BÀI TẬP LIÊN QUAN ĐẾN NHÓM LỆNH NHẬP XUẤT

Bài tập 1.1. Viết chương trình hiển thị Hello World! lên màn hình

```
Hello World!
--------------------------------
Process exited with return value 12
Press any key to continue . . .
```

Các bước thực hiện:

Bước 1. Khởi động Dev-C++

- Click Start - Dev-C++
  Hoặc D_Click shortcut Dev-C++ trên Desktop
- Màn hình giao diện Dev-C++ được hiển thị

Bước 2. Tạo chương trình (source)

- Click File – New – Source File (Ctrl + N)
- Tạo khung chương trình như sau:

Chương trình	**Giải thích**
#include <stdio.h> //chuong trinh chinh int main() { printf("Hello World!"); //ket thuc chuong trinh return 0; }	*Dòng 1:* Header file có tên stdio.h trong thư viện C chuẩn, định nghĩa 3 kiểu biến, một số macro và các hàm đa dạng để thực hiện input và output. Về cơ bản stdio- standard input output sẽ thực hiện các thao tác nhận ký tự từ bàn phím, xuất ra màn hình. *Dòng 2,6:* Ghi chú (comment) sẽ được bỏ qua khi biên dịch *Dòng 3:* Hàm main() sẽ được gọi với một số lượng đối số truyền vào là tùy ý và hàm sẽ trả về giá trị là một số nguyên Integer. Sau hàm main là cặp dấu ngoặc () quy định kiểu khai báo hàm. Kế đó: *Dòng 4,8:* là cặp dấu {} để xác định vùng/khối chương trình của hàm. *Dòng 5*: printf() gọi là hàm hiển thị dữ liệu có định dạng, nó cho phép hiển thị được tất cả các loại dữ liệu (các loại hằng, biến, biểu thức hoặc lời gọi hàm. Hàm printf("Hello World!") sẽ hiển thị cụm từ Hello World! ra màn hình. *Dòng 7:* Kiểu số nguyên có giá trị là 0 chính là kết quả trả về của hàm main() *Hoặc có thể sử dụng void main() thì không cần phải return một giá trị nào đó.*
Ngoài ra người dùng có thể sử dụng đoạn chương trình sau: #include <stdio.h> //chuong trinh chinh void main() { printf("Hello World!"); //ket thuc chuong trinh }	

Chú ý: Từng dấu chấm phẩy (;) ở cuối các câu lệnh printf("Hello World!") và return 0;
Dấu ngoặc kép "" cho cụm từ Hello World! trong hàm printf

Bước 3: Lưu tập tin (Save) với phần mở rộng (.c)

- File hoặc nhấn phím Ctrl + S
- Save as type: Drop_chọn "C source files (*.c)
- File name: Baitap1.c

Bước 4: Thực thi chương trình

- Bấm F9 để biên dịch (compile) chương trình: dịch sang mã máy để CPU vận hành
  Hoặc Click Execute – Compile F9
- Bấm F10 để thực thi tập tin thực thi (.exe) đã biên dịch ở bước trên

Sửa lỗi cú pháp trong Dev-C

Cũng trong chương trình trên, hãy xóa bỏ dấu chấm phẩy ; của câu lệnh printf(“Hello World!”) và biên dịch (F9) lại lần nữa. Hãy chú ý tới thông báo sau:

```
1  #include <stdio.h>
2  //chuong trinh chinh
3  int main()
4  {
5      printf("Hello World!")
6      //ket thuc chuong trinh
7      return 0;
8  }
```

Dòng 1: …Baitap1.c In function 'main':

Cho biết tên tập tin (chương trình nào gặp lỗi (trong thực tế một dự án (Project) có thể có nhiều tập tin (source)

Dòng 2: 7 [Error] expected ';' before 'return'

Tạm dịch: Dòng 7 [Lỗi] mong đợi dấu ; trước câu lệnh return

Thực tế có rất nhiều kiểu lỗi khác nhau, lập trình viên cần đọc kỹ các mã lỗi khi biên dịch chương trình, quá trình này gọi là bắt lỗi (debug). Có rất nhiều dạng lỗi khác nhau như lỗi cú pháp, lỗi khi thực hiện chương trình. Trong đó, lỗi cú pháp tương đối dễ debug, lập trình viên phải thực hiện theo đúng yêu cầu của hàm chức năng hoặc thư viện yêu cầu.

Hãy sửa lại chương trình trên (thêm dấu ; vào dòng 8) và thực thi lại chương trình

Ngoài ra, lập trình viên hãy xóa ký hiệu dấu ngoặc kép “ của chuỗi Hello World! và tiến hành phân tích thông báo lỗi và tiến hành debug.

Bài tập 1.2. Viết chương trình hiển thị tên và lớp học của bạn lên màn hình

```
Ho va ten: Hacker
Lop hoc: Tin hoc co ban A
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Xem chương trình mẫu sau:

```
1  #include <stdio.h>
2  //chuong trinh chinh
3  int main()
4  {
5      printf("Ho va ten: Hacker \n");
6      printf("Lop hoc: Tin hoc co ban A");
7      //ket thuc chuong trinh
8      return 0;
9  }
```

Để hiển thị nội dung trên 2 dòng màn hình, ta cần thêm vào ký hiệu kết thúc đoạn (newline hoặc return, có mã ASCII tương ứng là 10 và 13).

Sau đó sẽ hiển thị tiếp nội dung ở dòng tiếp theo của màn hình hiển thị.

Các ký tự điều khiển trong C được sử dụng trong hàm printf()

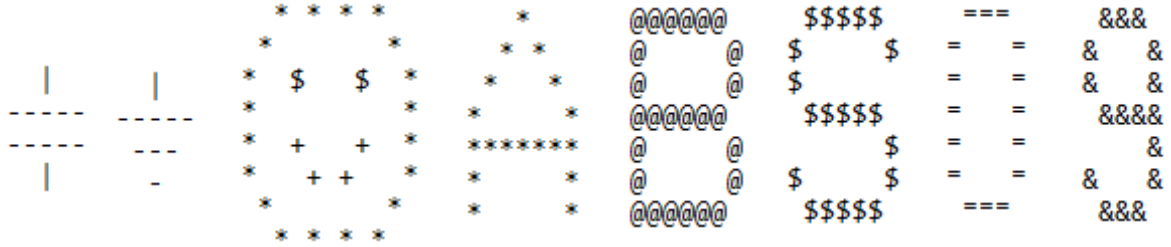
\n	Nhảy xuống dòng kế tiếp, trở về cột đầu tiên
\t	Tab ngang (khoảng cách dài hơn phím ký tự trắng (Space)
\r	Nhảy về đầu hàng, không xuống hàng (tương tự như chức năng Shift + Enter trong Word)
\a	Tiếng kêu bip (phát ra từ loa)
\\	In ra dấu \
\”	In ra dấu “
\’	In ra dấu ‘
%%	In ra dấu %

Bài tập tương tự

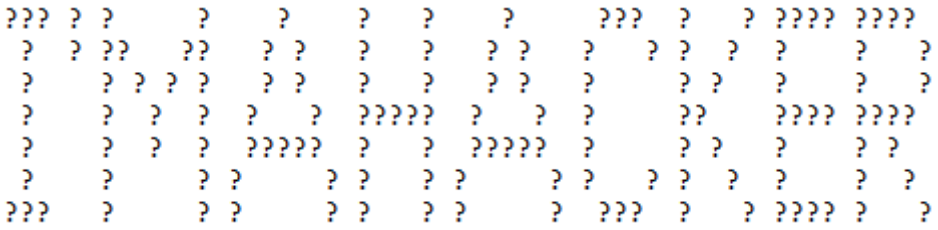
a. Viết chương trình hiển nội dung sau lên màn hình

