I. Biến.

- Cú pháp: <kiểu dữ liệu> <tên biến>;
  <kiểu dữ liệu> <tên biến> = <giá trị của biến>;
- Khai báo nhiều biến, cú pháp:
  <kiểu dữ liệu> <tên biến 1>, <tên biến 2>, <tên biến 3>;
  <kiểu dữ liệu> <tên biến 1> = <giá trị 1>, <tên biến 1> = <giá trị 2>;
- Quy tắc đặt tên biến:
  - Tên biến có thể bao gồm chữ in hoa và in thường, số, dấu '_'.
  - Tên biến không được bắt đầu bằng số và không có ký tự khoảng trắng.
  - Tên biến không được là những keywords của ngôn ngữ C (Như là: int, float, static,…).

  VD: Tên có ý nghĩa: max, min, sum, solve, isPrime, calcAverage,…
  Tên không có ý nghĩa: a, b, c, A, B, C, a1, a2, a3,…
- Các kiểu dữ liệu:

STT	Tên	Định dạng	Mô tả	Vùng giá trị	Ví dụ
1	int	%d	Kiểu dữ liệu số nguyên	-32768 tới 32767	int a = 1;
2	double	%lf	Kiểu dữ liệu là số thập phân	2.3E - 308 tới 1.7E + 308	double b = 7.95;
3	char	%c hoặc %s nếu là chuỗi	Kiểu dữ liệu lưu trữa giá trị của 1 ký tự trong bảng mã ASCII.	-128 to 127	char c = 'A'; char s = "abc";

- Biến toàn cục (Global): Được khai báo nằm ngoài tất cả các cặp dấu ngoặc nhọn {}, thường được đặt sau các dòng khai báo thư viện (#include <stdin.h>). Là biến tồn tại và có thể sử dụng trên toàn chương trình.
- Biến cục bộ (Local): Được khai báo trong cặp dấu ngoặc nhọn {}. Được khai báo trong cặp dấu ngoặc nhọn nào thì chỉ có thể tồn tại và sử dụng trong cặp dấu ngoặc nhọn đó, khi chương trình thoát khỏi cặp dấu ngoặc nhọn thì biến đó sẽ biến mất.

Ví dụ 1:

```
#include <stdio.h>

int globalVariable = 50;
```

```
void otherFunction(){
    int localVariable2 = 100;

    printf("%d\n", globalVariable);
    // printf("%d", localVariable1); // Error
    printf("%d\n", localVariable2);
}

int main() {
    int localVariable1 = 200;

    otherFunction();

    printf("%d\n", globalVariable);
    printf("%d\n", localVariable1);
    // printf("%d", localVariable2); // Error

    return 0;
}
```

Ví dụ 2:

```
#include <stdio.h>

int n = 10;

void func(){
    int n = 0;
    printf("%d", n);
}

int main(){
    func();

    return 0;
}
```

→ Không nên đặt tên biến local trùng tên với biến global đã có.

II. Nhập dữ liệu.

1) Hàm xuất dữ liệu printf().

- Cú pháp 1: **printf(“**<Chuỗi>**”);**
- Cú pháp 2: **printf(“**<Chuỗi chứa các định dạng>**”,** <Các giá trị>**);**
- <Chuỗi chứa các định dạng> là chuỗi thông thường và các định dạng cho các giá trị cần in ra (%d, %fd, %c, %s).
- <Các giá trị> là danh sách các giá trị cần được in ra tại các vị trí chứa các định dạng trong <Chuỗi chứa các định dạng>, các giá trị này cần được sắp xết theo đúng thứ tự xuất hiện của các định dạng và đủ số lương, được ngăn các nhau bởi dấu phẩy.

Ví dụ:

```
printf("first string\n");
int a = 1;
double b = 5.8;
char c = 'M';
printf("a = %d\nb = %lf\nc = %c\nd = %d", a, b, c, 58);
```

2) Hàm nhập dữ liệu scanf().

