## Bài 58: Con trỏ hàm trong C++ (Function pointers)

**Dẫn nhập**

Ở bài học trước, mình đã chia sẻ cho các bạn về [HÀM CÓ ĐỐI SỐ MẶC ĐỊNH TRONG C++ (Default arguments)](https://www.howkteam.vn/course/khoa-hoc-lap-trinh-c-can-ban/ham-co-doi-so-mac-dinh-trong-c-default-arguments-3902). Đối số mặc định rất hữu ích để chỉ định giá trị mặc định cho các tham số, và thường được sử dụng trong C++

Trong bài học này, chúng ta sẽ cùng tìm hiểu về **Con trỏ hàm trong C++ (Function pointers)**.

**Nội dung**

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về:

* [CƠ BẢN VỀ HÀM VÀ GIÁ TRỊ TRẢ VỀ (Basics of Functions and Return values)](https://www.howkteam.vn/course/khoa-hoc-lap-trinh-c-can-ban/co-ban-ve-ham-va-gia-tri-tra-ve-basic-of-functions-and-return-values-1339)
* [CON TRỎ CƠ BẢN TRONG C++ (Pointers)](https://www.howkteam.vn/course/khoa-hoc-lap-trinh-c-can-ban/con-tro-co-ban-trong-c-2714)

Trong bài ta sẽ cùng tìm hiểu các vấn đề:

* Đặt vấn đề
* Con trỏ hàm là gì?
* Gán địa chỉ của hàm cho con trỏ hàm
* Gọi một hàm bằng con trỏ hàm
* Truyền con trỏ hàm vào hàm dưới dạng đối số
* Đối số mặc định của tham số hàm kiểu con trỏ hàm
* std::function trong C++11
* Khai báo con trỏ hàm với từ khóa auto trong C++11

**Đặt vấn đề**

Cùng xem ví dụ sau:

* [**C++**](https://howkteam.vn/learn)

**1 int** func(**int** a)

2 {

3 // do something

4 **return** a;

5 }

6

**7 int** main()

8 {

9 **cout** << func << '\n'; // in địa chỉ hàm func trong bộ nhớ

10 **cout** << func(1) << '\n'; // đi đến địa chỉ hàm func và thực thi hàm

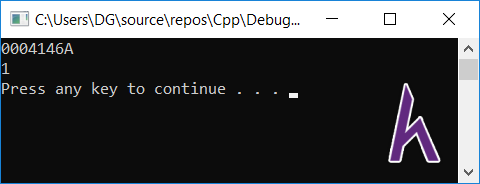
11

12 **return** 0;

13 }

14

**Output:**



Giống như các biến, hàm cũng được lưu trữ tại một địa chỉ trong bộ nhớ. Khi hàm được gọi, chương trình sẽ đi đến địa chỉ của hàm trong bộ nhớ, sau đó thực thi mã lệnh tại vùng nhớ đó.

Vì hàm cũng có địa chỉ trong bộ nhớ, nên ta cũng có thể khai báo một con trỏ cho một hàm.

**Con trỏ hàm là gì?**

**Con trỏ hàm** là một biến **lưu trữ địa chỉ của một hàm**, thông qua biến đó, ta có thể gọi hàm mà nó trỏ tới.

Cú pháp khai báo con trỏ hàm:

**<kiểu trả về>** (**\*<tên con trỏ>**)(**<danh sách tham số>**);

**Ví dụ:**

* [**perl**](https://howkteam.vn/learn)

**1 int**(\*fcnPtr)(**int**); // con trỏ hà**m** nhận vào 1 biến kiểu **int** và trả về kiểu **int**

2 void(\*fcnPtr)(**int**, **int**); // con trỏ hà**m** nhận vào 2 biến kiểu **int** và **tr**ả về kiểu void

3

**Chú ý:** Dấu ngoặc () quanh **\*fcnPtr**là bắt buộc.

**Gán địa chỉ của hàm cho con trỏ hàm**

Giống như mọi con trỏ khác, con trỏ hàm **phải được định nghĩa giá trị**trước khi sử dụng.