```
Ho va ten: Hacker "Dai hoc Nha Trang"
Lop hoc: Tin hoc co ban A '2021
```

b. Viết chương trình tạo ký tự cho màn hình LCD 16x2. Biết rằng mỗi ký tự được hiển thị trên một lưới có kích thước là 8*8 pixel. Sau đây là một số ký tự không có sẵn trong thư viện của các phần mềm lập trình cho LCD hoặc xuất hiện trên bàn phím.



c. Viết chương trình tạo font chữ "I'M A HACKER" và hiển thị lên màn hình



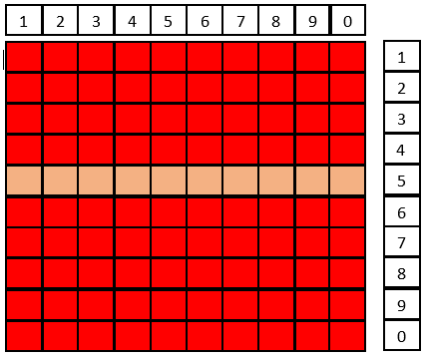
d. Trong các thiết bị cảm biến nhiệt độ, phần lớn các cảm biến này sẽ hiển thị giá trị nhiệt độ là Fahrenheit (độ F) hoặc Celsius (độ C). Hãy viết chương trình chuyển đổi từ độ C sang độ F và ngược lại. Ví dụ: 32^0C = 89.59998^0F và 100^0F = 37.7777778^0C

Công thức chuyển đổi là: $\frac{Celsius}{100} = \frac{Fahrenheit-32}{180}$

e. Viết chương trình đổ chỗ 2 số A và B (A,B là các số nguyên được nhập từ bàn phím) và tiến hành đổi chỗ 2 số này. Lưu ý là không được sử dụng biến tạm

```
Nhap so thu 1: 10
Nhap so thu 2: 20
So A va B sau khi doi cho la: 20 va 10
```

f. Có một lô gạch được sắp xếp như hình (có 10 hàng gạch, mỗi hàng có 1o viên gạch, trọng lượng mỗi viên theo quy định là 100gr). Thế nhưng thực tế thì có 1 hàng có trọng lượng mỗi viên chỉ là 90gr. Do đó, để xác định chất lượng đồng đều của gạch người ta tiến hành cân tổng số gạch nói trên. Nếu tổng số cân có được bé hơn số cân quy định thì sẽ biết được hàng nào là gạch kém chất lượng. Biết rằng, mỗi hàng gạch có chất lượng đồng nhất (hàng theo hàng). Hãy nêu cách làm và viết chương trình để sau 1 lần cân duy nhất giúp xác định hàng gạch kém chất lượng kể trên với R: số hàng, N: số viên gạch trên mỗi hàng, K: trọng lượng của viên gạch kém chất lượng.



Ví dụ: R=10, N=10, K=0.9, số hàng gạch có chất lượng kém là 5. Hãy đề xuất cách cân.

Bài tập 2.1. Viết chương trình nhập vào một số bất kỳ và hiển thị lên màn hình

Gợi ý:

Chương trình yêu cầu nhập một số nguyên từ bàn phím và hiển thị số đó lên màn hình. Việc này đòi hỏi 2 thao tác: (1) lưu trữ số vừa nhập vào bộ nhớ tạm; (2) hiển thị nội dung của ô nhớ tạm (lưu giữ số đã nhập) lên màn hình.

Chương trình không thể thực hiện việc hiển thị trực tiếp giá trị nhập từ bàn phím lên màn hình, đó là quy trình xử lý của CPU, hệ điều hành mà lập trình viên không thể can thiệp. Do đó, chúng ta cần xác định địa chỉ của ô nhớ cần sử dụng để lưu trữ giá trị được nhập từ bàn phím. Có 2 cách để thực hiện việc này: (1) Xác định địa chỉ cụ thể của vùng nhớ tạm, do đó lập trình viên cần biết rõ vị trí nào của vùng nhớ tạm đang còn trống để lưu trữ. Việc này đòi hỏi sự chính xác tuyệt đối vì nếu không xác định đúng địa chỉ ô nhớ thì giá trị được nhập sẽ lưu lên vùng nhớ đã có giá trị của một chương trình khác hoặc là vùng nhớ bảo tồn của hệ điều hành. Có thể nhận ra sự phức tạp của vấn đề ở đây, với các dòng vi điều khiển như Intel – Atmel 8051, AVR – Arduino, Microchip – PIC… số lượng vùng nhớ từ tạm (thanh ghi) khoảng 128 byte – 1KB nhưng đối với máy tính cá nhân (PC: Desktop, Laptop) thì số lượng ô nhớ vô cùng lớn 1GB (~1 tỷ byte) đến 8GB hoặc hơn; (2) dùng một biến lưu trữ tạm thời để lưu giữ giá trị (biến này được hệ điều hành cấp phát vùng nhớ một cách ngẫu nhiên). Vì vậy, cách tiếp cận duy nhất ở bài toán này là sử dụng biến để lưu trữ giá trị được nhập từ bàn phím. Xem đoạn chương trình mẫu sau:

```c
#include <stdio.h>
//khai bao bien
int number;
//chuong trinh chinh
int main()
{
    printf("Nhap 1 so nguyen tu ban phim:");
    scanf("%d",&number);
    printf("so vua nhap la: %d",number);
    //ket thuc chuong trinh
    return 0;
}
```

```
Nhap 1 so nguyen tu ban phim:2021
so vua nhap la: 2021
```

```c
#include <stdio.h>
//khai bao bien
float sothuc;
//chuong trinh nhap vao kieu so thuc (thap phan)
int main()
{
    printf("Nhap 1 so thuc tu ban phim:");
    scanf("%f",&sothuc);
    printf("so thuc vua nhap la: %f",sothuc);
    //ket thuc chuong trinh
    return 0;
}
```

