Status	Finished
Started	Friday, 6 December 2024, 12:20 AM
Completed	Friday, 6 December 2024, 12:25 AM
Duration	4 mins 24 secs
Grade	**10.00** out of 10.00 (**100**%)

Question 1

Correct

Mark 10.00 out of 10.00

Con trỏ - Pointers

Con trỏ (Pointers), tham chiếu (References) và cấp phát bộ nhớ động (Dynamic Memory Allocation) là những tính năng cực kì mạnh mẽ trong ngôn ngữ lập trình C/C++. Những tính năng này cho chép các lập trình viên quản lý và tối ưu tài nguyên của máy tính một cách hiểu quả nhất. Tuy nhiên, đây cũng là nội dung khá phức tạp và gây khó khăn cho người mới bắt đầu.

Con trỏ (Pointers) cho phép chúng ta truy cập và kiểm soát nội dung của các vùng nhớ trong bộ nhớ máy. Nếu được sử dụng đúng cách, con trỏ có thể giúp tăng hiệu năng của chương trình rất nhiều. Tuy nhiên, sử dụng con trỏ không hợp lý có thể dẫn tới rất nhiều vấn đề khác nhau, từ việc làm cho mã nguồn (code) trở nên khó đọc và khó bảo trì cho đến gây ra các lỗi nguy hiểm như thất thoát bộ nhớ (memory leaks) hay tràn bộ nhớ đệm (buffer overflow). Các hacker có thể dựa vào những lỗi này để gây hại cho chương trình của bạn. Nhiều ngôn ngữ lập trình gần đây (như Java hay C#) đã loại bỏ các cú pháp sử dụng con trỏ để tránh những rắc rối mà nó có thể gây ra.

1. Biến con trỏ

Trong máy tính của chúng ta, mỗi ô nhớ (computer memory location) có một địa chỉ (address) và dữ liệu (content) tại ô nhớ đó. Địa chỉ (address) của ô nhớ là một số thường được biểu diễn ở hệ 16 (hexa), ví dụ `0x7ffc7570aa94`. Giá trị của ô nhớ thường được lưu ở dạng nhị phân (binary), người lập trình có thể dịch nó thành các kiểu dữ liệu khác nhau như số nguyên `int`, số thực `double`, kí tự `char` hay xâu `string`.

Tuy nhiên, thông thường các lập trình viên thường gặp khó khăn trong việc sử dụng con trỏ một cách trực tiếp, vì thế các ngôn ngữ lập trình đã đưa ra khai niệm biến (variable). Thông qua tên của biến, lập trình viên có thể truy cập đến dữ liệu của vùng nhớ có địa chỉ mà hệ điều hành đã cấp phát cho biến đó. Cùng với tên biến, lập trình viên cũng phải xác định kiểu của biến (ví dụ: `int`,`double`,`char`) để giúp chúng ta có thể dễ dàng dịch nội dung của biến từ dạng nhị phân (kiểu dữ liệu dùng để lưu trong bộ nhớ máy tính) sang dạng dữ liệu phù hợp mà con người có thể đọc và hiểu được.

Mỗi ô nhớ trong bộ nhớ máy tính thường có kích cỡ 8 bit (1 byte). Để lưu một biến kiểu nguyên `int` 4-bytes trong bộ nhớ, chúng ta cần sử dụng 4 ô nhớ.

Hình vẽ dưới đây biểu diễn mỗi quan hệ giữa địa chỉ (address) và nội dung (content) của bộ nhớ máy tính (computer memory) với tên (name), kiểu (type) và giá trị (value) của biến được sử dụng khi lập trình.

1.1 BIẾN CON TRỎ (POINTER VARIABLE)

Một biến con trỏ (pointer), giống như một biến thông thường khác như `int` hay `double`, được dùng để lưu một giá trị nào đó. Nhưng thay vì được dùng để lưu các giá trị số nguyên (`int`), số thực (`double`) ..., thì biến con trỏ lưu địa chỉ (address) của một vùng nhớ trong bộ nhớ máy tính.