- Cú pháp: **scanf(“**<Các định dạng>**”,** <Các địa chỉ của các biến cần lưu>**);**
- <Các định dạng> là chuỗi các ký tự %d, %lf, %c, %s không cần ngăn cách nhau bởi ký tự nào hết (Nếu có cũng không sao).
- <Các địa chỉ của các biến cần lưu> thường được lấy bằng cú pháp **&**<Tên biến>.
- Sẽ có sự khác biệt khi nhập giá trị là một chuỗi các ký ưu **(Xem ở phần Mảng)**.

Ví dụ:

```
int a, b;
scanf("%d%d", &a, &b);
printf("a + b = %d", a+b);
```

Bài tập: Nhập vào 2 số nguyên a, b. In ra màn hình các chuỗi sau:

“1) 2a + 3b = **[Kết quả biểu thức]**”.

“2) S1 = **[Diện tích hình chữ nhật cạnh a và b]**”.

“3) S2 = **[Diện tích tam giác vuông có 2 cạnh góc vuông là a và b]**”.

II. Ép kiểu (Chuyển kiểu).

- Cú pháp: **(**<Kiểu mới>**)** <Số hoặc tên biến muốn đổi kiểu>
- Dùng để làm tròn **xuống** một số thực:

Ví dụ:

```
double a = (int)4.7;
printf("%lf", a);

double b = 3.7;
double c = (int)b;
printf("\n%lf", c);
```

- Dùng trong phép chia 2 số nguyên **(Xem ở mục III.1 phần lưu ý)**.

III. Các toàn tử.

1) Toán tử số học.

Ký hiệu	Mô tả	Ví dụ
+	Cộng toán hạng bên trái cho toán hạng bên phải	int a=6, b=3; int c = a + b; // c = 9
-	Trừ toán hạng bên trái cho toán hạng bên phải	int a=6, b=3; int c = b - c; // c = 1
*	Nhân toán hạng bên trái với toán hạng bên phải	int a=6, b=3;

		int c = b * c; // c = 18
/	Chia toán hạng bên trái cho toán hạng bên phải	int a=6, b=3; int c = b / c; // c = 2
%	Chia toán hạn bên trái cho toán hạng bển phải rồi lấy phần dư	int a=6, b=3; int c = b % c; // c = 0

++	Toán tử tăng lên 1 đơn vị.
	++i thì i được tăng trước rồi mới lấy kết quả để thực hiện biểu thức.
	i++ thì i được đưa vào thực hiện biểu thức rồi mới tăng i.
		int i = 0, j = 0;
		printf("%d %d", ++i, j++);
		printf("\n%d %d", i, j);

--	Toán tử giảm đi 1 đơn vị.
	++i thì i được giảm trước rồi mới lấy kết quả để thực hiện biểu thức.
	i++ thì i được đưa vào thực hiện biểu thức rồi mới giảm i.
		int i = 0, j = 0;
		printf("%d %d", --i, j--);
		printf("\n%d %d", i, j);

- **Lưu ý:** Ở phép chia:
  - Nếu cả số chia và số bị chia đều là số nguyên, thì kết quả chỉ lấy phần nguyên.
  - Nếu số chia hoặc số bị chia là số thập phân, thì kết quả là số thập phân.

Ví dụ:

```
int a = 10, b = 3;
double c1 = (double)(a/b);
double c2 = (double)a/b;
double c3 = a/(double)b;
printf("%lf %lf %lf", c1, c2, c3);
```

2) Toán tử gán mở rộng.

Ký hiệu	Ví dụ	Tương đương với
+=	a += b;	a = a + b;
-=	a -= b;	a = a - b;
*=	a *= b;	a = a * b;
/=	a /= b;	a = a / b;
%=	a %= b;	a = a % b;

3) Toán tử quan hệ.

- Xét a = 1, b = 1, c = 2 cho các ví dụ bên dưới.

Ký hiệu	Mô tả	Ví dụ

==	So sánh bằng.	a == b trả về 1. a == c trả về 0.
!=	So sánh khác.	a != c trả về 1. a != b trả về 0.
>	So sánh lớn hơn.	c > b trả về 1. b > c trả về 0.
<	So sánh bé hơn.	a < c trả về 1. c < a trả về 0.
>=	So sánh lớn hơn hoặc bằng.	c >= b trả về 1. b >= a trả về 1. b >= c trả về 0.
<=	So sánh bé hơn hoặc bằng.	b <= c trả về 1. b <= a trả về 1. c <= b trả về 0.

