# **Bài 9. Con trỏ và bộ nhớ cấp phát động**

**Mục tiêu:**

1. *Luyện tập sử dụng các biến và mảng được cấp phát động*
2. *Thử nghiệm các loại lỗi do sử dụng con trỏ và bộ nhớ động.*

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

**Yêu cầu nộp bài** *: Toàn bộ phần A và C1*

*Bài này đánh giá theo cố gắng làm việc. Bạn cần làm đủ. Chương trình của bạn có thể chưa chạy đúng, thậm chí chưa chạy, nhưng vẫn có thể được tính là có làm và đạt.   
Bạn có thể làm thêm C2 để bù điểm cho các bài trước. Viết vào* ***readme.txt*** *nếu có nhu cầu này.*

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

## Thực hành

1. ***Trả về mảng động.*** Viết một hàm **concat(const char\*, const\* char)** nhận tham số là hai hằng xâu kí tự, trả về một xâu kí tự là kết quả của việc nối hai xâu đó. Chẳng hạn nếu tham số lần lượt có nội dung là “Hello“ và “World” thì xâu trả về có nội dung “HelloWorld”.  
   Thảo luận: C++ không cho phép hàm có giá trị trả về là một mảng kiểu thông thường. Đối với các hàm xử lý ra kết quả là một mảng, thay vì trả về mảng, hàm chỉ có thể trả về địa chỉ của mảng kết quả. Vấn đề là mảng kết quả đó đặt ở đâu? Ta có một vài lựa chọn:
   1. *mảng kết quả được khai báo sẵn là biến toàn cục, các hàm tự do truy cập.*
   2. *mảng kết quả được nơi gọi hàm khai báo sẵn và truyền địa chỉ vào hàm concat*.
   3. *Mảng kết quả được khai báo là biến địa phương của concat và concat sẽ trả về địa chỉ của mảng đó cho nơi gọi hàm*
   4. *mảng kết quả được cấp phát động từ bên trong concat(), và concat() sẽ trả về địa chỉ của mảng đó cho nơi gọi hàm*

Trong nhiều trường hợp, đặc biệt ở bài này, (a) không phải lựa chọn tốt vì ta thường không biết mảng này cần có kích thước bao nhiêu để khai báo sẵn kích thước cố định cho hàm toàn cục. Hơn nữa, dữ liệu để ở khu vực biến toàn cục do được các hàm tự do truy cập nên dễ bị lỗi. (b) cũng không phải lựa chọn tốt nếu vì nơi gọi hàm phải biết kết quả sẽ có kích thước bao nhiêu phần tử để mà chuẩn bị trước cho đủ; concat phải phụ thuộc vào nơi gọi, nó sẽ bị lỗi nếu nơi gọi hàm không chuẩn bị đúng kích thước cần thiết. (c) là lựa chọn tồi vì mảng kết quả địa phương sẽ bị gỡ khỏi bộ nhớ ngay khi concat kết thúc, việc nơi gọi hàm truy nhập để lấy kết quả sẽ là truy nhập vào vùng bộ nhớ không hợp lệ (dangling references). (d) là lựa chọn hợp lý nhất, concat chủ động xin cấp phát lượng bộ nhớ đủ dùng, nó không bị phụ thuộc vào nơi gọi hàm, và vùng bộ nhớ concat cấp phát vẫn tiếp tục hợp lệ sau khi nó kết thúc.  
Do đó, bạn cần dùng lựa chọn (d) cho bài này.

*Gợi ý bổ sung (bôi đen để xem)*:

* trong hàm concat, bạn có thể dùng hàm trong thư viện <cstring>, chẳng hạn strcpy() hay strncpy() để sao chép xâu, strcat hay strncat để nối xâu. Xem chi tiết và ví dụ tại http://www.cplusplus.com/reference/cstring/.
* Ý tưởng thuật toán: (1) tính ra kích thước của xâu kết quả, (2) cấp phát động mảng kết quả với kích thước vừa tính, (3) chép xâu thứ nhất vào đầu mảng động, (4) chép/nối xâu thứ hai vào cuối mảng. (5) trả về địa chỉ mảng kết quả.

Hãy demo hàm concat trong hàm main(), đừng mất công nhập dữ liệu từ bàn phím, hãy hardcode dữ liệu thử nghiệm.

