TÀI LIỆU TRAINING CUỐI KÌ 1 2020 - 2021

Nhập môn Lập trình

Ban học tập Công Nghệ Thông Tin

MỤC LỤC

I. Mảng (Array):

1. Khái niệm:

Mảng được sử dụng để lưu trữ nhiều giá trị trong một biến, thay vì khai báo các biến riêng biệt cho mỗi giá trị.

Để khai báo một mảng, xác định loại biến, chỉ định tên của mảng theo dấu ngoặc vuông và chỉ định số phần tử cần lưu trữ:

```
int numbers[3];
```

//khai báo mảng numbers có 3 phần tử, 3 phần tử chưa được khai báo

Phía trên chúng ta vừa khai báo một biến lưu trữ một mảng gồm 3 số nguyên (int). Để thêm giá trị vào, chúng ta có thể sử dụng một dãy số đặt trong dấu ngoặc nhọn và các số cách nhau bởi dấu phẩy.

Mình muốn lưu ngày, tháng, năm sinh nhật của BHT CNTT, mình làm như sau: `int numbers[3] = {11, 11, 2016} ;`

Một cách khác, thay vì khai báo số lượng phần tử mảng, bạn có thể khai báo các giá trị biến.

```
int numbers[] = {11, 11, 2016} ;
```

Cũng có thể khai báo một mảng mà không có giá trị của các phần tử và thêm chúng sau:

```
int numbers[5];
numbers[0] = 11;
numbers[1] = 11;
...
```


Truy cập các phần tử trong mảng:

Điền số thứ tự phần tử vào sau biến mảng, lưu ý thứ tự đầu tiên là 0.

```
int numbers[] = {11, 11, 2016} ;
cout << numbers[0];
```

//xuất ra phần tử đầu tiên với số thứ tự 0

Thay đổi phần tử trong mảng

Để thay đổi giá trị của một phần tử cụ thể trong mảng, hãy truy cập tới nó và gán giá trị mới.

```
numbers[0] = 12;
```

//phần tử đầu tiên (11) đã trở thành 12

```
numbers[1] *= numbers[0];
```

//phần tử thứ hai là kết quả của phép toán sau:

numbers[1] = numbers[1] * numbers[0] = 11*12 = 121;

Mảng 2 chiều

Để khai báo một biến lưu trữ một bảng giá trị (có dòng và cột), số dòng phải đặt trước số cột.

```
char letters[4][2];
```

//biến ký tự letters lưu trữ một mảng gồm 4 dòng và 2 cột

2. Bài tập tham khảo:

Bài 1: Tìm lỗi sai trong đoạn code sau.

```
#include <iostream>
using namespace std;

void nhapMang(int a, int n){
 for (int i = 0; i < n; i++){
 cin >> a[i];
 }
}

float tinh_trung_binh(int a[], int n){
 float sum = 0;
 int dem = 0;
 for (int i = 0; i < n; i++) {
 if (a[i] % 2 = 0) {
 sum += a[i];
 dem++;
 }
 }
 return sum / dem;
}

int main() {
 int k = 1000, ans;
 int a[k], n;
 cout << "n = ";
 cin >> n;

 nhapMang(a, n);

 ans = tinh_trung_binh(a, n);

 cout << ans << endl;

 return 0;
}
```

Lời giải:

```
#include <iostream>
using namespace std;

void nhapMang(int a, int n) // a phải là array ( a[ ] )
{
 for (int i = 0; i < n; i++)
 {
 cin >> a[i];
 }
}
```

```
float tinh_trung_binh(int a[], int n)
{
 float sum = 0;
 int dem = 0;
 for (int i = 0; i < n; i++) {
 if (a[i] % 2 = 0) // sai ở vị trí dấu '=', đáp án đúng là '==' {
 sum += a[i];
 dem++;
 }
 }
 return sum / dem;
}

int main() {
 int k = 1000, ans; // biến ans phải thuộc kiểu float
      int a[k], n; // giá trị khi khai báo mảng phải là một hằng số cụ thể

 cout << "n = ";
 cin >> n;

 nhapMang(a, n);

 ans = tinh_trung_binh(a, n);

 cout << ans << endl;

 return 0;
}
```