```
Nhap 1 so thuc tu ban phim:3.14161592
so thuc vua nhap la: 3.141616
```

Dòng 3: Biến number dùng để lưu giữ giá trị số được nhập. Đề bài yêu cầu là nhập số nguyên nên kiểu biến number là int.

Dòng 8: Hàm scanf() đọc dữ liệu từ stdin theo đúng định dạng (format) và lưu trữ các giá trị vào các biến tương ứng. Căn cứ vào bảng các định dạng nhập thông dụng của hàm scanf() trong C theo sau đó là %d (số nguyên) hoặc %f (số thực).

Dấu & trong &number được hiểu là địa chỉ của biến number trong vùng nhớ tạm.

Dòng 9: Khi cần hiển thị giá trị đã được lưu thì chỉ cần đọc giá trị chứ không cần đọc địa chỉ của ô nhớ (không cần dấu &).

Chú ý: *Hàm scanf() không chấp nhận khoảng trống giữa hai chuỗi (khác với hàm gets()), tức là bạn chỉ có thể nhập một chuỗi liền nhau, nếu bạn nhập cả phần khoảng trống thì phần nội dung sau khoảng trống đầu tiên sẽ không được chấp nhận.*

Các kiểu dữ liệu trong C

Kiểu	Độ rộng bit	Dãy giá trị
char	1 byte	-127 tới 127 hoặc 0 tới 255
unsigned char	1 byte	0 tới 255
signed char	1 byte	-127 tới 127
int	4 byte	-2.147.483.648 tới 2.147.483.647
unsigned int	4 byte	0 tới 4.294.967.295
signed int	4 byte	-2.147.483.648 tới 2.147.483.647
short int	2 byte	-32.768 tới 32.767
unsigned short int	2 byte	0 tới 65.535
signed short int	2 byte	-32.768 tới 32.767
long int	4 byte	-2.147.483.647 tới 2.147.483.647
signed long int	4 byte	Tương tự như long int
unsigned long int	4 byte	0 tới 4.294.967.295
float	4 byte	+/- 3.4e +/- 38 (~7 chữ số)
double	8 byte	+/- 1.7e +/- 308 (~15 chữ số)
long double	8 byte	+/- 1.7e +/- 308 (~15 chữ số)
wchar_t	2 hoặc 4 byte	1 wide character

Các định dạng nhập thông dụng của hàm scanf() trong C

Định dạng	Kiểu dữ liệu	Ý nghĩa
%c	char	ký tự
%s	char *	chuỗi ký tự
%d	int, short	Số nguyên dạng thập phân
%f	float	Số thực
%lf	double	Số thực chính xác gấp đôi

Bài tập tương tự

a. Nhập vào bàn phím một số thực và in ra số sau khi làm tròn 3 chữ số

```
Nhap 1 so thuc tu ban phim:123.4567890
so thuc vua nhap la: 123.457
```

b. Nhập số từ bàn phím: toán hạng 1 là số nguyên, toán hạng 2 là số thực. Hãy tính tổng và in ra kết quả sau khi tính tổng. Tương tự cho các phép toán trừ, nhân và chia.

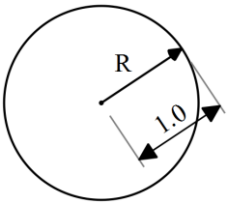
```
Nhap 1 so thuc tu ban phim:123.456
Nhap 1 so nguyen tu ban phim:14
Tong cua 123.456 + 14 la: 137.456
Hieu cua 123.456 - 14 la: 109.456
Tich cua 123.456 * 14 la: 1728.384
Thuong cua 123.456 / 14 la: 8.818
```

c. Viết chương trình hiển thị Họ và tên SV và Mã số SV trên 2 dòng của màn hình. Với 2 biến đầu vào là Hoten (kiểu char) và maso (kiểu int).

```
Nhap thong tin cua SV
Ho va ten SV: Doan Vu Thinh
MSSV: 60131333
```

Bài tập tổng hợp về các hàm cơ bản trong C

a. Viết chương trình tính chu vi và diện tích của hình tròn với số đo của bán kính được nhập từ bàn phím. Ví dụ: R = 1, Chu vi (C) là 3.14 và Diện tích (S) là 6.28.

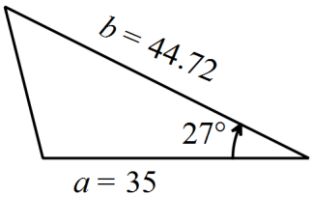


Chú ý:

R là kiểu số thực;

π *là hằng số có giá trị* 3.1416

b. Viết chương trình tính diện tích của tam giác. Với số đo 2 cạnh của tam giác và 1 góc được tạo bởi 2 cạnh của tam giác là kiểu số thực được nhập từ bàn phím.



Ví dụ: cạnh a (35 cm), cạnh b (44,72 cm), góc (27^0)

Diện tích là: 355.293 cm^2

Pi(radian ~3.1416)=180^0

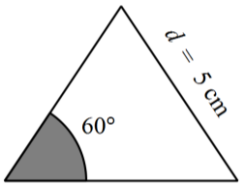
? (radian) = 27^0

Sin(?radian)

Gợi ý: Sử dụng thư viện math.h cho các phép toán lượng giác

Hệ số pi trong thư viện math.h có tên là M_PI

c. Viết chương trình tính diện tích của tam giác đều. Với số đo cạnh của tam giác là kiểu số thực và được nhập từ bàn phím.



Ví dụ: cạnh tam giác (d=5cm). Từ đó, S = 10.825 cm^2

Gợi ý: chứng minh công thức $S = {d^2 * \sqrt{3}}/{4}$

d. Viết chương trình chuyển đổi giá trị điện áp (Volte) thành giá trị nhiệt độ (^{0}C) của bộ chuyển đổi ADC-10 bit bên trong vi điều khiển (AVR hoặc PIC). Với công thức biểu diễn mối quan hệ giữa giá trị ADC và điện áp như sau:

$$V_{đo} = \frac{Gia\_tri\_ADC}{2^n - 1} * V_{ref}$$

Trong đó:

- V_{do} là gia trị điện áp đo được bởi cảm biến nhiệt độ (LM35, PT100)
- V_{ref} là điện áp tham chiếu - là giá trị điện áp lớn nhất mà ADC có thể chuyển đổi được. Mặc định khi được cấp nguồn điện áp tham chiếu của Arduino là 5V (có nghĩa là 5V ứng với giá trị ADC max là 1023, với board Arduino 3.3V thì 3.3V ứng với giá trị lớn nhất là 1023).
- n là độ phân giải của ADC, giải thích ở trên.

Yêu cầu:

- Nhập giá trị của điện áp đo được tại chân ADC (kiểu số thực, đơn vị đo là Volt)
- Giá trị của điện áp tham chiếu cho ADC (kiểu số thực): chọn 1 trong 3 mức điện áp sau: 2.5 Volt, 3.2 Volt hoặc 5.0 Volt
- Độ phân giải của ADC (kiểu số nguyên): chọn n =8 hoặc 10

  Chú ý: Giá trị điện áp đo được khi nhập phải bé hơn hoặc bằng (≤) Vref. Khi giá trị Vdo ≥ Vref thì Gia_tri_ADC = 2^n-1

Ví dụ: n = 8, Vdo = 1.5, Vref = 2.5 Ta có: Gia_tri_ADC la: 153

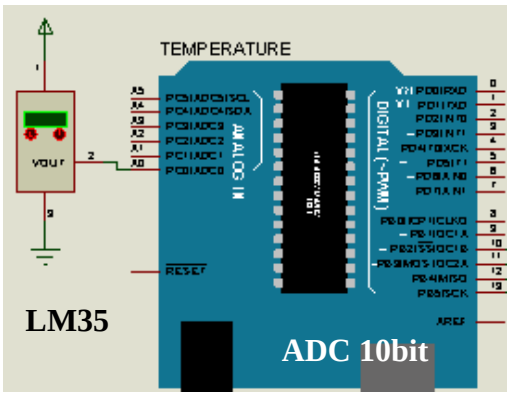
n = 10, Vdo = 2.12, Vref = 5.0 Ta có: Gia_tri_ADC la: 433

e. Cho sơ đồ mạch điện sau:

1. LM35 là cảm biến nhiệt độ, với đặc tính cứ 10mV sự thay đổi của điện áp ứng với 1°C.
2. ADC tích hợp bên trong Arduino có độ phân giải 10 bit
3. Giá trị điện áp đo được tại thời điểm khảo sát là 0.26 Volt và là đầu vào của ADC
4. Điện áp tham chiếu Vref là 5 Volt

Giá trị nhiệt độ tại thời điểm khảo sát là:

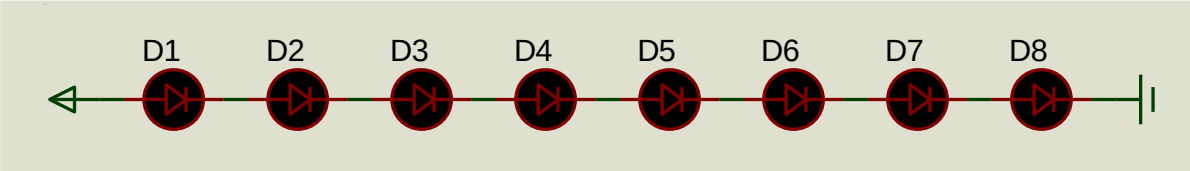
25.904 ^{0}C



Yêu cầu: Hãy lập trình để đọc giá trị điện áp tại các thời điểm khác nhau (mỗi lần đọc từ bàn phím), sau đó hãy hiển thị giá trị nhiệt độ đó lên màn hình.

Chú ý: Giá trị của Vref ≥ Điện áp đầu vào của cảm biến LM35

f. Cho dãy đèn LED được mắc nối tiếp như sau:



Trong quá trình vận hành, chẳng may có 1 bóng LED (D1 ~ D8) bị hỏng. Để xác định chính xác LED bị hỏng và tiến hành thay thế LED hỏng. Giải thuật là:

- *Bước 1:* Chia đôi dãy LED (D1~D4) và (D5~D8). Sau đó, kiểm tra từng dãy. Nếu dãy nào không sáng thì tiếp tục chia đôi dãy đó ra. Ví dụ trong trường hợp này là D5~D8
- *Bước 2:* Chia đôi dãy D5~D6 và D7~D8. Kiểm tra từng dãy như Bước 1, ta lại gặp dãy D5~D6 bị hỏng.
- *Bước 3:* Xác định lần lượt từng LED và tiến hành thay thế.

Kết luận với số LED là 8 thì số bước thực hiện kiểm tra là 3.

Hãy lập trình với số LED đầu vào là số nguyên được nhập từ bàn phím. (n số chẵn) và hiển thị số bước cần thực hiện là bao nhiêu.

Ví dụ: n = 8, số bước là 3

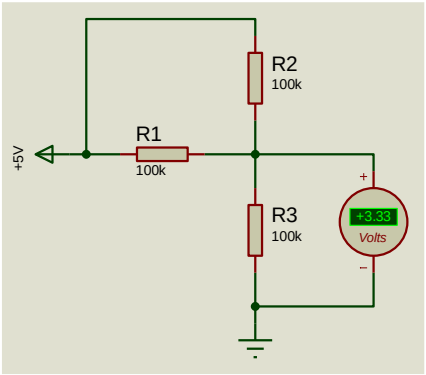
g. Cho mạch điện sau:

Nguồn cung cấp: 5 Volt

Điện trở: R1 = R2 = R3 100 KΩ

Giá trị điện áp trên R3 là: 3.33V

Hãy viết chương trình thay đổi giá trị điện trở cho R1, R2, R3 và điện áp đầu vào (+5V đến +20V) và xác định giá trị điện áp rơi trên R3



BÀI TẬP LIÊN QUAN ĐẾN ĐIỀU KIỆN VÀ RẼ NHÁNH

Bài tập 3.1. Nhập 2 số nguyên từ bàn phím (a,b) và in ra màn hình số lớn nhất trong 2 số

