BÀI THỰC HÀNH SỐ 7

1. MỤC TIÊU

- Hiểu và vận dụng được lệnh **`do ... while`**
- Viết các hàm do người lập trình tự định nghĩa (**`user define function`**) với lệnh **`while/for/do...while.`**

2. LÝ THUYẾT CẦN GHI NHỚ

2. 1. Cấu trúc `do-while`

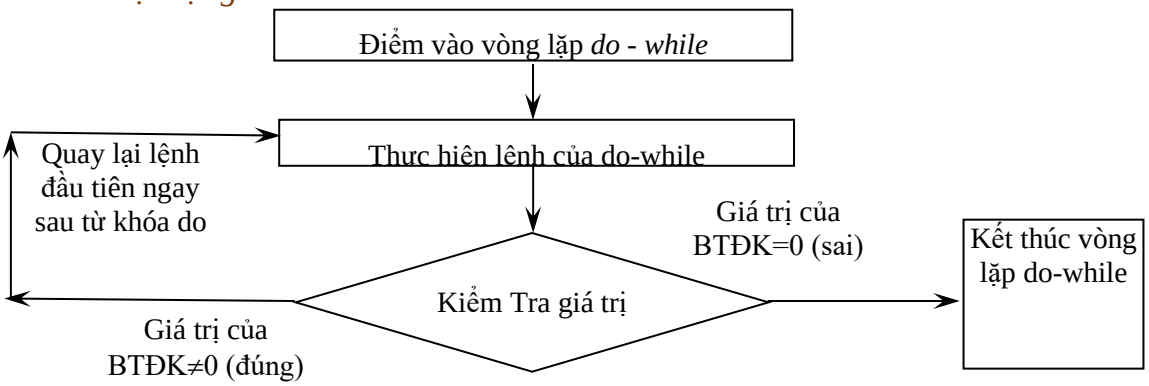
2.1.1. *Công dụng*

Cấu trúc **`do-while`** thường được dùng khi cho thực hiện trước một nhóm lệnh, sau đó mới thực hiện kiểm tra một số điều kiện để quyết định xem có thực hiện lại nhóm lệnh đó hay không? Nếu kết quả kiểm tra biểu thức điều kiện là true (đúng), chương trình cho thực hiện lại nhóm lệnh, ngược lại (false – sai) sẽ kết thúc việc thực hiện của lệnh **`do...while`**.

2.1.2. *Cú pháp*

```
do
{
    Nhóm Lệnh;
}while (btđk);
```

2.1.3. *Lưu đồ hoạt động của do – while*



2. 2. 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.

3. BÀI TẬP THỰC HÀNH CƠ BẢN

Từ buổi thực hành này đến cuối môn học, yêu cầu sinh viên tổ chức chương trình dưới dạng các hàm chức năng cho tất cả các bài tập.

Bài 1. Cho nhập số nguyên dương n, liệt kê tất cả các ước số của n.

Hướng dẫn

```
void liet_ke_uoc_so(int n){
    for(int i =1; i<=n/2; i++){
        if(n%i==0){  // la uoc so
            // in i ra màn hình
        }
}
```

Bài 2. Viết chương trình cho nhập nhiều lần, 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 (tương tự bài tập 84 phần 3.4.2.1, chỉ khác là ở đây sử dụng lệnh do ... while)).

Hướng dẫn

- Sử dụng do… while

```
void nhap_so_n(int n){
  // khai bao bien tong va gan gia tri = 0;
    do{
        // nhap n
       // tinh tong
    }
    while(n!=0)
}
```

Bài 3. 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***.