**#include <iostream>**

**#include <cstring>**

**using namespace std;**

**char\* concat(const char\* a, const char\* b)**

**{**

**int lena = 0;**

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

**lena++;**

**char\* ans = new char;**

**strcpy(ans, a);**

**strcpy(ans + lena, b);**

**return ans;**

**}**

**int main() {**

**char a[] = "Hello", b[] = "World";**

**cout << concat(a, b);**

**}**

1. ***Lỗi sử dụng con trỏ.*** Hãy chạy chương trình thử nghiệm các đoạn code sau. Chú thích vào code đâu là các dòng gây lỗi và đó là lỗi gì.

int\* p = new int;

int\* p2 = p;

\*p = 10;

delete p; **//lỗi p2 ko trỏ đến đâu cả**

\*p2 = 100;

cout << \*p2;

delete p2;

1. Hãy chạy chương trình thử nghiệm đoạn lệnh sau.Chạy thử và cho biết lệnh in cerr nào không chạy như mong muốn. Đó là vì lỗi gì? (cerr là để báo lỗi ra màn hình, nó tương tự như cout nhưng khác với cout ở chỗ cerr sẽ in ngay lập tức, còn cout có thể trễ và có thể mất phần cuối nếu chương trình crash trước khi nó kịp in )

char\* a = new char[10];

char\* c = a + 3;

for (int i = 0; i < 9; i++) a[i] = 'a';

a[9] = '\0';

cerr <<"a: " << "-" << a << "-" << endl;

cerr <<"c: " << "-" << c << "-" << endl;

delete c;**//vì c kp là con trỏ đc khai báo new**

cerr << "a after deleting c:" << "-" << a << "-" << endl;

Trả lời bằng cách chú thích vào code.

*Gợi ý bổ sung (bôi đen để xem)*:

Tài liệu đặc tả chuẩn C++ (Mục *5.3.5.2 Delete)* viết rằng:

“*...In the second alternative (delete array), the value of the operand of delete shall be the pointer value which resulted from a previous array new-expression.72) If not, the behavior is undefined. ....*”

nghĩa là

“*...Ở lựa chọn thứ hai (delete array), giá trị của toán hạng của toán tử delete phải là giá trị con trỏ thu được từ một biểu thức new array trước đó.72) Nếu không, hành vi của chương trình là không xác định. ....*”

Có nghĩa, bạn đã gặp may nếu đoạn chương trình trên không gây lỗi khi chạy, nhưng điều đó không đảm bảo lần chạy sau hoặc đoạn code tương tự sẽ không gây lỗi.

1. Lỗi giải phóng bộ nhớ. Các biến toàn cục, biến địa phương là các biến được cấp phát và giải phóng tự động. Các biến do lập trình viên cấp phát động (toán tử new) mới là loại biến được phép giải phóng. Tuy nhiên C++ lại không kiểm soát một con trỏ đang trỏ đi đâu khi lập trình viên giải phóng vùng bộ nhớ mà nó trỏ tới. Hãy viết một chương trình thử nghiệm lỗi sử dụng bộ nhớ động theo kiểu tạo một con trỏ trỏ tới vùng nhớ của một biến địa phương sau đó giải phóng con trỏ đó. Chuyện gì xảy ra? Hãy viết câu trả lời ở dạng chú thích trong chương trình.

???

## Câu hỏi

## Bài tập

1. ***Xử lý xâu.*** Làm phiên bản khác của bài Xử lý xâu từ BT08 (ngoại trừ is\_palindrome), trong đó tất cả các tham số hàm đều ở dạng const. Nghĩa là hàm không được sửa giá trị của các xâu kí tự đầu vào, thay vào đó, nó cần cấp phát mảng mới để lưu kết quả và trả về địa chỉ của mảng kết quả.

#include <iostream>

using namespace std;

char\* reverse(char a[])

{

char \*ans = new char;

int len = 0;

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

len++;

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

ans[i] = a[len - i - 1];

ans[len] = '\0';

return ans;

}

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

{

char\* ans = new char;

int stepBack = 0, i;

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

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

if (a[i] == c)

stepBack++;

}

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

return ans;

}

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

{

char \*ans = new char;

int len;

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

ans[len] = a[len];

if (len < n){

while (len < n)

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

}

ans[len] = '\0';

return ans;

}

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

{

char \*ans = new char;

int len;

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

if (len < n){

int more = n - len;

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

ans[i] = '-';

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

ans[i] = a[i - more];

ans[n] = '\0';

}

else{

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

ans[i] = a[i];

ans[len] = '\0';

}

return ans;

}

char\* truncate(char a[], int n)

{

char\* ans = new char;

int len;

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

ans[len] = a[len];

if (len > n)

ans[n] = '\0';

else ans[len] = '\0';

return ans;

}

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;

}

char\* trim\_left(char a[])

{

char \*ans = new char;

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++)

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

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

return ans;

}

char\* trim\_right(char a[])

{

char \*ans = new char;

int len, Move = 0;

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

ans[len] = a[len];

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

Move++;

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

return ans;

}

int main()

{

char a[] = "cabcdcec ";

cout << trim\_right(a);

