# CHƯƠNG 3: VÒNG LẶP

# Bài 1: VÒNG LẶP FOR

### 1. Khái niệm

Mục đích của vòng lặp là để giải quyết các công việc có sự lặp đi lặp lại nhiều lần bằng những dòng code ngắn gọn nhất. Được sử dụng để lặp đi lặp lại 1 khối code có số lần lặp xác định.

Một số đặc điểm của vòng lặp for:

* Vòng lặp for có cú pháp phức tạp hơn, nhưng ngắn gọn hơn các vòng lặp while hay do-while khi sử dụng.
* Vòng lặp for hoàn toàn có thể thay thế vòng lặp while.
* Vòng lặp for thường được sử dụng cho các trường hợp biết trước số lần lặp lại khối công việc.

### 2. Cú pháp

|  |
| --- |
| for (khởi tạo giá trị biến lặp; điều kiện lặp; cập nhật biến lặp)  {   // các lệnh cần lặp  } |

Vòng lặp for được định nghĩa bởi từ khóa for và được chia làm 3 phần chính, **mỗi phần được ngăn cách bởi dấu chấm phẩy:**

* **Phần khởi tạo biến lặp**

Khác với vòng lặp while và do-while, biến vòng lặp có thể được khai báo và khởi tạo giá trị ngay bên trong phần khởi tạo của vòng lặp **for**.

Phần khởi tạo biến được thực thi đầu tiên và chỉ thực thi 1 lần duy nhất trong vòng lặp **for**.

* **Điều kiện lặp**

Phần này tương tự như vòng lặp **while**, khối lệnh của vòng lặp for sẽ được thực hiện nếu biểu thức điều kiện cho giá trị đúng. Vòng lặp **for** kiểm tra biểu thức điều kiện trước khi thực hiện khối lệnh.

* **Cập nhật biến lặp**

Phần này sẽ được thực thi cuối mỗi lần lặp, sau khi khối lệnh của vòng lặp **for** được thực thi. Phần này thường chịu trách nhiệm thay đổi giá trị biến vòng lặp được sử dụng trong biểu thức điều kiện (nhằm tránh tình trạng lặp vô hạn). Sau khi thực thi xong phần cập nhật biến vòng lặp, chương trình quay trở lại đánh giá biểu thức điều kiện của vòng lặp **for** và cứ như thế.

Vậy chúng ta rút ra được các bước thực hiện vòng lặp **for** qua lưu đồ như sau:

Diagram

Description automatically generated

• B1: Khởi tạo giá trị biến lặp lần duy nhất

• B2: Kiểm tra điều kiện lặp, nếu sai chuyển sang B5

• B3: Thực hiện nội dung trong thân vòng lặp

• B4: Cập nhật giá trị lặp và quay lại B2

• B5: Kết thúc vòng lặp.

VD1: Viết 1 chương trình in ra câu “Toi se hoc gioi C “10 lần

|  |
| --- |
| #include<stdio.h>  int main(){      for (int i = 0; i < 10; i++){          printf("Toi se hoc gioi C\n");      }  } |

Kết quả

|  |
| --- |
| Toi se hoc gioi C  Toi se hoc gioi C  Toi se hoc gioi C  Toi se hoc gioi C  Toi se hoc gioi C  Toi se hoc gioi C  Toi se hoc gioi C  Toi se hoc gioi C  Toi se hoc gioi C  Toi se hoc gioi C |

**Chú ý: Có dấu chấm phẩy sau lệnh for (i=0; i<10; i++); → các lệnh thuộc vòng lặp for sẽ không được thực hiện.**

VD2. Viết 1 chương trình nhập vào n số nguyên và tính tổng n số đó

|  |  |
| --- | --- |
| 1 | #include<stdio.h>  int main (){      int tong = 0;      int a;      int n;      printf("Nhap so luong phan tu:");      scanf("%d",&n);      for (int i = 1; i <= n; i++){          printf("Nhap vao so thu %d:", i);          scanf ("%d",&a);          tong += a;      }      printf ("Tong la: %d",tong);      return 0;  } |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 16 |

Kết quả in ra màn hình

|  |
| --- |
| Nhap so luong phan tu:6  Nhap vao so thu 1:2  Nhap vao so thu 2:1  Nhap vao so thu 3:4  Nhap vao so thu 4:6  Nhap vao so thu 5:7  Nhap vao so thu 6:4  Tong la: 24 |

Trong vòng lặp for có sử dụng từ 2 lệnh trở lên, nhớ sử dụng cặp ngoặc { } để bọc các lệnh đó lại.