* [**perl**](https://howkteam.vn/learn)

1 // khai báo **prototype**

**2 int** funcA();

**3 int** funcB();

4 void funcC();

5 double funcD(**int** a);

6

**7 int** main()

8 {

**9 int**(\*fcnPtr)() = funcA(); // lỗi, không dùng dấu ngoặc đơn () sau tên hà**m**

10 **int**(\*fcnPtrA)() = funcA; // ok, con **tr**ỏ fcnPtrA **tr**ỏ đến hà**m** funcA

11 fcnPtrA = funcB; // ok, fcnPtrA có thể **tr**ỏ đến **m**ột hà**m** khác có cùng cấu **tr**úc

12 // fcnPtrA = &funcB; tương tự câu lệnh **tr**ên

13

14 **int**(\*fcnPtr1)() = funcA; // ok

15 void(\*fcnPtr2)() = funcA; // lỗi, kiểu **tr**ả về của con **tr**ỏ hà**m** và hà**m** không **tr**ùng nhau

16 void(\*fcnPtr3)() = funcC; // ok

17 double(\*fcnPtr4)(**int**) = funcD; // ok

18

19 **return** 0;

20 }

21

Không giống như các kiểu dữ liệu cơ bản, C++ sẽ**ngầm chuyển đổi**một hàm thành một con trỏ hàm nếu cần (vì vậy bạn **không cần sử dụng toán tử (&)**để lấy địa chỉ của hàm).

**Chú ý:** Cấu trúc (tham số và kiểu trả về) của con trỏ hàm phải khớp với cấu trúc của hàm.

**Gọi một hàm bằng con trỏ hàm**

Con trỏ hàm có thể được sử dụng để **gọi hàm mà nó trỏ đến**. Có hai cách để thực hiện lời gọi hàm:

* [**C++**](https://howkteam.vn/learn)

1 #include<iostream>

**2 using** **namespace** **std**;

3

**4 void** swapNumber(**int** &a, **int** &b)

5 {

6 **int** temp = a;

7 a = b;

8 b = temp;

9 }

**10 int** main()

11 {

12 **void**(\*ptrSwap) (**int** &, **int** &) = swapNumber;

13

14 **int** a = 5, b = 10;

15 **cout** << "Before: " << a << " " << b << endl;

16

17 // gọi hàm tường minh

18 (\*ptrSwap)(a, b);

19 **cout** << "After: " << a << " " << b << endl;

20

21 // hoặc gọi hàm ngầm định

22 ptrSwap(a, b);

23 **cout** << "After: " << a << " " << b << endl;

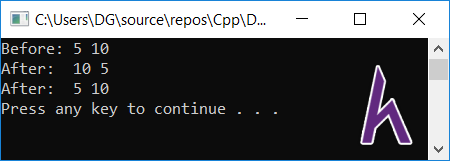
24

25 **return** 0;

26 }

28

**Output:**



**Chú ý:** Các **tham số mặc định** của hàm **không sử dụng được** thông qua con trỏ hàm. Tham số mặc định được compiler xác định tại thời điểm biên dịch (compile) chương trình, còn con trỏ hàm được sử dụng tại thời điểm chương trình đang chạy (run time).

**Truyền hàm vào hàm dưới dạng đối số**

Con trỏ hàm cũng là một biến con trỏ, do đó chúng ta có thể **sử dụng con trỏ hàm là tham số**của một hàm nào đó. Khi tham số của hàm là con trỏ hàm, đối số chính là địa chỉ của hàm.

**Ví dụ:**Viết chương trình thực hiện việc sắp xếp tăng, giảm mảng 1 chiều các số nguyên. Nếu chưa có kiến thức về con trỏ hàm, có thể bạn sẽ thực hiện như bên dưới:

* [**C++**](https://howkteam.vn/learn)

1 #include<iostream>

**2 using** **namespace** **std**;

3

4 // hoán đổi giá trị hai số

**5 void** swapNumber(**int** &a, **int** &b)

6 {

7 **int** temp = a;

8 a = b;

9 b = temp;

10 }

11 // hàm sắp xếp tăng sử dụng thuật toán selection sort

**12 void** selectionSortAsc(**int** \*arr, **int** n)

13 {

14 **int** i, j, min\_idx;

15

16 // One by one move boundary of unsorted subarray

17 **for** (i = 0; i < n - 1; i++)

18 {

19 // Find the minimum element in unsorted array

20 min\_idx = i;

21 **for** (j = i + 1; j < n; j++)

22 {

23 **if** (arr[min\_idx] > arr[j])

24 {

25 min\_idx = j;