Bài 2: Tìm lỗi sai trong đoạn code sau (bài toán xoá các phần tử giống nhau trong mảng):

```
void nhapMang(int a[], int n)
{
 for (int i = 0; i < n; i++)
 {
 cin >> a[i];
 }
}

void xuatMang(int a[], int n)
{
 for (int i = 0; i < n; i++)
 {
 cout << a[i] << endl;
 }
}

void xoaPhanTu(int a[], int &n, int vt)
{
 for (int i = vt; i < n - 1; i++)
 a[i] = a[i + 1];
}

void xoaGiongNhau(int a[], int& n)
{
```

```
for (int i = 0; i < n - 1; i++)
 for (int j = i + 1; j < n; j++) {
 if (a[i] == a[j])
 xoaPhanTu(a, n, j);
 }
}

int main()
{
 int a[1000], n;

 cin >> n;

 nhapMang(a, n);

 xoaGiongNhau(a, n);

 xuatMang(a, n);

 return 0;

}
```

Lời giải:

```
void nhapMang(int a[], int n)
{
 for (int i = 0; i < n; i++)
 {
 cin >> a[i];
 }
}

void xuatMang(int a[], int n)
{
 for (int i = 0; i < n; i++)
 {
 cout << a[i] << endl;
 }
}

void xoaPhanTu(int a[], int &n, int vt)
{
 for (int i = vt; i < n - 1; i++) //phải giảm n sau khi chạy vòng lặp
a[i] = a[i + 1];
 n--;
}

void xoaGiongNhau(int a[], int& n)
{
 for (int i = 0; i < n - 1; i++)
 for (int j = i + 1; j < n; j++) //Xoá triệt để các số liền nhau rồi  mới tăng
j
 {
 if (a[i] == a[j])
```

```
xoaPhanTu(a, n, j);
 else
            j++;
 }
}

int main()
{
 int a[1000], n;

 cin >> n;

 nhapMang(a, n);

 xoaGiongNhau(a, n);

 xuatMang(a, n);

 return 0;

}
```

Bài 3: Dự đoán kết quả của chương trình sau:

```
#include<iostream>
using namespace std;

void mang(int &a[], int n)
{
 bool switch_bool = true;

 for (int i = 0; i <= n - 2; i++)
 {
 if (switch_bool)
 {
 if (a[i] > a[i + 1])
 swap(a[i], a[i + 1]);
 }
 else
 {
 if (a[i] < a[i + 1])
 swap(a[i], a[i + 1]);
 }
 switch_bool = !switch_bool;
 }
}

int main()
{
 int a1[] = { 0, 1, 3, 4, 5, 6, 7, 8, 10 };
int a2[] = { 4, 3, 2, 1, 5, 9, 8, 7, 6 };  int
n1 = sizeof(a1) / sizeof(a1[0]);  int n2 =
sizeof(a2) / sizeof(a2[0]);  mang(a1, n1);
 mang(a2, n2);
 return 0;
}
```

Mảng a1, a2 sau khi chạy chương trình là: ?

Đáp án: a1 = { 0, 3, 1, 5, 4, 7, 6, 10, 8 }

a2 = { 3, 4, 1, 5, 2, 9, 7, 8, 6 }

Bài 4: Viết chương trình xoá tất cả các chữ số trùng lặp trong một mảng (các chữ số được in ra không có số nào giống nhau).

Gợi ý giải:

```
#include <iostream>
using namespace std;

void nhapMang(int a[], int n)
{
 for (int i = 0; i < n; i++)
 {
 cin >> a[i];
 }
}

void xuatMang(int a[], int n)
{
 for (int i = 0; i < n; i++)
 {
 cout << a[i] << endl;
 }
}

int main()
{
 int a[1000], n;

 cin >> n;

 nhapMang(a, n);

 for (int i = 0; i < n; i++)
 {
 int j;
 for (j = 0; j < i; j++)
 if (a[i] == a[j])
 break;
 if (i == j)
 cout << a[i] << " ";
 }
 return 0;
}
```

Bài 5: Viết chương trình in ra các cặp số trong một mảng mà tổng của chúng bằng một số được nhập vào bàn phím.