VD3: Viết chương trình nhập vào số nguyên n. Tính tổng các giá trị lẻ từ 0 đến n.

|  |  |
| --- | --- |
| 1 | #include<stdio.h> |
| 2 | int main(){ |
| 3 | int n; |
| 4 | int tong = 0; |
| 5 | printf("Nhap n:"); |
| 6 | scanf("%d",&n); |
| 7 | for(int i = 0; i <= n; i++){ |
| 8 | if (i %2 == 1){ |
| 9 | tong +=i; |
| 10 | } |
| 11 | } |
| 12 | printf ("Tong cac so le tu 0 toi n la: %d ",tong); |
| 13 | return 0; |
| 14 | } |

Bạn thay các dòng từ 7 đến 11 bằng câu lệnh: for( tong=0, i = 1; i <= n; tong = tong+i, i+=2);

Chạy lại chương trình, quan sát và nhận xét kết quả vẫn y hệt như chương trình trên

Nhận xét: Bạn có thể viết gộp các lệnh trong thân for vào trong lệnh for. Tuy nhiên, khi lập trình bạn nên viết lệnh for có đủ 3 biểu thức đơn và các lệnh thực hiện trong thân for mỗi lệnh một dòng để sau này có thể đọc lại dễ hiểu, dễ sửa chữa.

### 3. Lược bỏ 1 số thành phần trong vòng lặp for

##### 3.1 Lược bỏ biểu thức khởi tạo giá trị biến lặp

-Nếu bỏ qua biểu thức khởi tạo, chương trình sẽ sử dụng giá trị của biến được khai báo trước đó hoặc giá trị mặc định của kiểu dữ liệu.

VD5: Tính tổng các số từ 1 tới n

|  |
| --- |
| #include <stdio.h>  int main() {      int i = 1, tong = 0;      int n;      printf("Nhap n:");      scanf("%d",&n);      for (; i <= n; i++) {          tong += i;      }      printf("Tong tu 1 den %d la: %d", n, tong);      return 0;  } |

##### 3.2 Lược bỏ biểu thức điều kiện lặp

Nếu bỏ qua biểu thức điều kiện, vòng lặp sẽ lặp vô tận nếu không có câu lệnh break trong khối lệnh.

VD6: In ra các số từ 0 tới 10

|  |
| --- |
| #include <stdio.h>  int main() {      int i;      for (i = 0; ; i++) {          if (i > 10) {              break;          }          printf("%d ", i);      }      return 0;  } |

##### 3.3 Lược bỏ biểu thức cập nhật biến lặp

Nếu thiếu biểu thức tăng/giảm, chúng ta có thể đặt nó trong khối lệnh của vòng lặp.

VD7. Tính tổng các số từ 1 tới n

|  |
| --- |
| #include <stdio.h>  int main() {      int n ;      int tong = 0;      printf("Nhap n:");      scanf("%d",&n);      for (int i = 1; i <= n;) {          tong += i;          i++;      }      printf("Tong tu 1 den %d la: %d", n, tong);      return 0;  } |

##### 3.4 Lược bỏ biểu thức khởi tạo giá trị biến lặp và biểu thức cập nhật biến lặp

Nếu vắng cả biểu thức khởi tạo và biểu thức tăng/giảm, ta có thể dùng một biến trước đó để khởi tạo giá trị ban đầu và sử dụng biến đó trong vòng lặp, sau đó tăng/giảm giá trị của biến trong khối lệnh của vòng lặp

VD8. Tính tổng các số từ 1 tới n

|  |
| --- |
| #include <stdio.h>  int main() {      int n;      printf("Nhap n:");      scanf("%d",&n);      int i = 1;      int tong = 0;      for (; i <= n;) {          tong += i;          i++;      }      printf("Tong tu 1 den %d la: %d", n, tong);      return 0;  } |