4) Toán tử logic.

- Xét a = true, b = false cho các ví dụ bên dưới.

Ký hiệu	Mô tả	Ví dụ
&&	Toán tử AND	(a && b) trả về false.
||	Toán tử OR	(a || b) trả về true.
!	Toán tử NOT	!b trả về true. !a trả về false.

IV. Mệnh đề if-else.

- Cú pháp:

1) if (<Điều kiện>) {

<Các câu lệnh thực hiện khi điều kiện đúng>;

}

2) if (<Điều kiện>) {

<Các câu lệnh thực hiện khi điều kiện đúng>;

} else{

<Các câu lệnh thực hiện khi điều kiện sai>;

}

3) if (<Điều kiện số 1>) {

<Các câu lệnh được thực hiện khi điều kiện số 1 đúng>;

} else if (<Điều kiện số 2>){

<Các câu lệnh được thực hiện khi điều kiện số 2 đúng>;

} else {

<Các câu lệnh được thực hiện khi các điều kiện trên sai>;

}

- **Lưu ý:** Nếu trong cặp dấu ngoặc nhọn chỉ có câu 1 lệnh duy nhất, ta có thể bỏ cặp dấu ngoặc nhọn.
- Toán tử 3 ngôi: là một dạng rút gọn của cấu trúc trúc re nhánh số 2 (ở bên trên). Có cú pháp:

(<Điều kiện>) ? <Giá trị hoặc lệnh số 1> : <Giá trị hoặc lệnh số 2>;

Bài tập 1: Nhập vào 1 số nguyên.

Nếu là số dương, in ra màn hình "Positive".

Nếu là sỗ âm, in ra màn hình "Negative".

Nếu là 0, in ra màn hình "Zero".

```
    int n;
    scanf("%d", &n);

//  Cách 1:
    if(n > 0){
        printf("Positive");
    } else if(n < 0){
        printf("Negative");
    } else{
        printf("Zero");
    }

//  Cách 2: không nên dùng
//  if(n > 0) printf("Positive");
//  else if(n < 0) printf("Negative");
//  else printf("Zero");
```

Bài tập 2: Nhập vào 1 số nguyên.

Nếu là số chẵn, in ra màn hình "Even number".

Nếu là sỗ lẻ, in ra màn hình "Odd number".

Làm bằng nhiều cách nhất có thể.

```c
    int n;
    scanf("%d", &n);

//  Cách 1:
//  if(n % 2 == 0){
//      printf("Even numberd");
//  } else{
//      printf("Odd number");
//  }

//  Cách 2
//  if(n % 2 == 0) printf("Even number");
//  else printf("Odd number");

//  Cách 3
//  (n % 2 == 0) ? printf("Even number") : printf("Odd number");

//  Cách 4
    printf((n % 2 == 0) ? "Even number" : "Odd number");
```

Bài tập 3: Nhập vào 2 số nguyên a và b.

Nếu a >= b, in ra màn hình "**[a]** >= **[b]**"

Nếu a < b, in ra màn hình "**[a]** < **[b]**".

```c
   int a, b;
    scanf("%d%d", &a, &b);

//  Cách 1:
//  if(a >= b){
//      printf("%d >= %d", a, b);
//  } else{
//      printf("%d < %d", a, b);
//  }

//  Cách 2
//  if(a >= b) printf("%d >= %d", a, b);
//  else printf("%d < %d", a, b);

//  Cách 3
//  (a >= b) ? printf("%d >= %d", a, b) : printf("%d < %d", a, b);

//  Cách 4
    printf((a >= b) ? "%d >= %d" : "%d < %d", a, b);
```

Bài tập 4: Nhập vào 2 số nguyên a và b.

Nếu a != 0 và b != 0, in ra màn hình "a != 0 and b != 0".

