường dùng), vị trí từ 0 đến p hoặc vị trí từ q đến n-1, …

16/- Viết hàm (ngoài các tham số cần thiết) nhận tham số là số x, và in ra tất cả các số có vị trí lớn hơn x.
✍ *Mở rộng*: khác X, nhỏ hơn hay bằng X, bội số của X, ước số của X, …).

5.3.1.2.3. Xuất dựa trên kết hợp giữa vị trí và giá trị

17/- In các phần tử có giá trị là số nguyên tố nằm tại những vị trí chẵn trong mảng.

18/- * Liệt kê tất cả các cặp giá trị (a,b) trong mảng thỏa điều kiện a>=b và vị trí (chỉ số) chứa số a < vị trí (chỉ số) chứa số b.

19/- Viết hàm nhận tham số vào 1 số K (với -999<=K<=999). In ra cách đọc chữ số tương ứng (sử dụng mảng chứa các chuỗi "không", "một", "hai", …).
Ví dụ: nhập -456 in ra: Am bon nam sau.
✍ *Mở rộng*: cho trường hợp giá trị của K<= ± 2.000.000.000.

20/- Viết chương trình nhập vào năm. In ra tên của năm âm lịch tương ứng.
Ví dụ: nhập 1999 in ra Kỷ Mão.

Biết rằng:

CAN	*Giáp*	*At*	*Bính*	*Đinh*	*Mậu*	*Kỷ*	*Canh*	*Tân*	*Nhâm*	*Quý*	
CHI	*Tý*	*Sửu*	*Dần*	*Mão*	*Thìn*	*Tỵ*	*Ngọ*	*Mùi*	*Thân*	*Dậu*	*Tuất*	*Hợi*

5.3.1.2.4. Các trường hợp xuất khác

21/- Mỗi yêu cầu sau đây được viết thành một hàm riêng biệt:

a)- In mảng vừa tạo thành 2 dòng:
- ✧ Dòng 1: các số lẻ.
- ✧ Dòng 2: các số chẵn.

b)- In mảng vừa tạo thành 2 dòng:
- ✧ Dòng 1: các số âm
- ✧ Dòng 2: các số dương.

c)- Vẽ biểu đồ ngang theo giá trị của các số có trong array.

Ví du: array có 3 số 2, 7, 4 in ra:
```
x x
x x x x x x x
x x x x
```

d)- Vẽ biểu đồ đứng theo giá trị của các số có trong array.

Ví du: array có 3 số 2, 0, 4, 1 sẽ in ra:
```
      x
      x
x     x
x     x     x
```

5.3.1.3. Kỹ thuật đặt cờ hiệu (kết quả trả về là có/không thỏa điều kiện cần kiểm tra)

5.3.1.3.1. Kiểm tra mảng có tồn tại giá trị thỏa điều kiện cho trước

22/- Kiểm tra mảng có chứa giá trị 0 hay không? Có trả về 1, không có trả về 0.

✍ *Mở rộng*:
- Mảng có chứa số âm, số nguyên tố, số hoàn thiện, …
- Thay giá trị 0 bằng tham số X. Kiểm tra xem mảng có chứa giá trị nhỏ hơn hay bằng X, bội số của X, ước số của X, …).

23/- Liệt kê các số nguyên tố nhỏ hơn giá trị của tham số X, nếu mảng không tồn tại số nguyên tố nào nhỏ hơn X thì phải xuất ra một câu thông báo (VD: mảng không chứa số nguyên tố nhỏ hơn X).

✍ *Mở rộng* cho các trường hợp số hoàn thiện, số chính phương, …

5.3.1.3.2. Kiểm tra tất cả các phần tử của mảng thỏa điều kiện

24/- Kiểm tra mảng một chiều chứa toàn số âm hay không? Có trả về true, không có trả về false.

✍ *Mở rộng*: cho mảng toàn dương, mảng toàn chẵn, mảng toàn lẻ, mảng đồng nhất, mảng toàn số nguyên tố, …

25/- Kiểm tra xem mảng đã cho có phải là mảng tăng hay không? (mảng tăng là mảng có các phần tử sau luôn lớn hơn hay bằng phần tử trước nó)..

✍ *Mở rộng*: trường hợp mảng giảm, mảng đối xứng, mảng cấp số cộng, …

5.3.1.4. Kỹ thuật đặt lính canh (áp dụng cho dạng bài toán "tìm kiếm một giá trị", "tìm kiếm một vị trí")

26/- Tìm vị trí của phần tử đầu tiên có giá trị bằng x. Nếu trong mảng không chứa giá trị x, hàm trả về giá trị -1.

