# **Bài 8. Con trỏ (tiếp)**

**Mục tiêu:**

1. *Luyện tập sử dụng con trỏ và địa chỉ của các biến*
2. *Sử dụng con trỏ khi thao tác với mảng.*

***Giới hạn:*** *không dùng C++String và các thư viện stl (chẳng hạn vector, algorithm)*

**Yêu cầu nộp bài** *: Toàn bộ phần A1-3 và C1-2.*

*Bài này đánh giá theo cố gắng làm việc. Bạn cần làm đủ bài và đúng yêu cầu đề bài.  
Nếu bạn làm A4 với đầy đủ chi tiết, bạn sẽ có thể bù điểm cho một bài tập trước. Viết vào* ***readme.txt*** *nếu có nhu cầu này (chú ý đặt tên file chính xác).*

***Lưu ý: chép bài hoặc cho chép bài sẽ dẫn đến trượt môn học!***

## Thực hành

Nếu bạn chưa thạo con trỏ, bạn có thể dùng phương pháp dưới đây để tự lần bước một đoạn code ngắn mà không cần máy tính, và qua đó sẽ nắm vững các thao tác con trỏ và bộ nhớ.

Vẽ một bảng biểu diễn vùng bộ nhớ được sử dụng trong đoạn chương trình đó, chẳng hạn:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

Giả sử mỗi ô có kích thước 32 bit, nghĩa là 1 biến int hoặc 1 con trỏ, hãy tự bịa ra địa chỉ cho các ô nhớ trên và dùng địa chỉ đánh nhãn cho chúng. Chẳng hạn:

|  |  |
| --- | --- |
| 0x1000 |  |
| 0x1004 |  |
| 0x1008 |  |
| 0x100c |  |
| ... |  |

Với mỗi biến trong đoạn code, hãy đặt nó vào một ô nhớ. Chẳng hạn, char\* s tại địa chỉ 0x1004 và int i tại địa chỉ 0x1008:

|  |  |
| --- | --- |
| 0x1000 |  |
| 0x1004 | Giá trị của s |
| 0x1008 | Giá trị của i |
| 0x100c |  |
| ... |  |

Giả sử ta có lệnh gán

s = “Hi Amy!”;

nghĩa là s trỏ tới một chuỗi kí tự có nội dung “Hi Ann!” ở đâu đó trong bộ nhớ, và giả sử trình biên dịch đặt chuỗi đó tại vùng bộ nhớ bắt đầu từ địa chỉ 0x100c, nghĩa là vùng bộ nhớ 0x1004 của s có giá trị 0x100c.

Giả sử ta có lệnh gán

i = 20;

có nghĩa là vùng bộ nhớ 0x1008 của i giữ giá trị 20. Ta có sơ đồ bộ nhớ:

|  |  |
| --- | --- |
| 0x1000 |  |
| 0x1004 | Giá trị của s: 0x100c |
| 0x1008 | Giá trị của i: 20 |
| 0x100c | Hi A |
| 0x1010 | my!\0 |

Có thể đọc từ sơ đồ bộ nhớ trên các thông tin sau:

|  |  |  |
| --- | --- | --- |
| **Biểu thức** | Mô tả | Giá trị trong ví dụ |
| **i** | Giá trị của biến int i | 20 |
| **&i** | Địa chỉ của biến i | 0x1008 |
| **s** | Giá trị của biến con trỏ s | 0x100c |
| **&s** | Địa chỉ của s | 0x1004 |
| **\*s** | Kí tự mà s trỏ tới, nghĩa là giá trị kiểu char của vùng bộ nhớ có địa chỉ là giá trị của s | ‘H’ |
| **s[3]** | Kí tự mà (s+3) trỏ tới, nghĩa là kí tự thứ 3 đứng sau vị trí s trỏ tới | ‘A’ |
| **s+4** | Địa chỉ thu được khi cộng 4 vào s | 0x1010 |
| **\*(s+4)** | Kí tự tại địa chỉ mà (s+4) trỏ tới - địa chỉ 0x1010 | ‘m’ |

Giả sử ta có lệnh gán

char\*\* p = &s;

có nghĩa là p là con trỏ tới con trỏ kiểu char và p được trỏ tới địa chỉ của biến s. Giả sử địa chỉ của p tại 0x1000, Ta có sơ đồ bộ nhớ:

|  |  |
| --- | --- |
| 0x1000 | Giá trị của p: 0x1004 |
| 0x1004 | Giá trị của s: 0x100c |
| 0x1008 | Giá trị của i: 20 |
| 0x100c | Hi A |
| 0x1010 | my!\0 |