1.2 KHAI BÁO BIẾN CON TRỎ

Tương các biến thông thường khác, biến con trỏ cần phải được khai báo trước khi sử dụng. Để khai báo một biến con trỏ, chung ta cần phải thêm đấu `*` vào trước tên biến. Biến con trỏ cũng cần được khai báo kiểu, ví dụ `int`, `double`, `boolean`.

```
// Cách khai báo con trỏ
type *ptr; // Cách 1
// hoặc
type* ptr; // Cách 2
// hoặc
type *ptr; // Cách 3
```

Ví dụ, chúng ta có thể khai báo các biến con trỏ như sau:

```
       int *iPtr; // Khai báo một biến con trỏ kiểu int tên là iPtr trỏ đến một vùng nhớ chứa số nguyên.
double * dPtr; // Khai báo một biến con trỏ kiểu double
```

Chú ý rằng, chúng ta cần phải đặt một dấu * trước **từng** tên biến con trỏ mà chúng ta muốn khai báo. Điều này có nghĩa là dấu * chỉ có tác dụng với tên biến đi theo ngay sau đó. Khi viết theo cú pháp khai báo con trò, dấu * không phải là một phép toán nhân mà chúng ta hay dùng.

```
    int *p1, *p2, i; // p1 và p2 là biến con trỏ, i là một biến thông thường kiểu int.
int* p1, p2, i; // p1 là biến con trỏ, p2 và i là biến thông thường kiểu int.
int * p1, * p2, i; // p1 và p2 là biến con trỏ, i là biến thông thường kiểu int.
```

Quy ước đặt tên biến: Chúng ta nên đặt tên cho biến con trỏ bắt đầu bằng chữ `p` hoặc kết thúc bằng chữ `Ptr` để giúp mã nguồn của chúng ta dễ đọc và bảo trì hơn. Ví dụ: `iPtr`, `numberPtr`, `pNumber`, `pStudent`.

1.3 KHỞI TẠO BIẾN CON TRỎ SỬ DỤNG PHÉP & (ADDRESS-OF-OPERATOR)

Ban đầu, khi khai báo một biến con trỏ, nội dung của biến con trỏ sẽ được gán bằng một địa chỉ ngẫu nhiên nào đó mà chúng ta không quan tâm tới (địa chỉ không hợp lệ). Điều này **cực kì nguy hiểm** vì nó có thể gây hại đến các các chương trình khác đang chạy trong máy tính. Vì vậy, chúng ta cần phải khởi tạo biến con trỏ bằng cách gán nó với một địa chỉ mà chúng ta muốn làm việc với nó. Chúng ta có thể gán con trỏ với địa chỉ của một biến thông thường thông qua phép &.

Phép & (address-of-operator) trả về địa chỉ của biến thông thường. Ví dụ, nếu biến `number` là một biến thông thường kiểu `int`, lệnh `&number` sẽ trả về địa chỉ trong bộ nhớ máy tính mà biến `number` được lưu.

Ví dụ:

```
int number = 88; // Khai báo biến thông thường và gán gía trị
int *pNumber; // Khai bảo biến con trỏ
pNumber = &number; // Khởi tạo biến con trỏ, gán giá trị con trỏ bằng địa chỉ của biến number

int *numberPtr = &number; // Chúng ta cũng có thể khởi tạo con trỏ ngay sau lệnh khai báo
```

Như đã mô tả ở hình trên, biến `number` chứa số nguyên `int` có giá trị bằng 88, và được lưu trong 4 ô nhớ bắt đầu từ ô có địa chỉ `0x22ccec`. Câu lệnh `&number` trả về địa chỉ của biến `number` (là địa chỉ của ô nhớ đầu tiên, `0x22ccec`). Địa chỉ này sau đó được gán vào (khởi tạo) biến `pNumber`.

1.4 PHÉP LẤY GIÁ TRỊ CON TRỎ *

Khi sử dụng dấu * trước một biến con trỏ, chúng ta sẽ lấy được giá trị của ô nhớ mà con trỏ đó chỉ tới. Ví dụ, nếu `pNumber` là một biến con trỏ kiểu `int`, lệnh `*pNumber` sẽ trả về giá trị của ô nhớ mà `pNumber` chỉ tới.

Ví dụ:

```
    int number = 98; // Khai báo biến kiểu int
int* pNumber = &number; // Khai báo biến con trỏ kiểu int và khởi tạo nó bằng địa chỉ của biến number
cout << pNumber << endl; // In ra giá trị của con trỏ pNumber, là một địa chỉ
cout << *pNumber << endl; // In ra giá trị của ô nhớ mà biến pNumber chỉ đến
*pNumber = 99; // Sửa giá trị của vùng nhớ mà biến pNumber chỉ tới.
cout << *pNumber << endl; // In ra gía trị của vùng nhớ vừa bị thay đổi
cout << number << endl; // Giá trị của biến number cũng thay đổi theo.
```

Lưu ý: Phép * có ý nghĩa khác nhau khi sử dụng trong câu lệnh khai báo và trong biểu thức. Khi dấu * được sử dụng trong khai báo biến (ví dụ: `int * pNumber;`), nó chỉ ra rằng biến được khai báo ngay phía sau đấu * là một biến con trỏ. Khi sử dụng trong một biểu thức (ví dụ: `*pNumber = 99; std::cout << *pNumber;`), nó trả về giá trị của vùng nhớ mà con trỏ chỉ tới.