Nếu không thì in ra màn hình "a = 0 or b = 0".

```c
    int a, b;
    scanf("%d%d", &a, &b);

//    Cách 1:
//    if(a != 0 && b != 0){
//      printf("a != 0 and b != 0");
//  }else{
//      printf("a = 0 or b = 0");
//  }

//  Cách 2:
//  if(a != 0 && b != 0) printf("a != 0 and b != 0");
```

```
//  else printf("a = 0 or b = 0");

//  Cách 3:
//  (a != 0 && b != 0) ? printf("a != 0 and b != 0") : printf("a = 0 or b = 0");

//  Cách 4:
    printf((a != 0 && b != 0) ?"a != 0 and b != 0" : "a = 0 or b = 0");
```

Bài tập 5: Nhập vào 1 số nguyên n.

Nếu số đó thuộc đoạn [0, 100] thì in ra "**[n]** is in the range [0, 100]".

Nếu không thì in ra "**[n]** is not in the range [0, 100]".

```
    int n;
    scanf("%d", &n);

//  Cách 1:
//  if(n >= 0 && n <= 100){
//      printf("%d is in the range [0, 100]", n);
//  } else{
//      printf("%d is not in the range [0, 100]", n);
//  }

//  Cách 2:
//  if(n >= 0 && n <= 100) printf("%d is in the range [0, 100]", n);
//  else printf("%d is not in the range [0, 100]", n);

//  Cách 3:
//  (n >= 0 && n <= 100) ? printf("%d is in the range [0, 100]", n) : printf("%d is not i
n the range [0, 100]", n);

//  Cách 4:
    printf((n >= 0 && n <= 100) ? "%d is in the range [0, 100]" : "%d is not in the range
[0, 100]", n);
```

Bài tập 6: Nhập vào 3 số nguyên a, b, c. In ra số lớn nhất.

```
    int a, b, c;
    scanf("%d%d%d", &a, &b, &c);

    int max = a;
    if(b > a && b > c){
        max = b;
    }
    if(c > a && c > b){
        max = c;
    }
    printf("Max = %d", max);
```

Bài tập 7: Nhập vào 3 số nguyên a, b, c. In ra số nhỏ nhất.

```
    int a, b, c;
    scanf("%d%d%d", &a, &b, &c);
    int min = a;
    if(b < a && b < c){
        min = b;
    }
    if(c < a && c < b){
        min = c;
    }
    printf("\nMin = %d", min);
```

V. Vòng lặp for.

- Cú pháp:

for (<Khởi tạo giá trị biến lặp>; <Điều kiện lặp>; <Cập nhất biến lặp>) {

<Các câu lệnh thực hiện khi điều kiện lặp còn đúng>;

}

- Nên tạo biến lặp trong cặp dâu ngoặc tròn của vòng lặp for thay vì tạo ở phía trước. Ví dụ:

```
for (int i = 0; i <= n; i++) {
    printf(“%d ”, i);
}
```

- Nếu chương trình thông báo lỗi thì làm như sau: **Project → Project Options… → Compiler → Code Generation →** Ở mục **Language Standard (-std)** chọn **ISO C99 →** chọn **OK**.
- **Lưu ý:** Biến lặp là biến local, chỉ tồn tại trong vòng lặp, khi vòng lặp kết thúc thì biến lặp cũng sẽ mất.

Ví dụ 1: Nhập vào 1 số nguyên n. In ra các số từ 0 đến n, mỗi số cách nhau một khoản trắng.

```
int n;
scanf(“%d”, &n);
for (int i = 0; i <= n; i++) {
    printf(“%d ”, i);
}
```

Ví dụ 2: Nhập vào 2 số nguyên a, b (a < b). In ra các số từ a đến b, mỗi số cách nhau bởi một khoản trắng.

```
int a, b;
scanf("%d%d", &a, &b);

for (int i = a; i <= b; i++) {
    printf("%d ", i);
}
```

Bài tập 1: Nhập vào một số nguyên n với n > -5. In ra màn hình các số từ n tới -5.

```
int n;
scanf("%d", &n);

for(int i = n; i >= -5; i--){
    printf("%d ", i);
}
```

Bài tập 2: Nhập vào một số nguyên n. In ra màn hình các số từ -n tới n và từ n tới –n.

```
int n;
scanf("%d", &n);

for(int i = -n; i <= n; i++){
    printf("%d ", i);
}
```

```
printf("\n");

for(int i = n; i >= -n; i--){
    printf("%d ", i);
}
```