26 }

27 }

28

29 // Swap the found minimum element with the first element

30 swapNumber(arr[min\_idx], arr[i]);

31 }

32 }

33

34 // hàm sắp xếp giảm sử dụng thuật toán selection sort

**35 void** selectionSortDesc(**int** \*arr, **int** n)

36 {

37 **int** i, j, max\_idx;

38

39 // One by one move boundary of unsorted subarray

40 **for** (i = 0; i < n - 1; i++)

41 {

42 // Find the maximum element in unsorted array

43 max\_idx = i;

44 **for** (j = i + 1; j < n; j++)

45 {

46 **if** (arr[max\_idx] < arr[j])

47 {

48 max\_idx = j;

49 }

50 }

51

52 // Swap the found maximum element with the first element

53 swapNumber(arr[max\_idx], arr[i]);

54 }

55 }

56 /\* Function to print an array \*/

**57 void** printArray(**int** arr[], **int** size)

58 {

59 **int** i;

60 **for** (i = 0; i < size; i++)

61 **cout** << arr[i] << " ";

62 **cout** << endl;

63 }

64

**65 int** main()

66 {

67 **int** arr[] = { 64, 25, 12, 22, 11 };

68 **int** n = **sizeof**(arr) / **sizeof**(**int**);

69

70 // Sắp xếp tăng

71 selectionSortAsc(arr, n);

72

73 **cout** << "Asc array: \n";

74 printArray(arr, n);

75

76 // Sắp xếp giảm

77 selectionSortDesc(arr, n);

78

79 **cout** << "Desc array: \n";

80 printArray(arr, n);

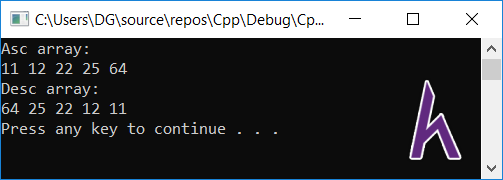
81

82 **return** 0;

83 }

84

**Output:**



Chương trình trên sử dụng thuật toán selection sort để sắp xếp mảng. Bạn có thể thấy, 2 hàm selectionSortAsc() và selectionSortDesc() chỉ khác nhau ở câu lệnh so sánh bên trong vòng lặp thứ 2.

Khi sử dụng con trỏ hàm, bạn có thể tạo ra 1 hàm sắp xếp tổng quát cho 2 hàm trên:

* [**C++**](https://howkteam.vn/learn)

1 #include<iostream>

**2 using** **namespace** **std**;

3

**4 void** swapNumber(**int** &a, **int** &b)

5 {

6 **int** temp = a;

7 a = b;

8 b = temp;

9 }

10

**11 bool** asc(**int** a, **int** b)

12 {

13 **return** a > b;

14 }

15

**16 bool** desc(**int** a, **int** b)

17 {

18 **return** a < b;

19 }

20

**21 void** selectionSort(**int** \*arr, **int** n, **bool**(\*comparisonFcn)(**int**, **int**))

22 {

23 **int** i, j, find\_idx;

24

25 // One by one move boundary of unsorted subarray

26 **for** (i = 0; i < n - 1; i++)

27 {

28 // Find the minimum element in unsorted array

29 find\_idx = i;

30 **for** (j = i + 1; j < n; j++)

31 {

32 **if** (comparisonFcn(arr[find\_idx], arr[j]))

33 {

34 find\_idx = j;

35 }

36 }

37

38 // Swap the found minimum element with the first element

39 swapNumber(arr[find\_idx], arr[i]);

40 }

41 }

42

43 /\* Function to print an array \*/

**44 void** printArray(**int** arr[], **int** size)

45 {

46 **int** i;

47 **for** (i = 0; i < size; i++)

48 **cout** << arr[i] << " ";

49 **cout** << endl;

50 }

51

**52 int** main()

53 {

54 **int** arr[] = { 64, 25, 12, 22, 11 };

55 **int** n = **sizeof**(arr) / **sizeof**(**int**);

56

57 // Sắp xếp tăng

58 selectionSort(arr, n, asc);

59

60 **cout** << "Asc array: \n";

61 printArray(arr, n);

62

63 // Sắp xếp giảm

64 selectionSort(arr, n, desc);

65

66 **cout** << "Desc array: \n";

67 printArray(arr, n);

68

69 system("pause");

70 **return** 0;

71 }

72

Chương trình trên sử dụng con trỏ hàm là tham số thứ 3 của hàm selectionSort(). Khi có thêm những nhu cầu sắp xếp khác nhau, chúng ta chỉ cần viết thêm hàm có điều kiện sắp xếp, và thay đổi đối số thứ 3 khi gọi hàm, mà không phải viết lại toàn bộ thuật toán bên trong hàm.

