Question 1

Not answered

Mark 0.00 out of 10.00

[Reference Variable - Swap]

2. Biến tham chiếu (Reference Variable)

Một biến tham chiếu là một **alias**, hay là một tên khác của một biến đã tồn tại. Khi một biến được khai báo là tham chiếu của một biến khác, chúng ta có thể truy cập một biến sử dụng tên thật hoặc tên tham chiếu của nó.

Tham chiếu chủ yếu được dùng để truyền vào các hàm (pass-by-reference). Khi tham chiếu được truyền vào hàm, hàm sẽ làm việc trực tiếp với biến gốc thay vì tạo ra một bản copy (clone) và làm việc trên bản copy đó. Như vậy, mọi thay đổi trên biến sẽ được lưu lại trên biến gốc sau khi kết thúc hàm.

Biến tham chiếu có nhiều điểm tương tự với biến con trỏ. Trong nhiều trường hợp, biến tham chiếu có thể được sử dụng thay thế cho con trỏ, đặc biệt khi được sử dụng làm đối số của các hàm.

2.1 THAM CHIẾU &

Ở bài học trước, chúng ta sử dụng phép & để lấy địa chỉ của một biến trong các biểu thức. Ở phần này, chúng ta sẽ học thêm một cách sử dụng nữa của phép & để khai báo biến tham chiếu của một biến khác.

Phép & có ý nghĩa khác nhau khi sử dụng trong biểu thức và khi khai báo biến. Khi được sử dụng trong một biểu thức, phép & đóng vai trò là phép lấy địa chỉ (address-of operator), trả về địa chỉ của một biến (Ví dụ: nếu `number` là một biến kiểu `int`, `&number` sẽ trả về địa chỉ của vùng nhớ chứa biến `number`).

Tuy nhiên, khi & được sử dụng trong khai báo biến, thì biến được khai báo ngay sau đó được xem là biến tham chiếu. Biến tham chiếu thường được dùng như tên thay thế của một biến khác đã được khai báo trước đó.

Để khai báo một biến tham chiếu, chúng ta sử dụng cú pháp sau:

```
type &newName = existingName;
// hoặc
type& newName = existingName;
// hoặc
type & newName = existingName;
```

Ở các cách khai báo trên, biến `newName` tham chiếu đến biến `existingName`. Chúng ta có thể truy cập đến biến `existingName` sử dụng chính tên gốc của biến đó hoặc qua tham chiếu `newName`.

Ví dụ:

```
/* Khai báo và khởi tạo tham chiếu */
#include
using namespace std;

int main() {
    int number = 88;          // Khai báo biến number
    int & refNumber = number; // Khai báo biến refNumber tham chiếu đến biến number.

    cout << number << endl;    // In ra giá trị của biến number (88)
    cout << refNumber << endl; // In ra giá trị của tham chiếu (88)

    refNumber = 99;            // Gán giá trị mới cho tham chiếu refNumber
    cout << refNumber << endl;
    cout << number << endl;    // Giá trị của biến number cũng thay đổi theo (99)

    number = 55;               // Gán gía trị mới cho biến number
    cout << number << endl;
    cout << refNumber << endl; // Giá trị của tham chiếu cũng thay đổi (55)
}
```

2.3 THAM CHIẾU HOẠT ĐỘNG NHƯ THẾ NÀO?

Tham chiếu hoạt động như một con trỏ. Một tham chiếu được được sử dụng như tên thứ 2 của một biến. Nó chứa địa chỉ của biến, như mô tả dưới đấy:

 2.4 Tham chiếu và con trỏ

Con trỏ và tham chiếu có chức năng khá giống nhau trừ một số đặc điểm sau:

- 1. Một tham chiếu (reference) là một biến hằng số lưu địa chỉ. Tham chiếu phải được khởi tạo ngay sau khi khai báo.

```
int & iRef;   // Lỗi: 'iRef' được khai báo là tham chiếu nhưng chưa được khai báo.
```

  Khi đã khai báo và khởi tạo, biến tham chiếu không thể thay đổi giá trị.
- 2. Để truy vấn giá trị mà con trỏ chỉ đến, chúng ta sử dụng phép *. Tương tự, để gán giá trị của con trỏ với địa chỉ của một biến bình thường, chúng ta sử dụng phép &. Tuy nhiên, đối với biến tham chiếu, hai quy trình này được thực hiện một cách ngầm định, chúng ta không cần phải sử dụng các phép toán như trong con trỏ.

Ví dụ:

```
    /* Tham chiếu vs. Con trỏ */
#include
using namespace std;

int main() {
    int number1 = 88, number2 = 22;

    // Tạo một con trỏ trỏ đến biến number1
    int * pNumber1 = &number1;  // Khởi tạo con trỏ
    *pNumber1 = 99;             // Thay đổi giá trị mà con trỏ chỉ đến
    cout << *pNumber1 << endl;  // 99
    cout << &number1 << endl;   // 0x22ff18
    cout << pNumber1 << endl;   // 0x22ff18 (Giá trị của <a class="autolink" title="Biến con trỏ"
href="https://dev.uet.vnu.edu.vn/mod/quiz/view.php?id=4995">biến con trỏ</a> pValue)
    cout << &pNumber1 << endl;  // 0x22ff10 (Địa chỉ của <a class="autolink" title="Biến con trỏ"
href="https://dev.uet.vnu.edu.vn/mod/quiz/view.php?id=4995">biến con trỏ</a>)
    pNumber1 = &number2;        // Con trỏ có thể đổi giá trị để trỏ đến một địa chỉ khác

    // Tạo một tham chiếu cho biến number1
    int & refNumber1 = number1;  // Tham chiếu ngầm định ( không phải &number1)
    refNumber1 = 11;             // Thay đổi giá trị mà tham chiếu chỉ đến  - không cần sử dụng phép *
    cout << refNumber1 << endl;  // 11
    cout << &number1 << endl;    // 0x22ff18
    cout << &refNumber1 << endl; // 0x22ff18
    //refNumber1 = &number2;     // Lỗi: Tham chiếu không thể thay đổi giá trị, nó là một hằng số.
    refNumber1 = number2;        // refNumber1 vẫn là tham chiếu của biến number1
                                 // Lệnh trên chỉ copy giá trị của biến number2 vào tham chiếu refNumber1 (hay number1)
    number2++;
    cout << refNumber1 << endl;  // 22
    cout << number1 << endl;     // 22
    cout << number2 << endl;     // 23
}
```