```
Nhap so a = 1
Nhap so b = -3
so lon nhat trong 2 so 1 va -3 la 1
```

```
1  #include <stdio.h>
2  //khai bao bien
3  int a;
4  int b;
5  int MAX;
6  //chuong trinh chinh
7  int main()
8  {
9      printf("Nhap so a = ");
10     scanf("%d",&a);
11     printf("Nhap so b = ");
12     scanf("%d",&b);
13     MAX=a;
14     if(MAX<b){
15         MAX=b;
16     }
17     printf("SLN(%d,%d) = %d",a,b,MAX);
18     //ket thuc chuong trinh
19     return 0;
20 }
```

Bước 1: Chọn ngẫu nhiên 1 trong 2 số (a,b) làm số lớn nhất và gán cho MAX (dòng 13)

Bước 2: Kiểm tra MAX với số còn lại (b). Nếu MAX < b (dòng 14) thì gán MAX = b (dòng 15). Lúc này giá trị lớn nhất sẽ được thay đổi giá trị (b) còn ngược lại MAX không đổi.

Bước 3: In ra giá trị MAX (dòng 17)

Với cách tiếp cận này có thể dễ dàng tìm số lớn nhất cho nhiều số (3, 4, … n) đều thuận tiện khi chỉ cần so sánh MAX lần lượt với n – 1 số còn lại.

Lưu ý: cặp dấu ngoặc {} được dùng để thực thi các lệnh bên trong khối này.

Bài tập 3.2. Giải phương trình bậc 1 có dạng tổng quát như sau: $ax + b = 0$, với a, b là các số thực được nhập từ bàn phím.

```
Nhap he so a= 2
Nhap he so b= 7
phuong trinh co nghiem -3.500
```

```
1  #include <stdio.h>
2  //khai bao bien
3  float a,b;
4  //chuong trinh chinh
5  int main()
6  {
7      printf("Nhap he so a= ");
8      scanf("%f",&a);
9      printf("Nhap he so b= ");
10     scanf("%f",&b);
11     if(a==0){
12         if(b==0){
13             printf("phuong trinh co vo so nghiem");
14         }
15         else{
16             printf("phuong trinh vo nghiem");
17         }
18     }
19     else{
20         printf("phuong trinh co nghiem %.3f",-b/a);
21     }
22     //ket thuc chuong trinh
23     return 0;
24 }
```

Bước 1: Xét a = 0, pt có dạng b=0. Do đó, nếu b=0 pt có dạng 0 = 0 còn khi b ≠0 thì phương trình trở nên vô nghiệm (dòng 11 - 18), khi muốn rẽ nhánh sang điều kiện khác ta sử dụng cặp: *if (điều kiện) {…}else{}* Ngoài ra: trong hàm if có thể lồng nhiều điều kiện khác.

Bước 2. Khi a ≠0 thì nghiệm pt là $-b/a$.

Lưu ý: ta phải xé a=0 và a≠0 vì khi a=0 sẽ rơi vào trường hợp phép tính $-b/a$ có mẫu số là 0.

Bài tập 3.3. Viết chương trình chuyển từ ký tự số (0-9) thành chữ

```
Nhap so co toi da 1 chu so:0
Khong
```

```
Nhap so co toi da 1 chu so:6
Sau
```

Xem xét chương trình sau:

```c
#include <stdio.h>
//khai bao bien
int number;
//chuong trinh chinh
int main()
{
    printf("Nhap so can chuyen:");
    scanf("%d",&number);

    switch(number){
        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;
        }
    }
    //ket thuc chuong trinh
    return 0;
}
```

Có thể nhận ra đoạn chương trình trên không sử dụng cú pháp *IF (điều kiện) ELSE* để rẽ nhánh điều kiện, thay vào đó là cú pháp *SWITCH (điều kiện) CASE trường hợp*. Với cú pháp này sẽ giúp chương trình dễ đọc hơn.

Bài tập tương tự

a. Tìm số lớn nhất, nhỏ nhất trong 3 số a, b, c. Với a, b, c là các số thực được nhập từ bàn phím

```
Nhap so a = 1.234
Nhap so b = 0.872
Nhap so c = 3.267
SLN(1.234,0.872,3.267) = 3.267
SLN(1.234,0.872,3.267) = 0.872
```

b. Giải phương trình bậc 2 có dạng tổng quát như sau: ax^2+bx+c=0, với a, b, c là các số thực được nhập từ bàn phím.