##### 3.5 Lược bỏ biểu thức điều kiện lặp và biểu thức khởi tạo giá trị biến lặp

Nếu lược bỏ cả biểu thức điều kiện lặp và biểu thức khởi tạo giá trị biến lặp trong vòng lặp for, ta có thể dùng một biến trước đó để khởi tạo giá trị ban đầu và sử dụng biến đó trong vòng lặp và ta cần sử dụng một vòng lặp vô hạn và cần phải sử dụng lệnh break để thoát khỏi vòng lặp.

VD9. In ra các số từ 0 đến 10

|  |
| --- |
| #include <stdio.h>  int main() {      int i = 0;      for (;;i++) {          if (i > 10) {              break;          }          printf("%d ", i);      }      return 0;  } |

##### 3.6 Lược bỏ biểu thức điều kiện lặp và biểu thức cập nhật biến lặp

Nếu lược bỏ cả biểu thức điều kiện lặp và biểu thức cập nhật biến lặp trong vòng lặp for, thì ta cần sử dụng một vòng lặp vô hạn và cần phải sử dụng lệnh break để thoát khỏi vòng lặp. tăng/giảm giá trị của biến trong khối lệnh của vòng lặp

VD10. In ra các số từ 0 đến 10

|  |
| --- |
| #include <stdio.h>  int main() {      for (int i = 0; ; ) {          if (i > 10) {              break;          }          printf("%d ", i);          i++;      }      return 0;  } |

##### 3.7 Vòng lặp for vắng mặt cả 3 biểu thức

Nếu vòng lặp for vắng mặt cả 3 biểu thức (khởi tạo, điều kiện lặp và cập nhật) thì sẽ tạo ra một vòng lặp vô hạn và lệnh trong khối lệnh sẽ được thực thi vô tận cho đến khi có một lệnh break hoặc return được sử dụng để thoát khỏi vòng lặp.

VD11. In ra các số từ 0 đến 10

|  |
| --- |
| #include <stdio.h>  int main() {      int i = 0;      for (;;) {          if (i > 10) {              break;          }          printf("%d ", i);          i++;      }      return 0;  } |

# BÀI 2: VÒNG LẶP WHILE

#### 1.Khái niệm

Được sử dụng để lặp đi lặp lại 1 khối lệnh mà không biết trước số lần lặp

Vòng lặp thực hiện lặp lại trong khi biểu thức còn đúng

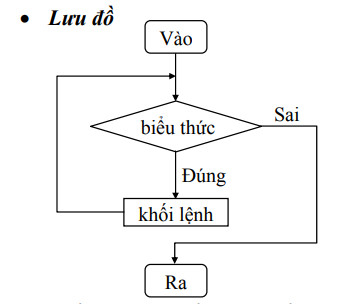
#### 2.Cú pháp

|  |
| --- |
| **while (biểu thức)**  **{**  **// Khối lệnh**  **}** |

Từ khóa while phải viết bằng chữ thường

Nếu khối lệnh bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

**Lưu đồ**



Trước tiên biểu thức được kiểm tra

**Nếu sai thì**

Kết thúc vòng lặp while

(khối lệnh không được thi hành 1 lần nào)

**Nếu đúng**

Thực hiện khối lệnh

Lặp lại kiểm tra biểu thức

+ Biểu thức: có thể là một biểu thức hoặc nhiều biểu thức con. Nếu là nhiều biểu thức con thì cách nhau bởi dấu phẩy (,) và tính đúng sai của biểu thức được quyết định bởi biểu thức con cuối cùng.

+ Trong thân while (khối lệnh) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.

+ Trong thân while có thể sử dụng lệnh continue để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).