Gợi ý giải:

```
#include <iostream>
using namespace std;

void nhapMang(int a[], int n)
{
 for (int i = 0; i < n; i++)
 {
 cin >> a[i];
 }
}


int main()
{
 int a[1000], n, sum, count = 0;

 cin >> n;

 cin >> sum;

 nhapMang(a, n);

 for (int i = 0; i < n; i++)
 for (int j = i + 1; j < n; j++)
 if (a[i] + a[j] == sum)
 {
 cout << "\n" << a[i] << "," << a[j] << endl;  count++;
 }

 cout << "Số lần xuất hiện cặp số mà tổng của nó bằng " << sum << " là ";
 cout << count << endl;

 return 0;
}
```

II. Mảng ký tự:

1. Khái niệm:

Mảng ký tự là mảng chứa toàn ký tự.

Khai báo tương tự với mảng thông thường.

```
char a[20];

char a[] = {'b', 'H', 'T', 'c', 'N', 'T',

't'}; char name[] = "Bht CNTT";
```

2. Một số hàm thường dùng:

Cho hai mảng ký tự char1 và char2.

Lấy dữ liệu nhập từ bàn phím:

```
fflush(stdin);

gets(char1);

cin.ignore();

cin.getline(char1, 50, '\n');
        //nhập mảng char1 tối đa 50 ký tự, tận cùng bằng phím enter
```

Khai báo thư viện `#include <string.h>`

```
strcpy(char2, char1);
    //sao chép chuỗi char1 vào chuỗi char2

strncpy(char2, char1, 5);
    //sao chép 5 ký tự đầu tiên của char1 vào char2

int a = strlen(char1);
    //tìm độ dài chuỗi char1

strcat(char2, char1);
    //nối char1 vào sau char2
```

3. Lưu ý về char:

Bản chất của các kí tự là con số. Các con số đại diện cho ký tự được liệt kê trong bảng mã ASCII. Một số ký tự thường dùng như

Char	*Number*	*char*	*Number*	*Char*	*Number*	*Char*	*Number*	*Char*	*Number*
space	32	B	66 *(65+1)*	N	78 *(65+13)*	Z	90 *(65+25)*	l	108 *(97+11)*
0	48	C	67 *(65+2)*	O	79 *(65+14)*	a	97	m	109 *(97+12)*
1	49 (48 + 1)	D	68 *(65+3)*	P	80 *(65+15)*	b	98 *(97+1)*	n	110 *(97+13)*
2	50 (48 + 2)	E	69 *(65+4)*	Q	81 *(65+16)*	c	99 *(97+2)*	o	111 *(97+14)*
3	51 (48 + 3)	F	70 *(65+5)*	R	82 *(65+17)*	d	100 *(97+3)*	p	112 *(97+15)*
4	52 (48 + 4)	G	71 *(65+6)*	S	83 *(65+18)*	e	101 *(97+4)*	q	113 *(97+16)*
5	53 (48 + 5)	H	72 *(65+7)*	T	84 *(65+19)*	f	102 *(97+5)*	r	114 *(97+17)*
6	54 (48 + 6)	I	73 *(65+8)*	U	85 *(65+20)*	g	103 *(97+6)*	s	115 *(97+18)*
7	55(48 + 7)	J	74 *(65+9)*	V	86 *(65+21)*	h	104 *(97+7)*	t	116 *(97+19)*
8	56 (48 + 8)	K	75 *(65+10)*	W	87 *(65+22)*	i	105 *(97+8)*	u	117 *(97+20)*
9	57 (48 + 9)	L	76 *(65+11)*	X	88 *(65+23)*	j	106 *(97+9)*	v	118 *(97+21)*
A	65	M	77 *(65+12)*	Y	89 *(65+24)*	k	107 *(97+10)*	w	119 *(97+22)*

Khai báo ký tự bằng cả hai cách đều được:

```
char a = 'a';
```

```
char a = 97;
```

4. Bài tập tham khảo:

Cho trước khoá là một hoán vị của n số (1, 2, ..., n). Khi đó để mã hoá một chuỗi ký tự ta có thể chia chuỗi thành từng nhóm n kí tự (riêng nếu nhóm cuối cùng không đủ n kí tự thì ta có thể thêm các dấu cách vào sau cho đủ) rồi hoán vị các kí tự trong từng nhóm. Sau đó, ghép lại theo thứ tự các nhóm ta được một chuỗi ký tự đã mã hoá.