✍ *Mở rộng*:
- Tìm vị trí cuối cùng phần tử có giá trị x.

- Tìm vị trí đầu tiên chứa giá trị nhỏ hơn x, vị trí cuối cùng chứa giá trị nhỏ hơn x.
- Tìm vị trí phần tử có giá trị x xuất hiện cuối cùng trong mảng.

27/- Viết hàm tìm vị trí của phần tử số âm đầu tiên có trong mảng. Nếu trong mảng không có số âm, hàm trả về giá trị -1.

✍ *Mở rộng*:
- Tìm vị trí của phần tử có giá trị là số lẻ/chẵn đầu tiên
- tìm vị trí của phần tử lớn nhất trong mảng
- Tìm vị trí các phần tử nguyên tố trong mảng
- In vị trí các phần tử là số nguyên tố có giá trị lớn hơn 23.
- Tìm số chính phương đầu tiên trong mảng.
- Trả về vị trí cuối cùng (thay vì tìm vị trí đầu tiên).

28/- Viết hàm tìm vị trí của phần tử nhỏ nhất trong mảng. Nếu có nhiều giá trị cùng nhỏ nhất, hàm trả về vị trí đầu tiên của giá trị nhỏ nhất.

✍ *Mở rộng*: Tìm vị *trí* của phần tử lớn nhất trong mảng

29/- Viết hàm tìm vị trí của phần tử Am lớn nhất đầu tiên trong mảng. *Nếu trong mảng không có số âm, hàm trả về giá trị -1.*

✍ *Mở rộng*:
- Tìm vị trí của phần tử có giá trị dương nhỏ nhất.
- Tìm vị trí các phần tử nguyên tố lớn/nhỏ nhất trong mảng
- Tìm vị trí chứa số chính phương lớn/nhỏ nhất trong mảng.
- Tìm vị trí đầu tiên của phần tử có giá trị là số hoàn thiện, số chính phương, số Armstrong.
- Tìm vị trí đầu tiên của phần tử có giá trị là số hoàn thiện nhỏ nhất, số chính phương nhỏ nhất, số Armstrong nhỏ nhất.

30/- Tìm giá trị của phần tử trong mảng "xa giá trị x nhất"

✍ *Mở rộng*:
- Tìm giá trị của phần tử trong mảng "gần giá trị x nhất".
- Tìm cả giá trị và vị trí của phần tử trong mảng "xa/gần giá trị x nhất".

31/- Tìm đoạn [a,b] sao cho đoạn này chứa tất cả các giá trị trong mảng

32/- Viết hàm nhận tham số là mảng các số nguyên (A), số lượng phần tử của mảng (n). Tìm giá trị x sao cho đoạn [-x,x] chứa tất cả các giá trị trong mảng.

33/- Tìm giá trị đầu tiên nằm trong khỏang (x,y) cho trước. Nếu không có trả về giá trị -1.

34/- Tìm giá trị đầu tiên trong mảng có dạng 2^k. Nếu không có thì trả về giá trị -1.

✍ *Mở rộng*: Tìm giá trị đầu tiên trong mảng có dạng 3^k, 4^k,…

35/- Tìm số chẵn lớn nhất nhỏ hơn mọi giá trị lẻ có trong mảng nguyên

36/- Tìm số nguyên tố nhỏ nhất lớn hơn mọi giá trị còn lại (không phải số nguyên tố) trong mảng.

37/- Tìm ước chung lớn nhất của tất cả phần tử trong mảng nguyên.

38/- Tìm bội số chung nhỏ nhất cho tất cả các phần tử trong mảng nguyên.

39/- Tìm vị trí trong mảng số nguyên thỏa điều kiện giá trị tại vị trí đó lớn hơn giá trị có trong 2 vị trí liền kề. Nếu không có trả về -1. Bỏ qua (không xét) vị trí đầu và cuối mảng.

✍ *Mở rộng*:

- Giá trị tại vị trí đó bằng tổng 2 giá trị có trong vị trí kế cận (A[i]= A[i-1]+A[i+1]).
- Giá trị tại vị trí đó bằng tích 2 giá trị có trong vị trí kế cận (A[i]= A[i-1]*A[i+1]).
- Xét cả 2 vị trí đầu và cuối mảng (chỉ có 1 phần tử kế cận).