```
Nhap he so a = 2
Nhap he so b = 3
Nhap he so c = -5
Phuong trinh co dang 2*x^2 + 3*x + -5 = 0
delta = 49
pt co 2 nghiem phan biet:
x1= 1.000
x2 = -2.500
```

c. Giải hệ phương trình bậc nhất 2 ẩn sau:

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$$

Gợi ý:

$$d = \begin{vmatrix} a_1 b_1 \\ a_2 b_2 \end{vmatrix} = a_1 * b_2 - a_2 * b_1 \quad d_x = \begin{vmatrix} c_1 b_1 \\ c_2 b_2 \end{vmatrix} = c_1 * b_2 - c_2 * b_1 \quad d_y = \begin{vmatrix} a_1 c_1 \\ a_2 c_2 \end{vmatrix} = a_1 * c_2 - a_2 * c_1$$

Nghiệm của hệ phương trình là: $x = \frac{d_x}{d}, y = \frac{d_y}{d}$

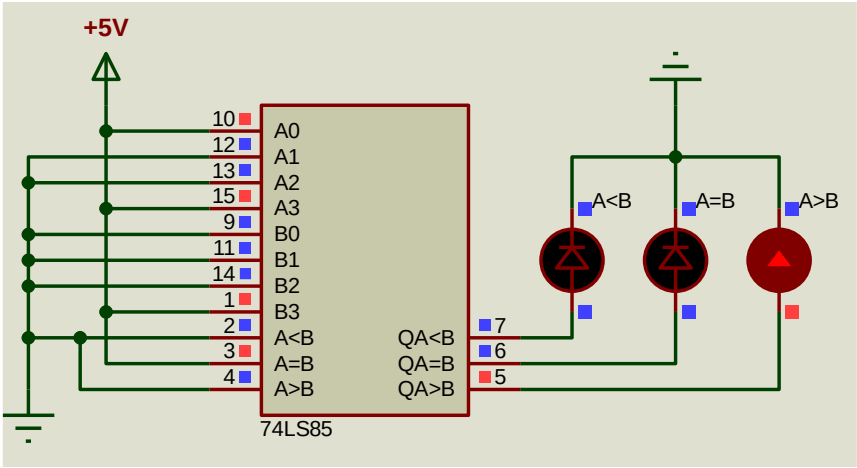
```
nhap a1 = 2
nhap b1 = 3
nhap c1 = 4
nhap a2 = 1
nhap b2 = 3
nhap c2 = 5
He phuong trinh co dang:
2x + 3y = 4
1x + 3y = 5
Nghiem (x,y) cua he phuong trinh la
(-1.000, 2.000)
```

d. Viết chương trình nhập số có 3 chữ số (số nguyên dương) từ bàn phím và chuyển thành chữ

```
Nhap so co toi da 3 chu so:205
Hai tram le Nam
```

```
Nhap so co toi da 3 chu so:75
Bay muoi lam
```

e. IC74LS85 là IC dùng để so sánh nhị phân. Xem sơ đồ mạch và nguyên lý hoạt động sau:



Số A có 4 bit (A3~A0): 1001

Số B có 4 bit (B3~B0): 1000

Kết quả: LED (A>B) được bật sáng có nghĩa là kết quả của IC so sánh nhị phân 4 bit cho biết số A lớn hơn số B.

Nguyên lý hoạt động:

So sánh bit có trọng số cao nhất (A3 và B3), nếu số nào có bit này lớn hơn (1>0) thì đưa ra kết quả A>B. Nếu A3=B3 thì chuyển sang so sánh A2 với B2, cũng nhận xét như khi so sánh A3 và B3. Cứ tiếp tục cho đến khi so sánh A0 và B0.

IC chỉ thực sự làm việc khi chân tín hiệu A=B (chân số 3 có mức tín hiệu 5V hoặc mức 1).

Viết chương trình mô tả nguyên lý hoạt động của bộ so sánh nhị phân 4 bit như trên. Trong đó số A và số B chỉ bao gồm các chữ số 0 và 1. Tín hiệu AEQB, AGTB và ALTB (chân số 3,4,2) được gán trực tiếp trong chương trình (define - hằng số). Ví dụ, trong hình trên thì AEQB = 1, các chân còn lại AGTB = 0 và ALTB = 0.

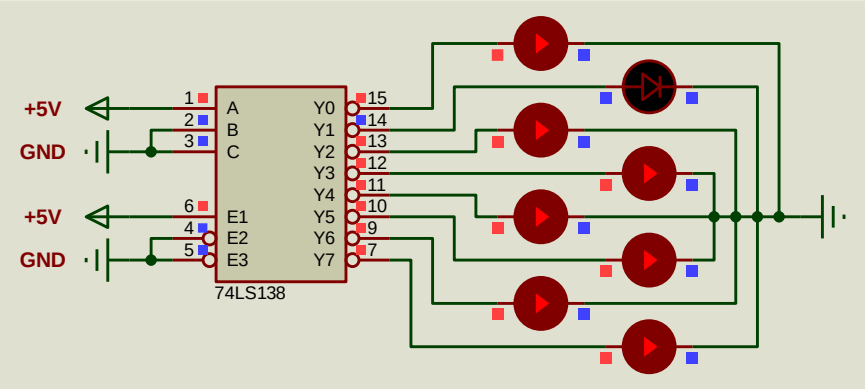
```
Nhap so A:1001
Nhap so B:0111
A>B
```

```
Nhap so A:1100
Nhap so B:1010
A>B
```

```
Nhap so A:0111
Nhap so B:1000
A<B
```

```
Nhap so A:1101
Nhap so B:1110
A<B
```

f. IC74138 là IC dùng để giải mã địa chỉ chọn chip, IC này có vai trò quan trọng trong việc lựa chọn địa chỉ cho thiết bị nào được phép hoạt động. 74LS138 có 3 đầu vào chọn 1 trong 8 đầu ra như sau:



C	B	A	$Y_i = 0$
0	0	0	Y_0
0	0	1	Y_1
0	1	0	Y_2
0	1	1	Y_3
1	0	0	Y_4
1	0	1	Y_5
1	1	0	Y_6
1	1	1	Y_7

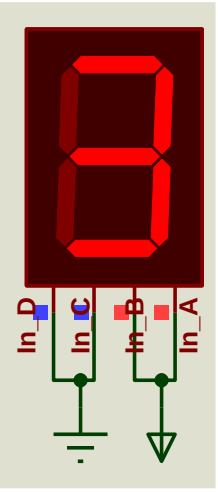
Hãy viết chương trình minh họa nguyên lý hoạt động cho IC 74138, trong đó:

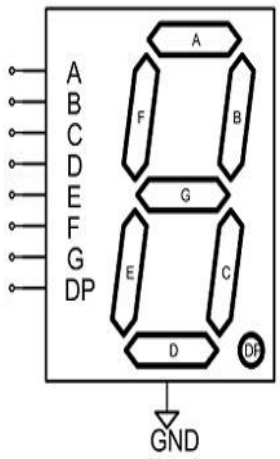
- A, B, C là 3 chân chọn địa chỉ (có giá trị là 0 và 1)
- E1, E2, E3 là 3 chân cho phép IC làm việc với giá trị tương ứng là 1, 0, 0
- Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7 là 8 chân đầu ra ứng với bảng trạng thái ở trên