Bài tập 3: Nhập vào số nguyên n. Tính tổng các số từ 0 đến n rồi in ra màn hình.

```
int n;
scanf("%d", &n);

int sum = 0;

for(int i = 1; i <= n; i++){
    sum = sum + i; // sum += i;
}

printf("%d", sum);
```

Bài tập 4: Nhập vào số nguyên n. Tính n! và in ra màn hình.

```
int n;
scanf("%d", &n);

int s = 1;

for(int i = 1; i <= n; i++){
    s = s * i; // s *= i;
}

printf("%d", s);
```

Bài tập 5: Nhập vào số nguyên a, x. In ra màn hình a mũ x.

```
int a, x;
scanf("%d%d", &a, &x);

int s = 1;

for(int i = 0; i < x; i++){
    s = s * a; // s *= a;
}

printf("%d", s);
```

Bài tập 6: Nhập vào số nguyên n. In ra màn hình tất cả các số chẵn trong đoạn [0,n].

```
int n;
scanf("%d", &n);

for(int i = 0; i <= n; i++){
    if(i % 2 == 0) printf("%d ", i);
}
```

Bài tập 7: Nhập vào số nguyên n. In ra màn các số chia hết cho 3 trong đoạn [0,n].

```
int n;
scanf("%d", &n);

for(int i = 0; i <= n; i++){
    if(i % 3 == 0) printf("%d ", i);
}
```

Bài tập 8: Nhập vào số nguyên n. Tính tổng các số lẻ trong đoạn [1,n] rồi in ra màn hình.

```
    int n;
    scanf("%d", &n);

    int s = 0;
    for(int i = 1; i <= n; i++){
        if(i % 2 == 1) s = s + i; // s += i;
    }
    printf("S = %d", s);
```

Bài tập 9: Nhập vào số tự nhiên n. In ra màn hình các ước của n.

```
    int n;
    scanf("%d", &n);

    int s = 0;
    for(int i = 1; i <= n; i++){
        if(n % i == 0) printf("%d ", i);
    }
```

VI. Vòng lặp while và do-while.

- Cú pháp vòng lặp while:

  while (<Điều kiện lặp>) {

  <Các câu lệnh thực hiện khi điều kiện lặp còn đúng>;

  }

- Cú pháp vòng lặp do-while:

  do {

  <Các câu lệnh thực hiện khi điều kiện lặp còn đúng>;

  } while (<Điều kiện lặp>);

- So sánh while và do-while:
  - Vòng lặp while: nếu **<Điều kiện lặp>** đúng thì mới bắt đầu thực hiện các lệnh bên trong.
  - Vòng lặp do-while: thực hiện các lệnh bên trong rồi mới xét tới **<Điều kiện lặp>**, nếu **<Điều kiện lặp>** đứng thì tiếp tục thực hiện tiếp các lệnh bênh trong.

Ví dụ 1: Nhập 1 số nguyên n. In ra màn hình các số chẵn trong đoạn [n, 100].

```
int n;
scanf("%d", &n);

while(n <= 100){
    if(n % 2 == 0){
        printf("%d ", n);
    }
    n++;
}
```

Ví dụ 2: Nhập vào một số nguyên n. Nếu n thỏa điều kiện 0 < n < 100 thì n ra màn hình, nếu không thì yêu cầu nhập lại.

```
int n;
do{
    printf("Moi nhap n: ");
    scanf("%d", &n);
} while(n <= 0 || n >= 100);

printf("%d", n);
```

Bài tập 1: Nhập vào 2 số nguyên dương a và b. Tìm bội chung nhỏ nhất của a và b:

1) Bằng cách thông thường.

```
    int a, b;
    scanf("%d%d", &a, &b);

    int x = a;
    if(b > a) x = b;
    while(x % a != 0 || x % b != 0){
        x++;
    }
    printf("%d", x);
```