40/- Hãy tìm giá trị đầu tiên trong mảng một chiều các số nguyên có chữ số đầu tiên là chữ số lẻ. Nếu trong mảng không tồn tại giá trị như vậy hàm sẽ trả về giá trị 0 (ví dụ: 110)

41/- Tìm giá trị toàn là chữ số lẻ và lớn nhất trong những số thỏa điều kiện. Không có trả về -1.

42/- Cho mảng một chiều các số nguyên hãy viết hàm tìm giá trị đầu tiên thỏa tính chất số gánh (ví dụ giá trị 12321).

43/- Tìm 1 giá trị có số lần xuất hiện nhiều nhất trong mảng

44/- Tìm chữ số xuất hiện nhiều nhất trong mảng

VD: mảng gồm 3 phần tử 15, 42, 14. Chữ số xuất hiện nhiều nhất là chữ số 1 và chữ số 4.

✍ *Mở rộng*: Tìm *chữ* số xuất hiện ít nhất trong mảng

45/- Tìm 2 giá trị gần nhau nhất trong mảng

46/- Viết hàm nhận tham số là mảng các số nguyên (A), số lượng phần tử của mảng (n) và số nguyên x. Tìm giá trị trong mảng các số nguyên "xa giá trị x nhất" (xanhat)

Ví dụ: cho mảng A

24	45	23	13	43	-12

Với giá trị x = 15, Khoảng cách từ x tới các phần tử khác trong mảng là:

9	30	8	2	28	27

Giá trị trong mảng xa giá trị x nhất là: 45

✍ *Mở rộng*: Tìm phần tử đầu tiên trong mảng "gần giá trị x nhất".

Ví dụ: cho mảng A

24	45	23	13	43	-12

Với giá trị x = 15, Khoảng cách từ x tới các phần tử khác trong mảng là:

9	30	8	2	28	27

Giá trị trong mảng gần giá trị x nhất là: 13

5.3.1.5. Kỹ thuật đếm / tính tổng / tính trung bình

47/- Viết hàm tính tổng các phần tử nằm ở vị trí chẵn trong mảng.

✍ *Mở rộng*: Viết hàm tính tổng các phần tử nằm ở vị trí chia chẵn cho 5, vị trí là số nguyên tố.

48/- Tổng các phần tử có chữ số đầu là chữ số lẻ

✍ *Mở rộng*:

- Tổng các phần tử có chữ số đầu là chẵn,…
- Tổng các phần tử có chữ số hàng chục là 5

49/- Tổng các phần tử lớn hơn phần tử đứng liền trước nó

50/- Tổng các phần tử lớn hơn trị tuyệt đối của phần tử liền sau nó

51/- Tổng các phần tử lớn hơn phần tử liền kề

52/- Tổng các phần tử đối xứng

53/- Viết hàm tính tổng của từng dãy con giảm có trong mảng.

54/- Tính tổng các phần tử cực đại trong mảng các số nguyên (phần tử cực *đại là phần tử lớn hơn các phần tử liền kề)*.

Ví dụ: *1 **5** 2 **6** 3 **5** 1 **8** 6 ⇒ in ra 24*

✍ *Mở rộng*: Tính tổng các phần tử cực tiểu trong mảng.

55/- Viết hàm tính giá trị trung bình của các số hoàn thiện trong mảng.

✍ *Mở rộng*:

- Tính giá trị trung bình của các phần tử có giá trị âm, các phần tử có giá trị là số lẻ/số chẵn,…
- Tính giá trị trung bình của các phần tử là bội của 3 và 5 trong mảng.
- Tính giá trị trung bình của các phần tử có giá trị là số nguyên tố, là số hoàn thiện, là số chính phương, …

5.3.1.6. Đếm

56/- Đếm số lần xuất hiện của giá trị x trong mảng.

✍ *Mở rộng*:

- Đếm số lần xuất hiện của giá trị dương, giá trị âm, số chẵn, số lẻ, bội số của 5, bội số của X, ước số của 7, ước số của X, …
- Đếm số lần xuất hiện của giá trị là số nguyên tố, là số hoàn thiện, là số chính phương, số Armstrong, …

57/- Cho biết sự tương quan giữa số lượng chẵn và lẻ trong mảng (bao nhiêu phần trăm số lẻ, bao nhiêu phần trăm là số chẵn)

58/- Đếm các phần tử có chữ số đầu là chữ số lẻ

✍ *Mở rộng*:

- Đếm các phần tử có chữ số đầu là chẵn, …
- Đếm các phần tử có chữ số hàng chục là 5
- Đếm các phần tử có chữ số hàng đơn vị là 1 trong các số 3, 6, 9

59/- Đếm số đối xứng trong mảng

60/- Đếm số lượng phần tử lớn hơn các phần tử liền kề, thực hiện cho 2 trường hợp:

- KHÔNG xét 2 phần tử đầu và cuối mảng vì không đủ 2 phần tử liền kề).
- CÓ xét 2 phần tử đầu và cuối mảng (2 phần tử này chỉ có 1 phần tử liền kề trước hoặc sau).