```
Nhap so C:1
Nhap so B:0
Nhap so A:1
Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0
 1  1  0  1  1  1  1  1
```

```
Nhap so C:1
Nhap so B:2
Nhap so A:3
Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0
Gia tri dau vao khong hop le
```

g. LED 7 đoạn là thiết bị không thể thiếu trong các bộ hiển thị số, nó được ứng dụng rộng rãi các bộ đếm (đèn giao thông, đồng hồ kỹ thuật số, các máy đo và các mạch hiện số khác). Bản chất của LED7 đoạn là mã BCD với 4 đầu vào lựa chọn và 8 đầu ra (a-h) được quy định ở bảng sau:





In_D	In_C	In_B	In_A	Number
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	1	1	9

Bảng mã BCD ứng với các chân LED

A	B	C	D	E	F	G	DP	Mã
0	0	0	0	0	0	1	1	0
1	0	0	1	1	1	1	1	1
0	0	1	0	0	1	0	1	2
0	0	0	0	1	1	0	1	3
1	0	0	1	1	0	0	1	4

A	B	C	D	E	F	G	DP	Mã
0	1	0	0	1	0	0	1	5
0	1	0	0	0	0	0	1	6
0	0	0	1	1	1	1	1	7
0	0	0	0	0	0	0	1	8
0	0	0	0	1	0	0	1	9

Viết chương trình nhập giá trị đầu vào cho In_A, In_B, In_C và In_D (chỉ các số 0 và 1), sau đó in ra màn hình mã BCD tương ứng với các tín hiệu A, B, C, … , DP như bảng mã ở trên.

```
Nhap so D:0
Nhap so C:1
Nhap so B:1
Nhap so A:1
A       B       C       D       E       F       G       DP
0       0       0       1       1       1       1       1
So hien thi: 7
```

BÀI TẬP LIÊN QUAN ĐẾN VÒNG LẶP

Bài tập 4.1. Nhập vào 1 số N thỏa mãn điều kiện (0 ≤ N ≤ 10) sau đó in ra số N vừa nhập

```
Nhap so N = 9
So N da nhap: 9
```

```
Nhap so N = -3
N chua thoa DK
Nhap so N = 7
So N da nhap: 7
```

```
Nhap so N = -3
N chua thoa DK
Nhap so N = 19
N chua thoa DK
Nhap so N = -7
N chua thoa DK
Nhap so N =
```

Nhìn vào kết quả khi thực hiện chương trình bên trên có thể nhận ra giá trị của N có thể được ghi nhận ngay lần đầu tiên, hoặc phải mất 2 lần nhập hoặc cũng có thể là không biết đến khi nào mới có được giá trị chấp nhận được.

```c
#include <stdio.h>
//khai bao bien
int N;
//chuong trinh chinh
int main()
{
  printf("\nNhap so N = ");
  scanf("%d",&N);
  while(N<0 || N>10){
    printf("N chua thoa DK");
    printf("\nNhap so N = ");
    scanf("%d",&N);
  }
  printf("So N da nhap: %d",N);
  //ket thuc chuong trinh
  return 0;
}
```

```c
#include <stdio.h>
//khai bao bien
int N;
//chuong trinh chinh
int main()
{
  do{
    printf("Nhap so N = ");
    scanf("%d",&N);
  }while(N<0 || N>10);
  printf("So N da nhap: %d",N);
  //ket thuc chuong trinh
  return 0;
}
```

Nhìn vào 2 chương trình phía trên có thể nhận ra sự khác biệt của cấu trúc: while (điều kiện) {…} và do{…} while (điều kiện). Có thể hiểu là với trường hợp while(điều kiện){…}, chương trình sẽ kiểm tra điều kiện của số N nhập vào trước, nếu điều kiện đó không thỏa mãn, chương trình sẽ yêu cầu người dùng nhập lại số N thêm lần nữa cho đến khi thỏa mãn điều kiện thì dừng. Trong khi đó với cú pháp do {…} while (điều kiện) chương trình sẽ yêu cầu nhập vào số N trước sau đó mới đối chiếu điều kiện. Mới nhìn qua thì có vẻ chẳng có gì khác biệt, tuy nhiên nếu nhìn vào chương trình phía bên trái sẽ có thêm 2 dòng (7,8) để yêu cầu nhập số N trước. Sau khi nhập xong số N thì mới có cơ sở để câu lệnh while (dòng 9) thực hiện việc kiểm tra điều kiện.

Tóm lại, cả 2 cách viết đều giải quyết cùng một vấn đề nhưng có vẻ như cách thứ 2 (hình bên phải) tiết kiệm được 2 dòng nhập dữ liệu đầu vào cho số N.

Bài tập 4.2. Nhập vào một số N từ bàn phím (0 ≤ N ≤ 100) và in ra tất cả các số lẻ trong khoảng từ 1 đến N

```
Nhap vao so N: 30
Cac so le trong khoang tu 1 den 30 la:
  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29
```

```c
#include <stdio.h>
//khai bao bien
int N;
int i;
//chuong trinh chinh
int main()
{
    do{
        printf("Nhap vao so N: ");
        scanf("%d",&N);
    }while(N<0 || N>100);
    printf("Cac so le trong khoang tu 1 den %d la: \n",N);
    for(i=1;i<=N;i++){
        if(i%2!=0){
            printf("%3d",i);
        }
    }
  //ket thuc chuong trinh
  return 0;
}
```

Nhận xét:

Trong chương trình trên, để in các số lẻ trong khoảng từ 1 đến N, có thể nhận thấy rằng giá trị N ở đây đã được xác định bởi đoạn chương trình (dòng 8 - 12). Do đó, để in các số lẻ trong khoảng biết trước chỉ cần thực hiện đúng N số lần lặp, mỗi 1 lần lặp là 1 lần kiểm tra số đang xét có thỏa mãn tính chất là số lẻ không (số không chia hết cho 2) bằng câu lệnh (dòng 14). Nếu số đang xét (i) là số lẻ thì in số đó ra màn hình. Ngược lại, chương trình sẽ tự động tăng bộ đếm của vòng lặp (i) lên 1 đơn vị. Số lần thực hiện là N lần. Chương trình trên cũng có thể thay thế bởi câu lệnh lặp while () như sau:

```c
#include <stdio.h>
//khai bao bien
int N;
int i;
//chuong trinh chinh
int main()
{
    do{
        printf("Nhap vao so N: ");
        scanf("%d",&N);
    }while(N<0 || N>100);
    printf("Cac so le trong khoang tu 1 den %d la: \n",N);
    i=1;
    while (i<=N){
        if(i%2!=0){
            printf("%3d",i);
        }
        i++;
    }
  //ket thuc chuong trinh
  return 0;
}
```

Bài tập tương tự

a. Nhập vào một số có tối đa 6 chữ số và hiển thị số theo thứ tự ngược lại.