Và ta có thể đọc tiếp từ sơ đồ bộ nhớ các thông tin sau:

|  |  |  |
| --- | --- | --- |
| **Biểu thức** | Mô tả | Giá trị trong ví dụ |
| **p** | Giá trị của biến p | 0x1004 |
| **&p** | Địa chỉ của biến p | 0x1000 |
| **\*p** | Giá trị của ô nhớ mà p trỏ tới (ô nhớ tại địa chỉ 0x1004). Ở đây do p là con trỏ tới char\*, nên ô nhớ đó có kiểu con trỏ - địa chỉ bộ nhớ. | 0x100c |
| **\*\*p** | Kí tự tại địa chỉ (\*p) | ‘H’ |

1. ***Hãy điền thêm vào chương trình dưới đây.*** Để in ra địa chỉ của biến x trong hàm f() và của biến y trong hàm g(). Bạn có nhận xét gì về địa chỉ của hai biến đó, thử giải thích vì sao.

Gợi ý (bôi đen để nhìn thấy): : vị trí của stack frame cho hàm f khi nó chạy trùng với vị trí của hàm g khi nó được gọi (đều bắt đầu trên đỉnh stack của hàm main), cả hai hàm f và g cùng có 1 tham số kiểu int, cùng kiểu trả về, x và y cùng là biến địa phương đầu tiên của hàm, do đó khả năng lớn là trình biên dịch xếp x và y ở cùng một địa chỉ.

void f(int xval)

{

int x;

x = xval;

cout << &x; // in địa chỉ của x tại đây

}

void g(int yval)

{

int y;

cout << &y;// in địa chỉ của y tại đây

}

int main()

{

f(7);

g(11);

return 0;

}

1. ***Tính kích thước biến.*** Chương trình dưới đây sử dụng các phép toán đối với con trỏ để tính kích thước của một biến kiểu char. Con trỏ cp trỏ tới biến thuộc một dữ liệu kiểu, khi cộng thêm 1 vào cp, ta có một con trỏ trỏ tới phần tử tiếp theo thuộc kiểu dữ liệu đó. Do đó, khi biết giá trị của hai con trỏ cp và (cp+1), ta có thể tính ra kích thước của kiểu dữ liệu đó.  
   Trong chương trình dưới đây, ta dùng cp trỏ tới kiểu char. Vì kích thước dữ liệu kiểu char là 1, cho nên chênh lệch giá trị của hai con trỏ chỉ là 1.
   1. Hãy thử dịch và chạy để quan sát hoạt động của chương trình.
   2. Sửa chương trình để xem output có thể tính được kích thước của kiểu int.**//4**
   3. Sửa chương trình để xem output có thể tính được kích thước của kiểu double.**//8**
   4. Chuyện gì xảy ra nếu thay cp++ bằng cp+=2 ? Hãy tính trước kết quả và thử lại bằng chương trình. (làm cho kết quả câu c)**//16**

int main( )

{

char a[4] = "abc";

for (char \*cp = a; (\*cp) != '\0'; cp++) {

cout << (void\*) cp << " : " << (\*cp) << endl;

}

return 0;

}

int main( )

{

double a[4] = {0,1,2,3};

for (double \*cp = a; cp - a < 4; cp++) {

cout << (void\*) cp << " : " << (\*cp) << endl;

}

return 0;

}

1. ***SwapPointer.*** Bạn đã làm thành công hàm swap với tham số kiểu con trỏ tới int.Nhưng hàm swap dưới đây không chạy như mong muốn. Đáng ra output phải là

I should print first

I should print second

Hãy giải thích lý do và sửa code để chạy đúng.   
Gợi ý (bôi đen để xem): Muốn swap hai biến int, bạn cần dùng tham số là con trỏ tới int, muốn swap hai biến con trỏ, bạn phải dùng tham số là con trỏ tới con trỏ.

void swap\_pointers(char\***&** x, char\***&** y)

{

char \*tmp;

tmp = x;

x = y;

y = tmp;

}

int main()

{

char a[] = "I should print second";

char b[] = "I should print first";

char \*s1 = a;

char \*s2 = b;

swap\_pointers(s1,s2);

cout << "s1 is " << s1 << endl;

cout << "s2 is " << s2 << endl;

return 0;

}

1. ***Sửa lỗi.*** Hãy tìm và sửa hết các lỗi sử dụng con trỏ trong chương trình sau:

int main() {

char \*\*s;

char foo[] = "Hello World";

\*s = foo;

printf("s is %s\n",s);

s[0] = foo;

printf("s[0] is %s\n",s[0]);

return(0);

}

## Câu hỏi

## Bài tập

1. Tất cả các câu trong bài này đều phải dùng con trỏ để xử lý mảng/xâu, KHÔNG dùng thư viện C++string, hay cstring, hãy tự viết lấy các hàm bạn cần sử dụng, chẳng hạn strlen, strcpy...