✍ *Mở rộng*:

- Đếm số lượng phần tử kề nhau mà cả 2 trái dấu.
- Đếm số lượng phần tử kề nhau mà cả 2 đều chẵn.
- Đếm số lượng phần tử kề nhau mà số đứng sau cùng dấu số đứng trước và có giá trị tuyệt đối lớn hơn.

5.3.1.7. Trung bình

61/- Trung bình cộng các số dương

✍ *Mở rộng*:

- Trung bình cộng các số âm, số chẵn, số lẻ.

- Trung bình cộng các số lớn hơn x, nhỏ hơn x
- Trung bình cộng các số nguyên tố, các số hoàn thiện, các số chính phương, …
- Trung bình nhân các số dương, số âm, số chẵn, số lẻ

62/- (*) Khoảng cách trung bình giữa các giá trị trong mảng

5.3.1.8. Kỹ thuật thêm

63/- Chèn thêm giá trị x vào vị trí k trong mảng một chiều nguyên (x, k là tham số đầu vào của hàm).

64/- Giả sử các phần tử trong mảng đã được sắp xếp tăng dần. Viết hàm thêm giá trị x vào mảng sao cho mảng vẫn tăng dần (không sắp xếp).

65/- Thêm giá trị y vào sau các phần tử có giá trị x trong mảng một chiều nguyên (x, y là tham số đầu vào của hàm).

66/- Viết hàm chèn phần tử có giá trị X vào vị trí đầu tiên của mảng.

67/- Viết hàm chèn phần tử có giá trị X vào phía sau phần tử có giá trị lớn nhất trong mảng.

68/- Viết hàm chèn phần tử có giá trị X vào trước phần tử có giá trị là số nguyên tố đầu tiên trong mảng.

69/- Viết hàm chèn phần tử có giá trị X vào phía sau tất cả các phần tử có giá trị chẵn trong mảng.

5.3.1.9. Kỹ thuật xóa

70/- Xóa phần tử có chỉ số k trong mảng một chiều nguyên.

✍ *Mở rộng*: chỉ số là số chẵn, số lẻ, số nguyên tố, bội số của m, …

71/- Xóa tất cả phần tử có giá trị bằng X.

✍ *Mở rộng*: giá trị nhỏ hơn / lớn hơn X, số chẵn, số lẻ, số âm, giá trị là số nguyên tố …...

72/- Xóa tất cả các số (dương) lớn nhất trong mảng một chiều nguyên.

✍ *Mở rộng*: cho các giá trị âm lớn nhất, chẵn lớn nhất, lẻ lớn nhất, nguyên tố lớn nhất, bội số lớn nhất (hay ước số lớn nhất) của số k, …

73/- Xóa tất cả các phần tử có giá trị xuất hiện nhiều hơn một lần trong mảng một chiều nguyên.

74/- Xoá phần tử tại vị trí lẻ trong mảng.

75/- Nhập vào giá trị X. Viết hàm xoá phần tử có giá trị gần X nhất.

5.3.1.10. Kỹ thuật xử lý mảng

76/- Đảo ngược thứ tự mảng một chiều các số nguyên.

77/- Đảo ngược thứ tự các giá trị chẵn trong mảng một chiều nguyên.

78/- Chuyển các phần tử có giá trị chẵn về đầu mảng.

79/- Chuyển các phần tử có giá trị âm về cuối mảng.

80/- Dịch trái xoay vòng 1 lần trên mảng một chiều.

✍ *Mở rộng*:

- Dịch phải xoay vòng 1 lần trên mảng một chiều.
- Dịch phải/trái k lần; …

5.3.1.11. Sắp xếp mảng

81/- Sắp xếp bằng phương pháp đổi chỗ trực tiếp.

82/- Sắp xếp sao cho số âm giảm dần, số dương tăng dần.

83/- Sắp xếp các phần tử chẵn nằm bên trái theo thứ tự tăng dần còn các phần tử lẻ bên phải theo thứ tự giảm dần.