```
Nhap so can dao nguoc: 1234567
So dao nguoc cua 1234567 la 7654321
```

b. Tính tổng $S = 1 + 3 + 5 + 7 + \cdots + n$, với n là số nguyên (0≤ n ≤50)

```
Nhap n = 10
Tong cua S la: 25
```

c.

d. Tính tổng $T = \sqrt{2 + \sqrt{4 + \sqrt{8 + \sqrt{... + \sqrt{2^n}}}}}$, với n là số nguyên (0≤ n ≤10)

```
Nhap n = 3
Tong cua T la: 2.394
```

e. Tính giai thừa của một số (n!) với n là số nguyên (0≤ n ≤10). *Ví dụ: 5! = 120*

f. Viết chương trình tìm ước chung lớn nhất UCLN(a,b), bội chung nhỏ nhất BCNN(a,b) của 2 số cho trước. Với a,b là các số nguyên được nhập từ bàn phím (0 < a,b < 1000)

```
Nhap a = 4
Nhap b = 6
UCLN(4,6) = 2
BCNN(4,6) = 12
```

g. Viết chương trình tính tổng cho 2 phân số có dạng $^a/_b + ^c/_d$, trong đó a,b,c,d là các số nguyên được nhập từ bàn phím

```
Nhap tu so 1: 1    Tong cua 2 phan so chua toi gian la:  Sau khi toi gian la:
Nhap mau so 1: 6         10                                    5
Nhap tu so 2: 2    ________                              ________
Nhap mau so 2: 8         24                                    12
```

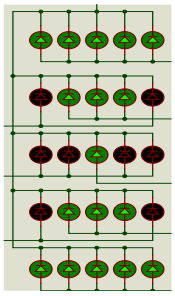
h. Viết chương trình in ra dãy số Fibonaccy, với 2 số đầu (n1, n2) và số phần tử (n+2). Với n1, n2, n là các số nguyên được nhập từ bàn phím (0 < a,b < 100)

```
Nhap so phan tu cua day Fibonaccy = 7
Nhap phan tu thu 1 = 1
Nhap phan tu thu 2 = 2
1       2       3       5       8       13      21
```

i. Viết chương trình kiểm tra một số N (được nhập từ bàn phím) có phải là số nguyên tố hay không, với điều kiện (0≤ n ≤1000)

```
Nhap so can kiem tra = 885          Nhap so can kiem tra = 61
885 khong phai la so nguyen to      61 la so nguyen to
```

j. Màn hình hiển thị LED có ma trận 5*5 như hình. Viết chương trình hiển thị các hình ảnh sau trên màn hình (mỗi pixel có thể chọn * @ hoặc $ để hiển thị)



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

k. Viết chương trình đổi số hệ 10 sang hệ 2, trong đó số hệ 10 là số nguyên nhập từ bàn phím

```
Nhap vao so he 10: -6
So he 2: 1111111111111010
```

```
Nhap vao so he 10: 6
So he 2: 0000000000000110
```

l. Viết chương trình đổi số hệ 2 thành hệ 10, trong đó số hệ 2 chỉ bao gồm các số 0 và 1

```
Nhap vao so he 2: 111
So he 10: 7
```

```
Nhap vao so he 2: 1001001
So he 10: 73
```

m. Viết chương trình đổi cơ số hệ 10 sang hệ 8, trong đó số hệ 10 là số nguyên nhập từ bàn phím

```
Nhap vao so he 10: 9
So he 8: 11
```

```
Nhap vao so he 10: 20
So he 8: 24
```

n. Viết chương trình đổi số hệ 10 sang hệ 16, trong đó số hệ 10 là số nguyên nhập từ bàn phím

```
Nhap vao so he 10: 15
So he 16: F
```

```
Nhap vao so he 10: 253
So he 16: FD
```

```
Nhap vao so he 10: 32672
So he 16: 7FA0
```

o. Gen hay DNA là vật chất mang thông tin di truyền, nằm trong tất cả các tế bào của con người. Với cách sắp xếp các cặp bazơ A-T-G-C khác nhau sẽ tạo nên những đặc điểm sinh học đa dạng, từ màu da, nước tóc đến nhóm máu. Hãy viết chương trình nhập vào từng ký tự trong bộ 4 mã hóa A T G C nói trên, kết thúc là phím số 0. Sau đó, hãy in ra màn hình: số lượng các base của các loại A, T, G, C và tỷ lệ phần trăm G-C có trong chuỗi DNA vừa nhập.

```
Nhap chuoi DNA (cac ky tu atgc):
attgtcgagacc
acctgtga
ttgaga
cctctggact
0
Tong so base loai a = 8
Tong so base loai t = 10
Tong so base loai g = 9
Tong so base loai c = 9
%GC = 0.500
```

BÀI TẬP LIÊN QUAN ĐẾN HÀM

Bài tập 5.1. Viết chương trình nhập vào 1 số nguyên và in ra số đó là số lẻ hay số chẵn

```
Nhap so N: 31  Nhap so N: 42
31 la so le    42 la so chan
```

Xem xét các chương trình sau:

```c
#include<stdio.h>
//khai bao bien
int N;
//cac ham
//chuong trinh chinh
int main(){
    //thao tac nhap so N
    printf("Nhap so N: ");
    scanf("%d",&N);
    //ktra N la chan hay le
    if(N%2!=0){
        printf("%d la so le",N);
    }
    else{
        printf("%d la so chan",N);
    }
    //ket thuc chuong trinh
    return 0;
}
```

Dòng 8-9: thao tác nhập số N từ bàn phím
Dòng 11-16: kiểm tra và in ra kết quả chẵn/lẻ

```c
#include<stdio.h>
//khai bao bien
int N;
//nhap so N
void nhap_N(){
    printf("Nhap so N: ");
    scanf("%d",&N);
}
//chuong trinh chinh
int main(){
    //thao tac nhap so N
    nhap_N();
    //ktra N la chan hay le
    if(N%2!=0){
        printf("%d la so le",N);
    }
    else{
        printf("%d la so chan",N);
    }
}
```

Dòng 12: thao tác nhập số N từ bàn phím được thay thế bằng cách gọi hàm "*nhap_N()*"