**Đối số mặc định của tham số hàm kiểu con trỏ hàm**

Tương tự như những kiểu dữ liệu cơ bản khác, chúng ta có thể cung cập một **đối số mặc định**cho tham số hàm kiểu con trỏ hàm.

**Ví dụ:**

* [**C#**](https://howkteam.vn/learn)

1 // mặc định hàm được sắp xếp tăng dần nếu không truyền vào đối số thứ 3

**2 void** selectionSort(**int** \*arr, **int** n, **bool**(\*comparisonFcn)(**int**, **int**) = asc);

**3 int** main()

4 {

5 **int** arr[] = { 64, 25, 12, 22, 11 };

6 **int** n = **sizeof**(arr) / **sizeof**(**int**);

7

8 // Sắp xếp tăng

9 selectionSort(arr, n);

10

11 // Sắp xếp giảm

12 selectionSort(arr, n, desc);

13

14 **return** 0;

15}

16

**std::function trong C++11**

C++11 cung cấp một cách thay thế cho việc sử dụng con trỏ hàm bằng cách sử dụng kiểu dữ liệu**std::function** thuộc thư viện **<functional>** trong **namespace std**.

**Ví dụ:**

* [**C++**](https://howkteam.vn/learn)

1 #include<functional>

2 #include<iostream>

**3 using** **namespace** **std**;

4

5 // khai báo prototype

**6 int** funcA();

**7 double** funcB(**int**);

**8 void** funcC(**int** &a, **int** &b);

9

**10 int** main()

11 {

12 function<**int**()> fncPtrA = funcA;

13 function<**double**(**int**)> fncPtrB = funcB;

14 function<**void**(**int**&, **int**&)> fncPtrC = funcC;

15

16 **return** 0;

17 }

18

Việc sử dụng kiểu dữ liệu **std::function** cũng tương tự như sử dụng con trỏ hàm, chỉ khác nhau về cách khai báo.

**Khai báo con trỏ hàm với từ khóa auto trong C++11**

Từ phiên bản **C++11 trở về sau**, từ khóa **auto**được dùng để **tự động nhận dạng kiểu dữ liệu**thông qua kiểu dữ liệu của**giá trị khởi tạo** ra nó. Vì vậy, từ khóa **auto**cũng có thể nhận dạng ra loại con trỏ hàm.

* [**C++**](https://howkteam.vn/learn)

1 #include<iostream>

**2 using** **namespace** **std**;

3

**4 void** swapNumber(**int** &a, **int** &b)

5 {

6 **int** temp = a;

7 a = b;

8 b = temp;

9 }

10

**11 int** main()

12 {

13 **auto** ptrSwap = swapNumber;

14

15 **int** a = 5, b = 10;

16 **cout** << "Before: " << a << " " << b << endl;

17

18 ptrSwap(a, b);

19 **cout** << "After: " << a << " " << b << endl;

20

21 system("pause");

22 **return** 0;

23 }

24

Từ khóa **auto**giúp cú pháp đơn giản hơn. Tuy nhiên, nhược điểm là tất cả các chi tiết về các tham số và kiểu trả về của hàm đều bị ẩn, do đó dễ mắc lỗi hơn khi sử dụng con trỏ hàm.

**Chú ý:** Từ khóa **auto** xác định kiểu dữ liệu tại**thời gian biên dịch**, nên nó không được sử dụng cho**tham số hàm**. Vì vậy việc sử dụng nó có phần bị hạn chế.

**Kết luận**

Qua bài học này, bạn đã nắm được những kiến thức về Con trỏ hàm trong C++ (Function pointers).

Con trỏ hàm (function pointers) thường được sử dụng khi chúng ta có các hàm có cùng kiểu trả về và danh sách tham số, hoặc khi bạn cần truyền một hàm cho hàm khác.

Con trỏ hàm có cú pháp khai báo khó nhớ và dễ gây ra lỗi nếu chưa nắm rõ, bạn có thể đơn giản hóa bằng cách sử dụng kiểu std::function của C++11.