84/- Sắp xếp các phần tử âm giảm dần từ trái sang phải, phần tử dương tăng dần từ phải sang trái.

85/- Sắp xếp sao cho số lẻ tăng dần, số dương giữ nguyên vị trí.

✍ *Mở rộng*: Sắp xếp mảng theo thứ tự tăng dần của các phần tử là số nguyên tố, số chính phương (các phần tử khác giữ nguyên vị trí)

5.3.1.12. Kỹ thuật mảng con

86/- Liệt kê các mảng con (> 2 phần tử) tăng trong mảng một chiều.

✍ *Mở rộng*:

- Liệt kê các mảng con toàn dương.
- Liệt kê mảng con tăng có tổng giá trị các phần tử là lớn nhất.
- Liệt kê mảng con tăng có số lượng phần tử nhiều nhất (dài nhất). Nếu có 2 dãy con dài bằng nhau thì xuất mảng con đầu tiên.

Ví dụ: Nhập mảng 1 4 2 3 1 2 6 8 3 5 7

Mảng con dài nhất là: 1 2 6 8

📖 *Hướng dẫn:*

- Khởi động các biến DauMax, CuoiMax, DauMoi, CuoiMoi = vị trí đầu mảng, DemMax, DemMoi = 1.
- Duyệt mảng:

Nếu còn tăng và chưa hết mảng thì (CuoiMoi = vị trí đang xét và DemMoi tăng 1)
Ngược lại thì so sánh:
Nếu DemMoi > DemMax
thì (DauMax = DauMoi và CuoiMax = CuoiMoi).

87/- Cho 2 mảng A, B các số nguyên (kích thước mảng A nhỏ hơn mảng B). Hãy kiểm tra xem A có phải là con của B hay không?

88/- (*) Viết chương trình tính trung bình cộng của các tổng các mảng tăng dần có trong mảng các số nguyên.

Ví dụ: *1 2 3 4 2 3 4 5 6 4 5 6 => TB = 15.*

5.3.1.13. Tổng hợp

89/- Nhập vào một mảng số thực kết thúc bằng việc nhập số 0 (số 0 không lưu vào mảng) hoặc khi đã nhập đủ 20 số. Kiểm tra có hay không các tính chất sau đây của mảng:

a. Mảng đơn nhất? (Không có phần tử trùng nhau trong mảng)

b. Mảng đan dấu? (2 phần tử kề nhau phải khác đấu. Mảng 1 phần tử xem như đan dấu)

c. Mảng tuần hoàn? (Mảng tuần hoàn: Nếu A_i, A_{i+1}, A_{i+2} là 3 phần tử liên tiếp trong mảng thì: $A_{i+1} \geq A_i$ và $A_{i+1} \geq A_{i+2}$ hoặc $A_{i+1} \leq A_i$ và $A_{i+1} \leq A_{i+2}$. Mảng có 2 phần tử xem như tuần hoàn).

d. Mảng tăng dần? Mảng giảm dần?

90/- Nhập vào một mảng số nguyên gồm n phần tử. Tạo menu và thực hiện các thao tác trên mảng:

a. Xuất mảng ra màn hình.
b. Đếm số phần tử là bội số của 3 của mảng (đếm số phần tử có điều kiện).
c. Tìm phần tử lớn nhất, nhỏ nhất và tính tổng, tích các phần tử của mảng.
d. Thêm vào mảng phần tử x tại vị trí k (k<n).
e. Xóa phần tử tại vị trí k (k<n).
f. Tìm phần tử x trong mảng, chỉ ra vị trí xuất hiện của x.
g. Tìm cặp phần tử có tổng bình phương đúng bằng k (k nhập từ bàn phím).
h. Sắp xếp mảng tăng dần.
i. Sắp xếp các phần tử ở vị trí chẵn tăng dần, vị trí lẻ giảm dần.

91/- Nhập mảng số nguyên a có n phần tử (0< n <=20). Tạo menu để thực hiện các công việc:

a. Sắp các phần tử vị trí lẻ tăng dần, các phần tử vị trí chẵn giảm dần
b. Sắp các phần tử dương về đầu mảng có thứ tự giảm dần, các phần tử âm cuối mảng có thứ tự tăng dần.
c. Sắp các số nguyên tố về đầu mảng có thứ tự tăng dần, các phần tử còn lại có thứ tự giảm dần.