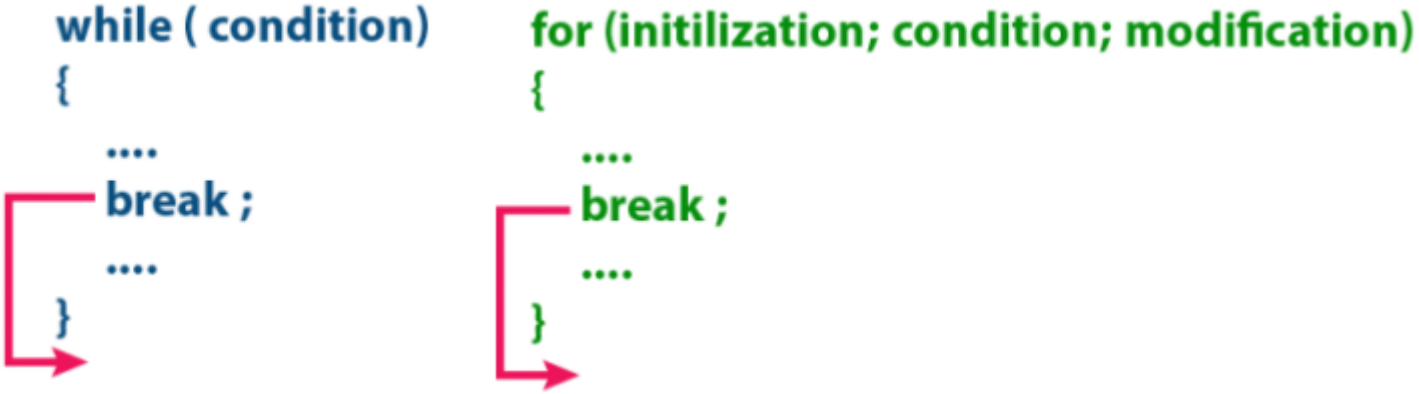
2) Thông qua ước chung lớn nhất bằng thuật toán Euclid.

Link tham khảo thuật toán Euclid: https://www.howkteam.vn/course/viet-ham-sap-xep-cac-phan-tu-le-tang-dan/tim-uoc-so-chung-lon-nhat-va-boi-so-chung-nho-nhat-cua-a-va-b-1444

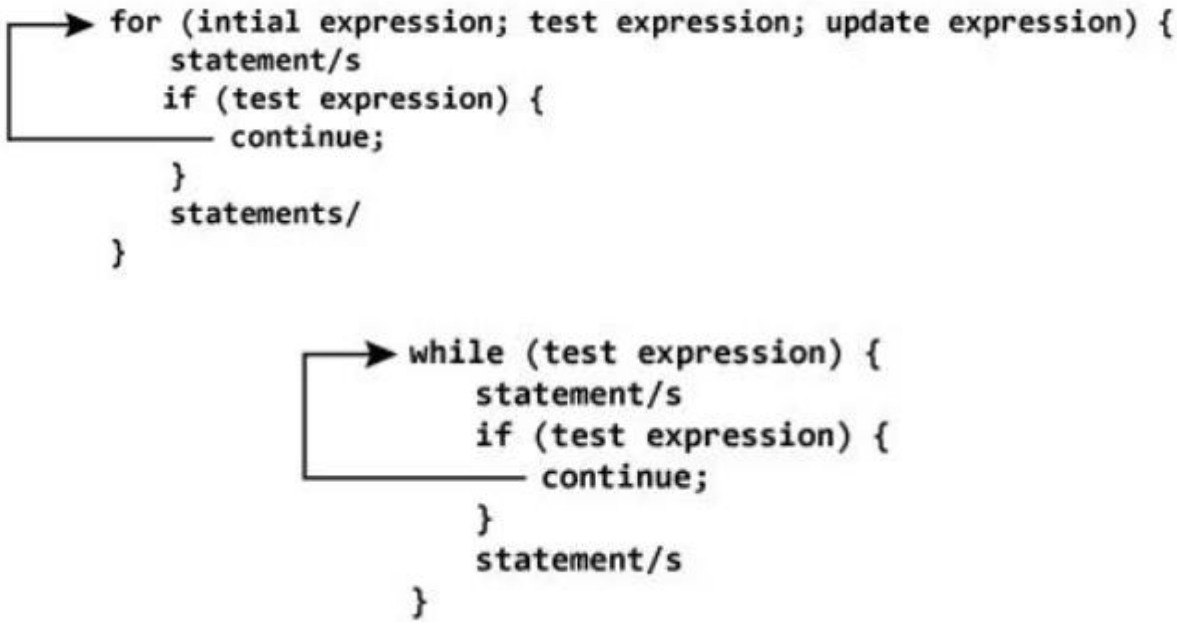
```
    int a, b;
    scanf("%d%d", &a, &b);

    int c = a, d = b;
    while(c != d){
        if(c > d){
            c = c - d;
        } else{
            d = d - c;
        }
    }

    printf("\nUCLN = %d", c);
    printf("\nBCNN = %d", a*b/c);
```