}

1. ***Banner (\*)***. Làm phiên bản mới hơn của bài Banner từ BT08, trong đó kích thước banner (vùng hiển thị) được nhập vào dưới dạng tham số dòng lệnh, không giới hạn giá trị lớn nhất của số dòng, số cột. Mảng chứa vùng hiển thị được cấp phát động thay vì khai báo sẵn. (chỉ dùng con trỏ và mảng thường của C)

## Trắc nghiệm

1. Khai báo nào sau đây là không hợp lệ:  
   i. int \*p = new int(10);  
   ii. int \*p = new int[10];  
   iii. int \*p = 10;  
   iv. int \*p = NULL;
2. Kết quả của a là bao nhiêu sau câu lệnh:  
   int \*p = new int[3]; p[0]=1; p[1]=2; p[2]=4; int a=&p[2]-&p[1]  
   i. 2  
   ii. 4  
   iii. 1  
   iv. Trình dịch báo lỗi
3. Phát biểu nào sau đây là không chính xác đối với mảng động  
   i. Cung cấp độ dài cho mảng động trước khi sử dụng  
   ii. Độ dài của mảng động có thể được thay đổi  
   iii. Phải cung cấp độ dài cho mảng động ngay khi khai báo
4. Phát biểu nào sau là không đúng đối với truyền mảng động cho hàm   
   i. Là kiểu truyền địa chỉ  
   ii. Là kiểu truyền con trỏ  
   iii. Là kiểu truyền giá trị  
   iv. Là kiểu truyền tham chiếu
5. Lời gọi hàm nào sau đây là không đúng  
   void printPtr(int \*p); void print(int p);  
   i. int a = 0; print(a);  
   ii. int \*a = new int(0); print(a);  
   iii. int a = 0; printPtr(a);  
   iv. int \*a = new int(0); printPtr(a);
6. Cho các khai báo sau, những câu lệnh nào không đúng về cách lập trình:  
   int \*aPtr = new int(0); int \*sPtr; int number = 4;  
   i. sPtr = aPtr;  
   ii. \*sPtr = number;  
   iii. number = aPtr;  
   iv. aPtr = &number;
7. Đoạn mã nào sau đây để in ra giá trị của một mảng động (con trỏ \*p) N phần tử:  
   i. for (int i=0; i<N; i++) cout << p+i << “ “;  
   ii. for (int i=0; i<N; i++) cout << \*p+i << “ “;  
   iii. for (int i=0; i<N; i++) cout << \*(p+i) << “ “;  
   iv. for (int i=0; i<N; i++) cout << \*p[i] << “ “;
8. Kết quả đoạn mã sau là gì:  
    int a=5, \*p1, \*p2;  
    p1 = new int(10);  
    p2 = new int(20);  
    \*p1 = a; p2 = &a; a++;  
    cout << " " << \*p1 << " " << \*p2 << endl;  
   i. 10 20  
   ii. 5 5  
   iii. 10 6  
   iv. 5 6
9. Giá trị của \*p sau đoạn mã sau là gì:  
    void a(int \*p) { \*p++; }  
    void b(int \*p) { (\*p)++; }  
    void c(int p) { p++; }  
    int main  
    {  
    int \*p = new int(5);  
    b(p); a(p); c(\*p);  
    }  
   i. 5  
   ii. 6   
   iii. 7   
   iv. 8
10. Kết quả đoạn mã sau là gì:  
     int \*p1, \*p2;  
     p1 = new int;  
     p2 = new int;  
     \*p1 = 10;  
     \*p2 = 20;  
     cout << \*p1 << " " << \*p2;  
     p1 = p2;  
     cout << " " << \*p1 << " " << \*p2;  
     \*p1 = 30;  
     cout << " " << \*p1 << " " << \*p2 << endl;  
    i. 10 20 20 20 30 20  
    ii. 10 20 10 20 30 30   
    iii. 10 10 20 20 30 30   
    iv. 10 20 20 20 30 30
11. Kết quả đoạn mã sau là gì:  
     int \*p1, \*p2, \*p3;  
     p1 = new int(10); p2 = new int(20); p3 = new int(\*p1 + \*p2);  
     p1 = p2; p2 = p3; p3 = p1;  
     cout << \*p1 << " " << \*p2 << " " << \*p3 << endl;  
    i. 10 20 30  
    ii. 20 30 10  
    iii. 20 30 20  
    iv. 30 20 10
12. Câu lệnh nào trả về giá trị phần tử thứ 3 của mảng int \*p = new int[10];  
    i. p+2;  
    ii. \*p+2;  
    iii. \*(p+2);  
    iv. \*p[2];
13. Phát biểu nào sau đây không đúng đối vởi mảng int \*p = new int[5];  
    i. p là địa chỉ của p[0]  
    ii. \*p là giá trị của p[0]  
    iii. \*p+1 là giá trị của p[1]  
    iv. p+1 là địa chỉ của p[1]