92/- Viết hàm liệt kê các bộ 4 số a, b, c, d trong mảng các số nguyên (có ít nhất 4 phần tử và đôi một khác nhau) sao cho a + b = c + d.

93/- (*) Cho mảng các số nguyên a gồm n phần tử (n<=30000) và số dương k (k<=n). Hãy chỉ ra số hạng lớn thứ k của mảng.

Ví dụ: Mảng a: 6 3 1 10 11 18
k = 3
Kết quả: 10

94/- Viết chương trình tính tổng tất cả các phần tử xung quanh trên mảng các số nguyên. Biết rằng *Phần tử xung quanh là hai phần tử bên cạnh cộng lai bằng chính nó; v*í dụ: 1 3 2 ➔ 1,2 là hai phần tử xung quanh của 3).

Ví dụ: *1 **3** 2 **5** 3 **9** 6 ➔ tổng 17*

95/- (**) Viết chương trình nhập vào hai số lớn a, b nguyên (a, b có từ 20 chữ số trở lên). Tính tổng, hiệu, tích, thương của hai số trên.

96/- Viết hàm tìm và xóa tất cả các phần tử trùng với x trong mảng một chiều các số nguyên, nếu không tồn tại phần tử x trong mảng thì trả về -1.

97/- (**) Viết hàm xoá những phần tử sao cho mảng kết quả có thứ tự tăng dần và số lần xoá là ít nhất.

98/- Cho mảng a gồm n số nguyên có thứ tự tăng dần. Nhập vào một phần tử nguyên X, viết hàm chèn X vào mảng sao cho mảng vẫn có thứ tự tăng dần (không sắp xếp).

99/- Viết chương trình tìm số lẻ nhỏ nhất lớn hơn mọi số chẵn có trong mảng.

100/- Viết hàm tìm giá trị chẵn nhỏ nhất nhỏ hơn mọi giá trị lẻ trong mảng các số nguyên.

101/- Viết hàm tìm phần tử xuất hiện nhiều nhất trong mảng các số nguyên.

102/- Viết chương trình đếm và liệt kê các mảng con tăng dần trong mảng một chiều các số nguyên.

Ví dụ: *6 5 3* ***2 3 4*** *2 7 các mảng con tăng dần là 2 3 4 và 2 7*

103/- Viết chương trình tìm mảng con tăng dần có tổng lớn nhất trong mảng một chiều.

104/- (*) Viết chương trình nhập vào một mảng số a gồm n số nguyên (n <= 100). Tìm và in ra mảng con tăng dài nhất

Ví dụ: Nhập mảng a: **1 2 3 6** 4 7 8 **3 4 5 6 7 8 9** 4 5
Mảng con tăng dài nhất: 3 4 5 6 7 8 9

105/- Viết chương trình nhập vào mảng số a gồm n số nguyên (n <= 100).

a. Hãy đảo ngược mảng đó.

Ví dụ: *Nhập a: 3 4 5 2 0 4 1*
Mảng sau khi đảo: 1 4 0 2 5 4 3

b. (*) Hãy kiểm tra xem mảng đã cho có thứ tự chưa (mảng được gọi là thứ tự khi là mảng tăng hoặc mảng giảm).

106/- Cho mảng A có n phần tử hãy cho biết mảng này có đối xứng hay không.

107/- Cho mảng A có n phần tử. Nhập vào số nguyên k (k >= 0), dịch phải xoay vòng mảng A k lần.

Ví dụ: *Mảng A: 5 7 2 3 1 9*
Nhập k = 2
Dịch phải xoay vòng mảng A:1 9 5 7 2 3

108/- (**) Viết chương trình in ra tam giác Pascal (dùng mảng một chiều).

109/- Tìm giá trị trong mảng các số thực " xa giá trị x nhất"

Ví dụ:

24	45	23	13	43	-12

Giá trị x: 15

Khoảng cách từ x = 15 tới các phần tử khác trong mảng là:

9	30	8	2	28	27

Giá trị trong mảng xa giá trị x nhất là: 45

110/- Tìm một vị trí trong mảng một chiều các số thực mà tại vị trí đó là giá trị "gần giá trị x nhất ".

Ví dụ:

24	45	23	13	43	-12

Giá trị x: 15

Khoảng cách từ x = 15 tới các phần tử khác trong mảng là:

9	30	8	2	28	27

Giá trị trong mảng gần giá trị x nhất là: 13

5.3.2. *THAO TÁC TRÊN NHIỀU MẢNG*

5.3.2.1. Nhập xuất mảng

111/- Cho mảng một chiều nguyên a. Viết hàm tạo mảng b từ a sao cho b chỉ chứa các giá trị lẻ của a.