1.5 KIỂU CỦA BIẾN CON TRỎ

Trong quá trình khai báo biến con trỏ, lập trình viên phải xác định kiểu dữ liệu của con trỏ đó. Khi đã khai báo, một con trỏ chỉ có thể chứa địa chỉ của vùng nhớ lưu kiểu dữ liệu đã khai báo cho con trỏ đó và không thể chứa kiểu dữ liệu khác.

Ví dụ **đúng**:

```
    int i = 88;
double d = 55.66;
int * iPtr = &i;    // Con trỏ kiểu int trỏ đến vùng nhớ chứa dữ liệu kiểu int
double * dPtr = &d; // Con trỏ kiểu double trỏ đến vùng nhớ chứa dữ liệu kiểu double
```

Ví dụ **sai**:

```
        int i = 88;
double d = 55.66;
int * iPtr = &d;    // LỖI: Con trỏ kiểu int không được chứa địa chỉ vùng nhớ chứa dữ liệu kiểu double
double * dPtr = &i; // LỖI: Con trỏ kiểu double không được chứa địa chỉ vùng nhớ chứa dữ liệu kiểu int
iPtr = i; // LỖI: Con trỏ chỉ chứa địa chỉ, không chứa giá trị.
```

1.6 LỖI CON TRỎ CHƯA KHỞI TẠO

Đoạn code sau đây có một lỗi nguy hiểm mà các lập trình viên hay mắc phải:

```
    int * iPtr;
*iPtr = 55;
cout << *iPtr << endl;
```

Biến con trỏ `iPtr` đã được khai báo nhưng chưa khởi tạo. Vì vậy, hệ điều hành sẽ gán cho nó ngẫu nhiên một giá trị địa chỉ trỏ đến một vùng nhớ ngẫu nhiên nào đó trong máy tính, vùng nhớ này có thể nằm ngoài quyền kiểm soát của chương trình đang chạy. Khi thực thi lệnh `*iPtr = 55;`, chương trình vô hình dung thay đổi giá trị ở một ô nhớ nào đó một cách không hợp pháp. Điều này có thể gây ra lỗi và làm cho chương trình ngừng hoạt động. Hơn nữa, hầu hết các trình biên dịch (compiler) không thông báo lỗi (error) hay cảnh báo (warning) cho lập trình viên về lỗi này trong quá trình biên dịch, nên chúng ta phải tự kiểm tra lại các lỗi kiểu như trên trước khi biên dịch chương trình.

1.7 CON TRỎ RỖNG - NULL

Chúng ta có thể khởi tạo một biến con trỏ bằng giá trị `0` hoặc `NULL`, khi đó con trỏ sẽ không trỏ đến bất cứ một vùng nhớ nào cả. Truy cập vào con trỏ có giá trị `NULL` sẽ gây ra lỗi `STATUS_ACCESS_VIOLATION`.

```
    int * iPtr = 0; // Khai báo biến con trỏ iPtr và gán nó với giá trị NULL
cout << *iPtr << endl; // LỖI: STATUS_ACCESS_VIOLATION
int *p = NULL; // Cách khác để khai con trỏ Null.
```

Bài tập

Hàm `double* getPointerToPi()` có nhiệm vụ như sau:

- Khai báo một biến kiểu `double` đặt tên là `pi` ở ngoài hàm (biến toàn cục) và gán giá trị cho của số $\pi = 3.14159$ cho biến đó.
- Khai báo một biến con trỏ kiểu `double*` và khởi tạo con trỏ bằng địa chỉ của biến `pi` đã khai báo ở trên.
- Trả về con trỏ đã khai báo.

Hãy viết mã mã C++ để hoàn thành hàm `double* getPointerToPi()` có yêu cầu như trên.

Answer:

```
1 double pi = 3.14159;
2 double* getPointerToPi()
3 {
```

```
4       double *ptr = &pi;
5       return (ptr);
6   }
7
```

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

Back to Course