VIII. Lệnh break và continue.

- Trong vòng lặp, khi gặp lệnh break, chương trình sẽ ngay lập tức kết thúc vòng lặp.



- Trong vòng lặp, khi gặp lệnh continue, chương trình sẽ bỏ qua phần lệnh còn lại trong lần lặp đó và tiến đến lần lặp tiếp theo.



Bài tập 1: Điền vào chỗ … để đoạn chương trình sau in ra màn hình các số từ 0 đến 10.

```
#include <stdio.h>

int main() {
    for(int i = 0; i <= 100; i++){
        . . .
        printf("%d ", i);
    }

    return 0;
}
```

Đáp án:

```
#include <stdio.h>
```

```c
int main() {
    for(int i = 0; i <= 100; i++){
        if(i > 10){
            break;
        }
        printf("%d ", i);
    }

    return 0;
}
```

Bài tập 2: Điền vào chỗ … để đoạn chương trình sau in ra màn hình các số lẻ trong đoạn [0,100].

1) Nếu điều kiện của vòng lặp while không phải là i<100 mà là i<=100 thì chuyện gì sẽ xảy ra? Tại sao?

2) Nếu trong vòng lặp, câu lệnh ***i++;*** không đươc để ở đầu mà để ở ngay sau lệnh ***printf("%d ", i);*** thì chuyện gì sẽ xảy ra? Tại sao?

3) Giữ nguyên trường hợp ở câu 2, nhưng khởi tạo giá trị ban đầu cho i là 1 thì chuyện gì sẽ xảy ra? Tại sao?

```c
#include <stdio.h>

int main() {
    int i = 0;
    while(i < 100){
        i++;
        . . .
        printf("%d ", i);
    }

    return 0;
}
```

Đáp án:

```c
#include <stdio.h>

int main() {
    int i = 0;
    while(i < 100){
        i++;
        if(i % 2 == 0){
            continue;
        }
        printf("%d ", i);
    }

    return 0;
}
```

1) Chương trình sẽ in ra tới số 101 rồi mới dừng lại, thay vì là số 99. Nguyên nhân do ở lần lập i có giá trị là 100 (là số chẵn) lệnh continue sẽ đc thực hiện → chương trình sẽ quay về xét điều kiện lặp → điều kiện lặp vẫn thỏa → ở lần lặp tiếp theo i được tăng lên 1 đơn vị thành 101 và được in ra màn hình → chương trình quay về xét điều kiện lặp → điều kiện lặp không thỏa → thoát vòng lặp.

2) Vòng lặp sẽ thành vòng lặp vô hạn và không in ra màn hình bất kỳ số nào. Nguyên nhân do giá trị ban đầu của i là 0 (là số chẵn) → lệnh continue sẽ được thực hiện và bỏ qua lệnh ***printf("%d ", i);*** và ***i++;*** → giá trị của I không thể bị thay đổi → trở thành vòng lặp vô hạng.

3) Vòng lặp sẽ thành vòng lặp vô hạn và chỉ in ra số 1. Nguyên nhân do giá trị ban đầu của i là 1 (là số lẻ) → lệnh continue không được thực hiện → giá trị 1 được in ra và i được tăng lên 1 đơn vị thành 2 → rơi vào trường hợp giống câu 2.

➔ **Cần xem xét kỹ điều kiện lặp khi dùng vòng lặp.**