✍ *Mở rộng*: cho số nguyên tố, số hoàn thiện, số chính phương, …

112/- Cho mảng một chiều nguyên a. Viết hàm tạo mảng b từ a sao cho b chứa vị trí của các phần tử có giá trị lớn nhất trong a.

113/- Viết chương trình cho phép người dùng nhập vào mảng A các số nguyên gồm n phần tử (1<n<15). Hãy tạo mảng B sao cho:

B[i] = (2*A[i] -1)nếu A_i>0
B[i] = (-2*A[i] +1) nếu A_i<0
B[i] = 1 nếu A[i]=0

114/- Cho mảng một chiều nguyên a. Viết hàm tạo mảng b từ a sao cho b[i]= tổng các phần tử lân cận với A[i] trong a.

5.3.2.2. Nhập / tách mảng

115/- Cho mảng số nguyên A có n phần tử. Tách A thành 2 mảng: B chứa các số lẻ, C chứa các số chẵn.

Ví dụ: Mảng A : 1 3 **8 2** 7 5 9 0 **10**
Mảng B : 1 3 7 5 9
Mảng C : 8 2 10

116/- Có hai mảng một chiều A, B. Hai mảng này đã được sắp xếp tăng dần.Hãy viết chương trình trộn hai mảng A và B lại để được mảng C có các phần tử tăng dần.

117/- Cho 2 mảng số nguyên a và b kích thước lần lượt là n và m. Viết chương trình nối 2 mảng trên thành mảng c theo nguyên tắc chẵn ở đầu mảng và lẻ ở cuối mảng.

Ví dụ: Mảng a : 3 2 7 5 9
Mảng b : 1 8 10 4 12 6
Mảng c : **2 8 10 4 12 6** 3 7 5 9 1

118/- Cho 2 mảng số nguyên A và B kích thước lần lượt là n và m. Viết chương trình thực hiện:

a. Sắp xếp hai mảng A, B theo thứ tự tăng dần.
b. Trộn xen kẻ 2 mảng trên thành mảng c sao cho mảng c cũng có thứ tự tăng dần.
c. Sắp xếp lại để có mảng B giảm dần (mảng A vẫn tăng dần). Trộn 2 mảng A và B thành mảng C tăng dần.

5.3.2.3. Đếm – liệt kê

119/- Cho 2 mảng A, B. Đếm và liệt kê các phần tử chỉ xuất hiện 1 trong 2 mảng.

120/- Cho 2 mảng A, B. Đếm và liệt kê các phần tử chỉ xuất hiện trong mảng A nhưng không xuất hiện trong mảng B.

121/- Cho 2 mảng A, B. Đếm phần tử xuất hiện trong cả 2 mảng.

5.3.2.4. Khác

122/- Viết chương trình nhập vào 1 số K (với -999<=K<=999). In ra cách đọc chữ số tương ứng (sử dụng mảng).

Ví dụ: nhập -132 in ra: Am mot tram ba muoi hai.

123/- Cho nhập số nguyên dương n gồm tối đa 9 chữ số. In ra số lần xuất hiện của mỗi số

Ví dụ: với n=12712851. Sẽ xuất ra màn hình: *số 1 xuất hiện 3 lần*
số 2 xuất hiện 2 lần
số 5 xuất hiện 1 lần
số 7 xuất hiện 1 lần
số 8 xuất hiện 1 lần

124/- (**) Viết chương trình tách 1 mảng các số nguyên thành 2 mảng A và B. Không dùng sắp xếp, thực hiện sao cho kết quả thu được là:

- Mảng A chứa toàn số lẻ tăng dần.
- Mảng B chứa toàn số chẵn giảm dần.

Hướng dẫn: Tìm vị trí chèn thích hợp khi trích phần tử từ mảng ban đầu sang 2 mảng A và B.

Ví dụ: Mảng ban đầu : 9 3 **8 2** 7 5 1 0 **10**
Mảng A : 1 3 5 7 9
Mảng B : 10 8 2

125/- (*) Cho mảng C có n phần tử (n < 200), các phần tử là các chữ số trong hệ đếm cơ số 16 (Hexa) (điều kiện mỗi phần tử <= n). Hãy tách mảng C ra các mảng con theo điều kiện sau: các mảng con được giới hạn bởi hai lần xuất hiện của cùng 1 con số trong mảng.