2.4 TRUYỀN THAM CHIẾU VÀO HÀM SỬ DỤNG BIẾN THAM CHIẾU HOẶC BIẾN CON TRỎ

Truyền bằng giá trị

Trong ngôn ngữ lập trình C/C++, các đối số được truyền vào hàm bằng gía trị (ngoại từ mảng `array`, bởi vì mảng thực chất là con trỏ). Có nghĩa là, các bản sao (clone) của các đối số sẽ được tạo ra và truyền vào hàm. Những thay đổi tác động lên bản sao của các đối số trong hàm sẽ không làm thay đổi bản gốc của đối số - được khai báo bên ngoài hàm.

Ví dụ:

```
    /* Truyền bằng giá trị */
#include
using namespace std;

int square(int);

int main() {
    int number = 8;
    cout <<  "In main(): " << &number << endl;  // 0x22ff1c
    cout << number << endl;         // 8
    cout << square(number) << endl; // 64
    cout << number << endl;         // 8 - bản gốc của đối số : gía trị không thay đổi
}

int square(int number) {  // non-const
    cout <<  "In square(): " << &number << endl;  // 0x22ff00
    number *= number;           // Chỉnh sửa trên bản sao
    return number;
}
```

Ở đoạn code trên, biến `number` ở hàm `main()` và biến `number` ở hàm `square()` là hoàn toàn khác nhau. Việc thay đổi biến `number` ở hàm `square()` không làm thay đổi giá trị của biến `number` ở hàm `main()`.

Truyền tham chiếu sử dụng đối số con trỏ.

Trong nhiều trường hợp, chúng ta muốn chỉnh sửa trực tiếp lên bản gốc của một biến chứ không phải bản copy. Để làm được điều này, chúng ta có thể truyền biến con trỏ làm đối số của hàm.

Ví dụ:

```
    /* Truyền tham chiếu sử dụng biến con trỏ */
#include
using namespace std;

void square(int *);

int main() {
    int number = 8;
    cout <<  "In main(): " << &number << endl;  // 0x22ff1c
    cout << number << endl;   // 8
    square(&number);          // Truyền địa chỉ của biến thay vì truyền tên biến
    cout << number << endl;   // 64
}

void square(int * pNumber) {  // Hàm nhận đối số con trỏ
    cout <<  "In square(): " << pNumber << endl;  // 0x22ff1c
    *pNumber *= *pNumber;      // Thay đổi vào ô nhớ mà biến pNumber chỉ đến.
}
```

Trong ví dụ trên, biến `pNumber` ở hàm `square()` và biến `number` ở hàm `main()` cùng trỏ đến một vùng nhớ trên máy tính, nên thay đổi ở hàm `square()` sẽ tác động lên hàm `main()` và được lưu lại sau khi hàm kết thúc.

Truyền tham chiếu với đối số tham chiếu

Thay vì truyền con trỏ vào hàm, biến tham chiếu có thể được sử dụng thay thế để tránh những rắc rối mà con trỏ đem lại.

Ví dụ:

```
/* Truyền tham chiếu sử dụng đối số tham chiếu */
#include
using namespace std;

void square(int &);

int main() {
    int number = 8;
    cout <<  "In main(): " << &number << endl;  // 0x22ff1c
    cout << number << endl;  // 8
    square(number);          // Truyền tham chiếu
    cout << number << endl;  // 64
}

void square(int & rNumber) {  // Hàm nhận đối số là tham chiếu kiểu int
    cout <<  "In square(): " << &rNumber << endl;  // 0x22ff1c
    rNumber *= rNumber;
}
```

Bài tập

Phép hoán đổi giá trị của hai biến được sử dụng trong nhiều bài toán, chẳng hạn như bài toán sắp xếp. Thông thường, người lập trình sẽ xây dựng một hàm riêng thực hiện phép toán này nhằm mục đích tối ưu việc viết code.

Là một lập trình viên, bạn hãy viết hàm `void swap(int& a, int& b)` thực hiện việc hoán đổi giá trị của hai biến , trong đó hàm nhận đối số đầu vào là hai biến tham số `a` và `b`.

For example:

Input	Result
1 2	2 1

Answer:

```
1
```

	Input	Expected	Got	
✓	1 2	2 1	2 1	✓
✓	3 4	4 3	4 3	✓
✓	123456789 987654321	987654321 123456789	987654321 123456789	✓
✓	100 1000	1000 100	1000 100	✓
✓	444 555	555 444	555 444	✓

Passed all tests! ✓

Back to Course