+ Muốn thoát khỏi vòng lặp while tùy ý có thể dùng các lệnh break, goto, return như lệnh for

VD12. Viết 1 chương trình in ra các số từ 1 -> n, n được nhập từ bàn phím

|  |
| --- |
| #include <stdio.h>  int main(){      int n;      int i = 0;      printf ("Nhap n:");      scanf ("%d",&n);      while (i++ < n){          printf("%d ",i);      }      return 0;  } |

Kết quả

|  |
| --- |
| Nhap n:20  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 |

VD13. Viết 1 chương trình in ra chuỗi số giảm dần từ 10 về 0

|  |
| --- |
| #include <stdio.h>  int main(){      int n = 11;      while (n--){          printf("%d ", n);      }      printf("\n");  return 0;  } |

Kết quả

|  |
| --- |
| 10 9 8 7 6 5 4 3 2 1 0 |

# BÀI 3: VÒNG LẶP DO … WHILE

#### 1.Khái niệm

do…while trong C là vòng lặp mà xử lý trong nó được thực thi ít nhất một lần. Giống như vòng lặp While thì vòng lặp do…while cũng sẽ thực thi các lệnh trong nó khi biểu thức điều kiện của nó là True. Tuy nhiên khác với vòng lặp while sẽ tiến hành đánh giá biểu thức điều kiện xem là True hay không rồi chạy các lệnh trong vòng lặp, thì vòng lặp do…while sẽ chạy các lệnh trong nó trước, sau đó mới tiến hành đánh giá biểu thức điều kiện để quyết định có lặp các lần lặp tiếp theo hay không.

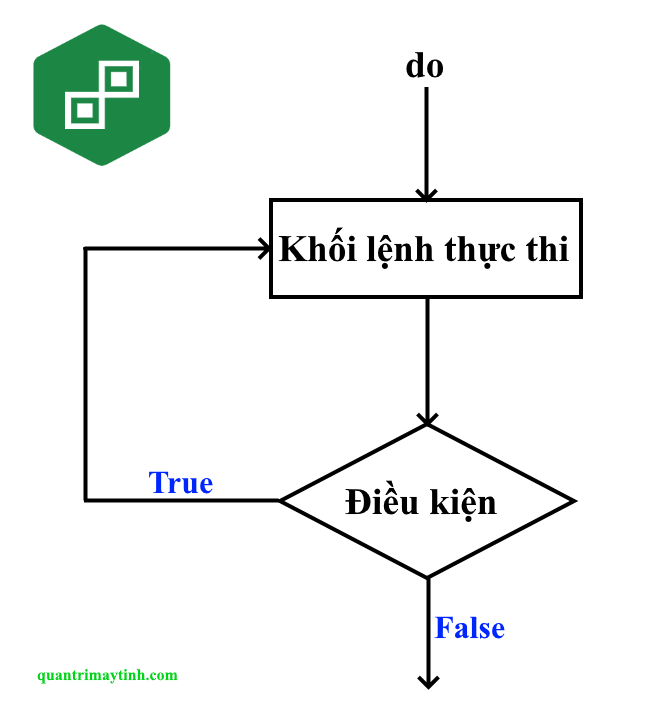
#### 2. Cú pháp

|  |
| --- |
| **do**  **{**  **// Khối lệnh**  **}**  **while (biểu thức);** |

Từ khóa **do**, **while** phải viết bằng chữ thường

Nếu khối lệnh bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

**Lưu đồ**



-Thực hiện khối lệnh

-Kiểm tra biểu thức

**+Nếu đúng**

thì lặp lại thực hiện khối lệnh

**+Nếu sai**

thì kết thúc vòng lặp (khối lệnh được thi hành 1 lần)

+ Biểu thức: có thể là một biểu thức hoặc nhiều biểu thức con. Nếu là nhiều biểu thức con thì cách nhau bởi dấu phẩy (,) và tính đúng sai của biểu thức được quyết định bởi biểu thức con cuối cùng.

+ Trong thân do…while (khối lệnh) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.

+ Trong thân do…while có thể sử dụng lệnh **continue** để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).