Ví dụ1: **123**A45**1**8B**23** ➔ có các mảng con là123A451, 23A4518B2, 3A4518B23
Ví dụ2: **123**A45**3**8B**21** ➔ có các mảng con là 123A4538B21, 23A4518B2, 3A453
Ví dụ3: 123456789 ➔ in ra "Khong co day con".

126/- (**) Cho hai số nguyên dương A, B. Hãy xác định hai số C, D tạo thành từ hai số A, B sao cho C là số lớn nhất, D là số nhỏ nhất. Khi gạch đi một số chữ số trong C (D), thì các số còn lại giữ nguyên tạo thành A, các chữ số bỏ đi giữ nguyên tạo thành B.

Ví dụ: A = 52568, B = 462384 -> C = 54625682384, D = 45256236884.

127/- Đếm số lần xuất hiện của giá trị lớn nhất trong mảng một chiều.

Ví dụ: Mảng: 5 6 11 4 4 5 4
So lon nhat la 11.
So 11 xuat hien 1 lan

✍ *Mở rộng*:

- Đếm số lần xuất hiện của giá trị nhỏ nhất, dương nhỏ nhất, âm lớn nhất.
- Đếm số lần xuất hiện của số nguyên tố nhỏ nhất, …).

128/- Đếm số lượng các giá trị phân biệt có trong mảng

✍ *Mở rộng*: Liệt kê tần suất xuất hiện các giá trị xuất hiện trong mảng (mỗi giá trị xuất hiện bao nhiêu lần).

129/- Liệt kê các giá trị xuất hiện trong mảng một chiều nguyên đúng 1 lần.

✍ *Mở rộng*:

- Liệt kê các giá trị xuất hiện trong mảng một chiều nguyên đúng k lần, nhiều hơn 1 lần
- Liệt kê các giá trị có số lần xuất hiện nhiều nhất trong mảng.

130/- Tìm các số nguyên tố nhỏ hơn 1000 bằng giải thuật sàng Erastosthene (giải thuật sàng Erastosthene dùng phương pháp đánh dấu để loại bỏ những số không phải là số

nguyên tố. Giải thuật có từ một nhận xét rằng nếu k là số nguyên tố thì các số 2*k, 3*k,... n*k sẽ không là số nguyên tố (vì đã vi phạm định nghĩa về số nguyên tố).

131/- Viết chương trình nhập vào một mảng số a gồm n số thực (n ≤ 100), nhập vào mảng số b gồm m số thực (m ≤ 100). In ra những phần tử:

a. Chỉ xuất hiện trong mảng a mà không xuất hiện trong mảng b.

b. Xuất hiện ở cả hai mảng.

c. Không xuất hiện ở cả 2 mảng

d. Thực hiện lại yêu cầu *(i)* nhưng chỉ in mỗi giá trị thoả điều kiện 1 lần (do trên mảng có thể có các giá trị trùng nhau).

TÀI LIỆU THAM KHẢO

[1]. Hoàng Kiếm – Giải một bài toán trên máy tinh như thế nào – Tập 1 & Tập 2- Nhà Xuất Bản Giáo Dục -2001

[2]. Dương Anh Dức – Trần Hạnh Nhi – Giáo trình cấu trúc dữ liệu - Trường Đại Học Khoa Học Tự Nhiên Tp.Hồ Chí Minh – 2003

[3]. Trần Đan Thư – Giáo trình lập trình C – Tập 1& Tập 2 – Nhà Xuất Bản Đại Học Quốc Gia – 2001

[4]. Lê Hoài Bắc – Lê Hoàng Thái – Nguyễn Tấn Trần Minh Khang – Nguyễn Phương Thảo – Giáo trình lập trình C – Nhà Xuất Bản Đại Học Quốc Gia Tp Hồ Chí Minh – 2003

[5]. Nguyễn Thanh Hùng – Các bài tập tin học chọn lọc – Nhà Xuất Bản Giáo Dục – 1996.

[6]. Nguyễn Tiến Huy – Bài giảng Kỹ thuật lập trình – 2003.

[7]. Nguyễn Tấn Trần Minh Khang – Bài giảng Kỹ thuật lập trình – 2003

[8]. Bùi Việt Hà – Lập Trình Pascal – Nhà Xuất Bản Giáo Dục – 2001

[9]. Phạm Văn Ất – Kỹ thuật lập trình C cơ sở và nâng cao – Nhà Xuất Bản Khoa Học Và Kỹ Thuật – 1996.