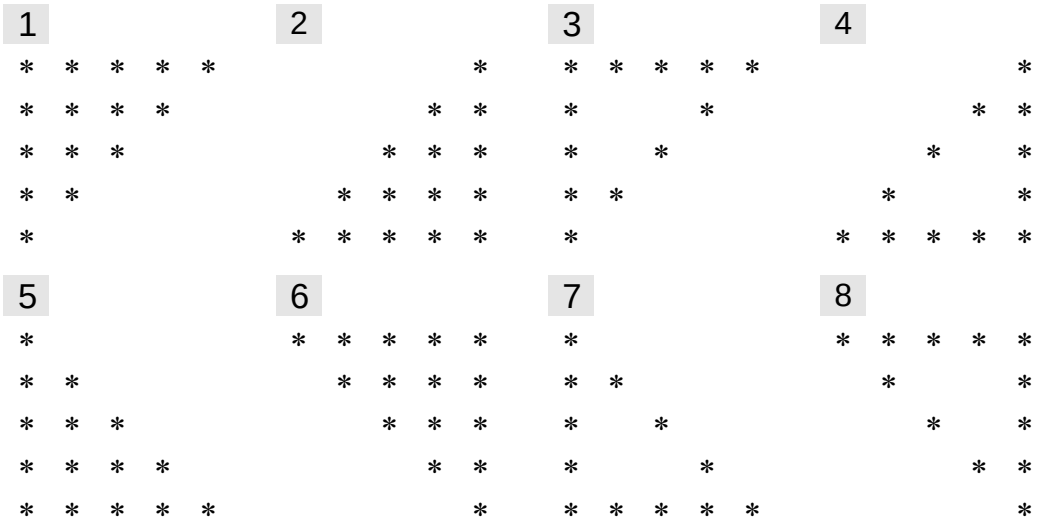
b. Điều kiện: **0 <= *n* <= 9**. Nếu nhập đúng, chương trình in ra cách đọc của số vừa nhập.

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 lượng phần tử của dãy là ***n*** .

Bài 4. Viết chương trình cho người dùng nhập cạnh kề (n) của tam giác vuông cân. Giả sử với n=5, chương sẽ lần lượt in ra các hình sau:



Hướng dẫn:

- Sử dụng 2 vòng lặp
- Câu 1:
  - số ngôi sao ở dòng 2 = số ngôi sao dòng 1 -1
  - số ngôi sao dòng 3 = số ngôi sao dòng 1 -2
  - dùng 2 vòng lặp: for(int i =1; i<=n; i++)

    for(int j =1, j<n-i; j++) { // in ngôi sao }
- Câu 2: hàng 1 có 1 ngôi sao, còn lại là dấu khoảng trắng
  - Hàng 2 có 2 ngôi sao, còn lại là dấu khoảng trắng
  - Nếu nhập vào n =5 thì
    - dòng 1 (i=1) có 1 dấu sao, (n-1)=4 dấu khoảng trắng
    - dòng 2 (i=2) có 2 dấu sao, (n-2)=3 dấu khoảng trắng

```
for(int i =1; i<=n; i++){
    for(int j =1; j<=n; j++){
        if(j<=n-i) // in dau khoảng cách
        else // in dấu *
    }
    printf("\n");
```

– Tương tự cho các trường hợp còn lại

Bài 5. Cho nhập số nguyên dương n. Kiểm tra xem n có phải là số nguyên tố hay không?

Hướng dẫn

- Số nguyên tố là số thõa điều kiện: chỉ chia hết cho 1 và cho chính nó (có 2 ước số là 1 và chính nó)
- Phương pháp kiểm tra số nguyên tố:
  - Nếu số x có giá trị<2 ➔ đây không là số nguyên tố
  - Nếu số x là 2➔ là số nguyên tố
  - Nếu số x >2 ➔ đếm ước số của của x từ 3 đến căn bậc 2 của x (chỉ xét số lẻ, không xét số chẵn)

```
bool kiem_tra_so_nt(int n){
   if(n<2) return false;
   else if(n==2) return true;
      else {
         for(int i =3;i<=sqrt(n); i+2){
            // kim tra i có phải là ước số của n không, nếu phải thì return false
         }
      }
  return true;
}
```

Bài 6. Cho nhập số nguyên dương n. Liệt kê các số nguyên tố < n.

Hướng dẫn

- Viết hàm kiểm tra số nguyên tố (bài 5)
- Viết vòng lặp từ 2=>n, kiểm tra từng phần tử trong vòng lặp có là số nguyên tố không và in ra

Bài 7. Cho nhập số nguyên dương n. Đếm các số nguyên tố <n.