+ Muốn thoát khỏi vòng lặp do…while tùy ý có thể dùng các lệnh **break, goto, return.**

VD14. Viết chương trình kiểm tra passwork

|  |
| --- |
| /\* Chuong trinh kiem tra mat khau \*/  #include <stdio.h>  # define PASSWORD 12345  int main() {      int in;      do {          printf("Nhap vao password: ");          scanf("%d", &in);      }      while (in != PASSWORD);  } |

Kết quả

|  |
| --- |
| Nhap vao password: 1234  Nhap vao password: 12345 |

# BÀI 4: VÒNG LẶP LỒNG NHAU

Cũng có thể đặt một vòng lặp bên trong một vòng lặp khác. Đây được gọi là vòng lặp lồng nhau. "Vòng lặp bên trong" sẽ được thực hiện một lần cho mỗi lần lặp lại "vòng lặp bên ngoài”

VD15: Vẽ hình chữ nhật đặc bằng các dấu '\*'

|  |
| --- |
| #include<stdio.h>  int main(){      int a,b;      printf("Nhap vao chieu dai HCN: ");      scanf("%d",&a);      printf("Nhap vao chieu rong HCN: ");      scanf("%d",&b);      for (int i = 1; i <= b; i++){          for (int j = 1; j <= a; j++){              printf("\*");          }          printf("\n");      }      return 0;  } |

Kết quả

|  |
| --- |
| Nhap vao chieu dai HCN: 9  Nhap vao chieu rong HCN: 5  \*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\* |

**-Các lệnh lặp for, while, do…while có thể lồng vào chính nó, hoặc lồng vào lẫn nhau. Nếu không cần thiết không nên lồng vào nhiều cấp dễ gây nhầm lẫn khi lập trình cũng như kiểm soát chương trình.**

So sánh sự khác nhau của các vòng lặp

- Vòng lặp for thường sử dụng khi biết được số lần lặp xác định.

- Vòng lặp thường while, do…while sử dụng khi không biết rõ số lần lặp.

- Khi gọi vòng lặp while, do…while, nếu biểu thức sai vòng lặp while sẽ không được thực hiện lần nào nhưng vòng lặp **do…while** thực hiện được 1 lần. )

-> **Số lần thực hiện ít nhất của while là 0 và của do…while là 1**

# BÀI 5: CÂU LỆNH ĐIỀU KHIỂN

# 1. Từ khóa break

Thông thường lệnh break dùng để thoát khỏi vòng lặp không xác định điều kiện dừng hoặc bạn muốn dừng vòng lặp theo điều kiện do bạn chỉ định. Việc dùng lệnh break để thoát khỏi vòng lặp thường sử dụng phối hợp với lệnh if. Lệnh break dùng trong for, while, do…while, switch. Lệnh break thoát khỏi vòng lặp chứa nó.

VD:16: In những số < 10 trong khoảng [0;20]

|  |
| --- |
| #include <stdio.h>  int main(){      int i;      for (i = 0; i < 20; i++){          if(i>=10)              break;          printf("%d ",i);      }      return 0;  } |

Kết quả

|  |
| --- |
| 0 1 2 3 4 5 6 7 8 9 |

# 2. Từ khóa continue

Được dùng trong vòng lặp for, while, do…while. Khi lệnh continue thi hành quyền điều khiển sẽ trao qua cho biểu thức điều kiện của vòng lặp gần nhất. Nghĩa là lộn ngược lên đầu vòng lặp, tất cả những lệnh đi sau trong vòng lặp chứa **continue** sẽ bị bỏ qua không thi hành.

VD17: In ra các chữ số lẻ từ 1 đến 10

|  |
| --- |
| #include <stdio.h>    /\*     In ra các chữ số lẻ từ 1 đến 10.   \*/    int main(){      for(int i = 0; i <= 10; i++){          if(i % 2 == 0){              continue;          }          printf("%d ", i);      }  } |

Kết quả

|  |
| --- |
| 1 3 5 7 9 |

# 3.Từ khóa goto

Câu lệnh goto trong C cung cấp một bước nhảy vô điều kiện từ 'goto' đến một câu lệnh có nhãn trong cùng một hàm.

Chú ý: Việc sử dụng câu lệnh goto không được khuyến khích sử dụng trong bất kỳ ngôn ngữ lập trình nào vì nó rất khó để theo dõi luồng điều khiển của chương trình, làm cho chương trình khó hiểu và khó bảo trì.