Viết các hàm sau đây, demo việc sử dụng chúng trong hàm main() của một chương trình.

* 1. ***Đảo xâu.*** Viết một hàm reverse(char a[]) với nhiệm vụ đảo ngược thứ tự của a).
  2. ***Xóa kí tự.*** Viết hàm delete\_char(char a[], char c) với nhiệm vụ xóa hết các kí tự c trong xâu a.
  3. ***Độn phải.*** Viết hàm pad\_right(char a[], int n) với nhiệm vụ độn thêm các kí tự space vào cuối xâu a để a có độ dài bằng n. Nếu a đã dài quá n thì không phải làm gì cả.
  4. ***Độn trái.*** Viết hàm pad\_left(char a[], int n) với nhiệm vụ độn thêm các kí tự space vào đầu xâu a để a có độ dài bằng n. Nếu a đã dài quá n thì không phải làm gì cả.
  5. ***Cắt xâu.*** Viết hàm truncate(char a[], int n) với nhiệm vụ cắt bớt đoạn cuối của xâu a để a có độ dài n nếu như a đang dài quá n kí tự.
  6. ***Đối gương.*** Viết hàm bool is\_palindrome(char a[]) với nhiệm vụ kiểm tra xem xâu a có đối xứng hay không. Trả về true nếu đối xứng, false nếu không.
  7. ***Lọc trái.*** Viết hàm trim\_left(char a[]) với nhiệm vụ xóa các kí tự trắng đứng ở đầu xâu a, nếu có.
  8. ***Lọc phải.*** Viết hàm trim\_right(char a[]) với nhiệm vụ xóa các kí tự trắng đứng ở cuối xâu a, nếu có.

#include <iostream>

using namespace std;

void reverse(char a[])

{

int len = 0;

while (a[len] != '\0')

len++;

for (int i = 0; i <= len / 2 - 1; i++)

swap(a[i], a[len - i - 1]);

}

void delete\_char(char a[], char c)

{

int stepBack = 0, i;

for (i = 0; a[i] != '\0'; i++){

a[i - stepBack] = a[i];

if (a[i] == c)

stepBack++;

}

a[i - stepBack] = '\0';

}

void pad\_right(char a[], int n)

{

int len;

for (len = 0; a[len] != '\0'; len++);

if (len < n){

while (len < n)

a[len] = '-', len++;

a[n] = '\0';

}

}

void pad\_left(char a[], int n)

{

int len;

for (len = 0; a[len] != '\0'; len++);

if (len < n){

int more = n - len;

for (int i = len - 1; i >= 0; i--)

a[i + more] = a[i];

for (int i = 0; i < more; i++)

a[i] = '-';

a[n] = '\0';

}

}

void truncate(char a[], int n)

{

int len;

for (len = 0; a[len] != '\0'; len++);

if (len > n)

a[n] = '\0';

}

bool is\_palindrome(char a[])

{

int len;

for (len = 0; a[len] != '\0'; len++);

for (int i = 0, j = len - 1; i < j; i++, j--)

if (a[i] != a[j])

return 0;

return 1;

}

void trim\_left(char a[])

{

int len, Move = 0;

for (len = 0; a[len] != '\0'; len++);

for (int i = 0; i < len && a[i] == ' '; i++)

Move++;

for (int i = 0; i < len - Move; i++)

a[i] = a[i + Move];

a[len - Move] = '\0';

}

void trim\_right(char a[])

{

int len, Move = 0;

for (len = 0; a[len] != '\0'; len++);

for (int i = len - 1; i >= 0 && a[i] == ' '; i--)

Move++;

a[len - Move] = '\0';

}

int main()

{

char a[] = "cabcdcec";

pad\_right(a, 19);//why error

cout << &a;

}

//tsao truyen char[] dia chi thay doi??

1. ***Banner.***  Làm phiên bản mới hơn của bài banner (ở bài 05.Function). Lần này tận dụng con trỏ và các hàm xử lý xâu kí tự trong thư viện cstring. *(nếu bạn chưa từng làm Banner ở các bài tập cũ, bạn được phép xin phiên bản code từ các bài trước của một bạn khác trong lớp để làm bài này. Nếu dùng code của người khác, cần ghi rõ vào file Readme là code gốc của ai.*)

(còn nữa)(thầy Khôi sẽ bổ sung vào đây)