Hướng dẫn

- Tương tự bài 6, sử dụng biến đếm

Bài 8. Cho nhập số nguyên dương n Tìm số nguyên tố đầu tiên <n và có giá trị gần với n nhất.

Hướng dẫn

- Viết hàm kiểm tra số nguyên tố
- Viết vòng lặp để kiểm tra số nguyên tố, vòng lặp này giảm dần từ n-1

Bài 9. Cho nhập số nguyên dương n. Tìm số nguyên tố đầu tiên >n và có giá trị gần với n nhất.

- Viết hàm kiểm tra số nguyên tố
- Viết vòng lặp kiểm tra các số nguyên tố >n, vòng lặp này bắt đầu từ n+1 và ngừng lặp khi tìm được số nguyên tố

```
int i =n+1;
while(kiem_tra_snt(n)==false){
    // tăng biến i lên 1 đơn vị
}
```

4. BÀI TẬP NÂNG CAO

Bài 1. 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***). Nếu nhập đúng, in ra các số lẻ nằm trong khoảng từ ***n*** đến ***m***. Khi người dùng nhập sai, thực hiện lần lượt các trường hợp sau:

a. Chỉ yêu cầu nhập lại ***m***.
b. Yêu cầu nhập lại cả ***n*** và ***m***.

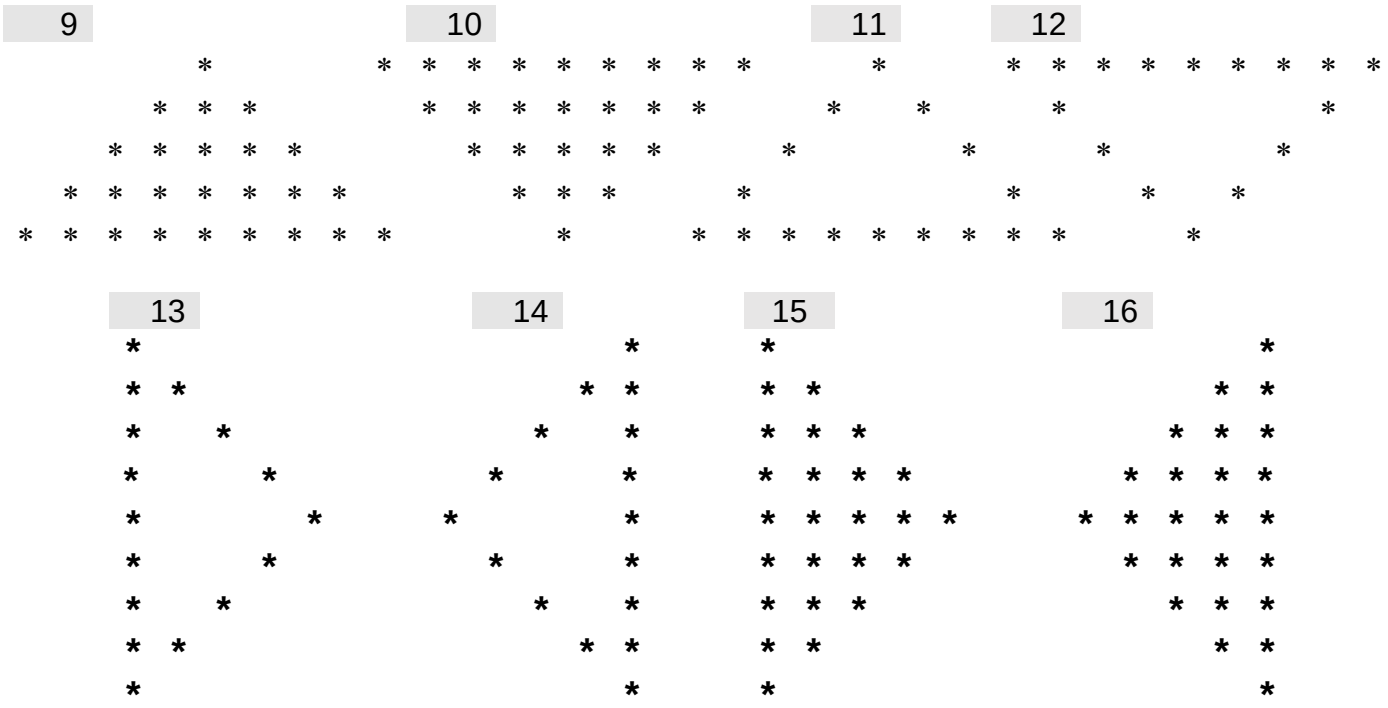
Bài 2. Cho nhập 2 số ***a***, ***b*** sao cho: số lớn nhất trong 2 số phải là một số dương và chia hết cho ***7***. Nếu nhập sai phải yêu cầu nhập lại cả 2 số cho đến khi đúng. Ngược lại, khi nhập đúng chương trình sẽ in ra các số chia chẵn cho ***7*** và nằm trong khoảng từ ***a*** đến ***b***.