```c
#include<stdio.h>
//khai bao bien
int N;
//nhap so N
void nhap_N(){
    printf("Nhap so N: ");
    scanf("%d",&N);
}
//in ket qua
void in_ketqua(){
    if(N%2!=0){
        printf("%d la so le",N);
    }
    else{
        printf("%d la so chan",N);
    }
}
//chuong trinh chinh
int main(){
    //thao tac nhap so N
    nhap_N();
    //ktra N la chan hay le
    in_ketqua();
}
```

Dòng 23: in ra màn hình kết quả là số chẵn hay lẻ được gọi từ hàm "*in_ketqua()*"

Từ 3 chương trình trên có thể rút ra một số nhận xét như sau:

1. Sử dụng hàm triệt để làm cho chương trình chính (*main()*) trở nên ngắn gọn hơn, điều này có ý nghĩa quan trọng nếu chương trình quá nhiều câu lệnh.
2. Sử dụng hàm sẽ tiết kiệm được công sức và thời gian khi phải sao chép đoạn mã có cùng chức năng, ví dụ các hàm in kết quả đôi khi trong 1 chương trình có thể phải viết đoạn mã in kết quả đến vài lần.
3. Hàm bắt đầu với int và void, có thể hiểu là int là hàm cần có giá trị trả về, ví dụ hàm int main() có giá trị trả về là 0 trong khi đó hàm bắt đầu void thì không có giá trị trả về. Vậy khi nào dùng void và khi nào dùng int cho tên của hàm?
4. Khi cần kiểm tra một tham số nào đó có thỏa mãn điều kiện cho trước chúng ta nên dùng int thay cho void. Bởi vì khi kết thúc hàm kiểm tra chúng ra sẽ căn cứ vào giá trị trả về mà thực hiện thao tác tiếp theo.

Bài tập 5.2. Tính tổng các số lẻ có trong khoảng 1 đến N (5 ≤ N ≤100)

```
Nhap so N: 7
Tong 1 + 3 + ... + 7 = 16
```

Xem xét chương trình sau:

```
1   #include<stdio.h>
2   //khai bao bien
3   int i,N,tong;
4   //ham nhap so N
5   void nhap_N(){
6       do{
7           printf("Nhap so N: ");
8           scanf("%d",&N);
9       }while(N<5 || N>100);
10  }
11  //ham kiem tra so le
12  int is_le(int a){
13      if(a%2!=0){
14          return 1;
15      }
16      else{
17          return 0;
18      }
19  }
```

```
20  //chuong trinh chinh
21  int main(){
22      //nhap so N
23      nhap_N();
24      //tong cac so le
25      tong=0;
26      for(i=1;i<=N;i++){
27          if(is_le(i)==1){
28              tong=tong+i;
29          }
30      }
31      //in ket qua
32      printf("Tong 1 + 3 + ... + %d = %d",N,tong);
33      //ket thuc chuong trinh
34      return 0;
35  }
```

Chú ý: câu lệnh ở dòng số 27: *if(is_le(i)==1)*, và cách khai báo tên hàm *is_le(int a)* ở dòng 12

Giải thuật để tính tổng các số lẻ trong khoảng từ 1 đên N, gồm có các bước sau:

- *Bước 1:* Nhập và kiểm tra số N
- *Bước 2:* Duyệt qua tất cả các số bắt đầu từ 1 cho đến N. Mỗi lần duyệt sẽ:
    - Bước 2.1. Kiểm tra số đang xét có phải là số lẻ hay không? Nếu đúng sẽ cộng dồn số đang xét vào biến *tong*. Nếu không thỏa điều kiện thì chuyển sang Bước 2.2
    - Bước 2.2. Tăng số cần kiểm tra lên 1 đơn vị
- *Bước 3.* In ra màn hình kết quả của *tong*

Cụ thể, ở bước kiểm tra lần lượt các số thứ *i* có phải là số lẻ hay không ta nhận thấy rằng việc kiểm tra này thực hiện n lần như nhau. Do đó, khi xây dựng hàm *is_le* để kiểm tra chúng ta có thể sử dụng tham số *a* để truyền vào cho hàm *is_le()* với a là kiểu số nguyên (cùng kiểu dữ liệu của biến *i*). Ngoài ra, mỗi lần kiểm tra, hàm *is_le(int a)* cần báo cho chương trình biết kết quả kiểm tra là gì để khi so sánh kết quả kiểm tra mà quyết định công việc tiếp theo. Do đó, chúng ta có thể sử dụng "*cờ*" để biết kết luận của hàm kiểm tra là: số 1 - báo hiệu số lẻ và số 0 - báo hiệu số chẵn. Vì thế, chúng ta bắt đầu hàm kiểm tra với *int is_le(int a)*. Trong đó, giá trị trả về được gán thông qua lệnh *return 0* hoặc *1*

Bài tập tương tự

a. Tính tổng các giai thừa: 1! + 2! + 3! …+N! (5 ≤ N ≤10)

```
Nhap so N: 6
Tong 1! + 2! + ... + 6! = 873
```

b. In ra tổng các số nguyên tố trong khoảng từ 2 đến N (100 ≤ N ≤1000)

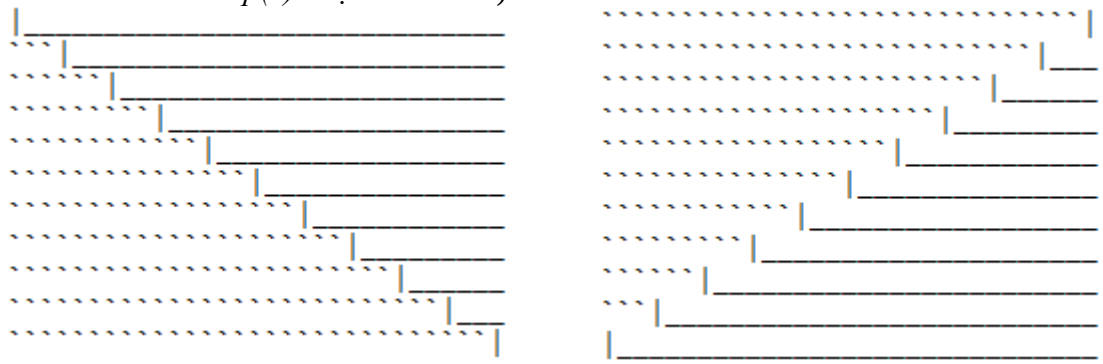
```
Nhap so N: 20
Cac so nguyen to: 2     3       5       7       11      13      17      19
Tong cac so nguyen to tu 2 den 20 la 77
```

c. Nhập vào một số nguyên N (5 ≤ N ≤10) và in ra các ước số (không bao gồm chính nó) và tính tổng của các ước số đó.

```
Nhap N= 42
Cac uoc so cua 42 la:     1    2    3    6    7   14   21
Tong cua cac uoc so khong bao gom chinh no = 54
```

Viết chương trình mô phỏng PWM (điều biến độ rộng xung) – dùng để điều khiển độ sáng tối của bóng đèn, tốc độ động cơ điện một chiều. Khoảng thời gian lấp đầy mức cao được quy ước cho thời gian sáng của bóng đèn (thời gian làm việc của động cơ) và ngược lại. Quá trình này giúp cho năng lượng tiêu thụ của bóng đèn hoặc động cơ thay đổi theo thời gian từ đó thay đổi độ sáng của đèn hay tốc độ của động cơ.

(*Gợi ý: sử dụng thư viện windows.h và hàm Sleep(x) để làm chậm quá trình hiển thị - chú ý chữ S của hàm Sleep(x) được viết HOA*)



d. Viết chương trình nhập vào một số N từ bàn phím (N<1.000.000.000), sau đó in ra ký số lớn nhất trong số vừa nhập là số mấy và ở vị trí nào.

```
Nhap so N: 1234   Nhap so N: 1243   Nhap so N: 1423   Nhap so N: 4123
MAX: 4, VT_MAX: 4 MAX: 4, VT_MAX: 3 MAX: 4, VT_MAX: 2 MAX: 4, VT_MAX: 1
```