Bài 3. Cho nhập 2 số nguyên dương ***a*** và ***b*** sao cho ước số chung lớn nhất của a và b phải là bội số của 5. Nếu nhập sai, cho nhập lại cả 2 số ***a*** và ***b***. Nếu nhập đúng, đếm xem từ ***a*** đến ***b*** có bao nhiêu số chia chẵn cho 3.

Bài 4. Nhập vào số nguyên dương ***x*** là lũy thừa của 2. Nếu nhập sai, yêu cầu nhập lại. Khi nhập đúng, in ra số ***x*** dưới dạng **`2^k`**. Với k là số lượng số 2 tham gia vào phép nhân để có được giá trị ***x***).

Ví dụ **`x=32 = 2*2*2*2*2`** => in ra **`2^5`**

Bài 5. Viết chương trình cho người dùng nhập cạnh (n) của tam giác. Giả sử với n=5, chương sẽ lần lượt in ra các hình sau:



Bài 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

Bài 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.

Bài 8. Trung bình cộng các số nguyên tố >=X và <=Y (với 3 <= X < Y <= 1000).

Bài 9. N bắt đầu bằng chữ số là số nguyên tố?

Bài 10. Tổng các chữ số có giá trị nguyên tố của N.

Bài 11. Cho nhập vào hai số n và m, in ra n số nguyên tố mà giá trị của chúng phải lớn hơn m.

Bài 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.

Bài 13. Viết chương trình minh họa cho định lý *Bertrand–Chebyshev* (hay định lý *Chebyshev*): "***với mọi n>1, luôn tồn tại một số nguyên tố nằm giữa hai số n và 2n***". Yêu cầu: cho nhập n, tìm số nguyên tố lớn nhất nằm trong đoạn từ n đến 2n.

Ví dụ n=2 ⇒ ở giữa 2 và 4 có số nguyên tố lớn nhất là 3,
n=3 ⇒ ở giữa 3 và 6 có số số nguyên tố lớn nhất là 5,

`n=4` ⇒ `ở giữa 4 và 8 có số nguyên tố 5 và 7 => chỉ in ra số nguyên tố lớn nhất là 7, v.v...`

Bài 14. Viết chương trình minh họa cho những số <100 về giả thuyết của nhà toán học người Đức - *Christian Goldbach* (1690-1764, trong một bức thư ngày 07/6/1742 gửi cho nhà toán học tài ba *Leonhard Euler*): ***mọi số tự nhiên chẵn lớn hơn 2 đều có thể viết được thành tổng của hai số nguyên tố*** (giả thuyết này đã được chỉ ra là đúng tới 4×10^{18}).

Ví dụ `4 = 2 + 2, 8 = 5 + 3, 20 = 13 + 7`

Bài 15. Nguyên tố cùng nhau (*coprime* hoặc *relatively prime*)

- Các số nguyên *a* và *b* được gọi là nguyên tố cùng nhau nếu chúng có ước số chung lớn nhất là 1.
  ⇨ Số `1` là nguyên tố cùng nhau với mọi số nguyên.
- Ví dụ
  - Nguyên tố cùng nhau: `6` và `35`; `6` và `25`
  - KHÔNG nguyên tố cùng nhau: `6` và `27` (vì USCLN của `6` và `27` là `3`)