Hãy hoàn thành hàm MaHoaXau bên dưới theo yêu cầu đề bài.

*Input:*Mỗi dòng gồm một chuỗi s, một số nguyên n và một chuỗi mậu khẩu k gồm n ký tự.

Output: In trên một dòng chuỗi ký tự đã mã hóa.

Input	Output
english 4 3241	gnlehs i

```
#include <iostream>
#include <cstring>
#include <stdio.h>
using namespace std;
void MaHoaXau(char s[], int n, char k[]){
  ..........................................................................
..........................................................................
..........................................................................
..........................................................................
}
int main()
{
 char s[50], k[50];
 int n;
     fflush(stdin);
     gets(s);
     cin>>n;
     fflush(stdin);
     gets(k);
 MaHoaXau(s,n,k);
 return 0;
}
```

Gợi ý giải:

```
void MaHoaXau(char s[], int n, char k[]){
//chuỗi s đã cho có bao nhiêu kí tự
 int doDaiS = strlen(s);
//kiểm tra nếu thiếu độ dài cho nhóm ký tự cuối dùng thì phải thêm khoảng trắng vào
int khoangtrang = n - doDaiS%n;
while(khoangtrang>0&&khoangtrang--) strcat(s," ");  //vòng for này in đáp án ra
 for(int i=0; i<doDaiS+khoangtrang; i += n){
 for(int j=0;j<n; j++){
 cout<<s[i+k[j]-48];
 }
    }
}
```

III. Con trỏ:

1. Khái niệm:

Chúng ta có thể dùng toán tử & để lấy ra giá trị của một biến.

Ví dụ:

```
int a = 4;
cout << a;
            //in ra giá trị của a (4)
cout << &a;
            //in ra địa chỉ trên bộ nhớ của biến a (ox6dfeec)
```

Tuy nhiên, **con trỏ** không lưu trữ bất kì giá trị nào mà *lưu trữ địa chỉ* trên bộ nhớ.

Giá trị con trỏ trỏ tới một kiểu dữ liệu cùng loại (ví dụ int hoặc char) và được khởi tạo với toán tử *. Địa chỉ của biến sẽ được gán vào con trỏ.

```
int n = 4;
      //biến n với kiểu số nguyên int
int *nmlt = &n;
      //con trỏ có tên nmlt được khởi tạo để lưu địa chỉ của biến n
cout << n <<endl;
      //xuất ra giá trị biến n (4)
cout << &n <<endl;
      //Xuất ra địa chỉ của biến n (ox6dfee8)
cout << nmlt << endl;
    //xuất ra giá trị mà con trỏ lưu trữ, chính là địa chỉ của biến n (ox6dfee8)
```

Có 3 cách để khai báo con trỏ:

```
int* n;
int *n;  //khuyến nghị
int * n;
```

Để lấy ra giá trị của biến bằng cách sử dụng con trỏ, tiếp tục dùng toán tử *.

```
int n = 4;
int *contro = &n;
cout << contro << endl;
      //xuất ra địa chỉ mà con trỏ lưu (ox6dfee8)
cout << *contro << endl;
      //xuất ra giá trị biến mà con trỏ trỏ đến (4)
```

Mọi tác động vào giá trị con trỏ đều dẫn đến sự thay đổi của biến:

```
int n = 4;
int *ctr = &n;
cout << n << endl;
      //in ra giá trị biến n (4)
cout << *ctr << endl;
      //truy cập và xuất giá trị của biến n (4)
*ctr = 2020;
      //thay đổi giá trị con trỏ
cout << *ctr << endl;
      //giá trị mới của con trỏ ctr là 2020
cout << n << endl;
      //giá trị của biến n hiện tại là 2020
```

2. Toán tử và con trỏ hằng

Hai toán tử cộng trừ đóng vai trò dịch chuyển ô nhớ của con trỏ:

```
int a = 10;
int *contro = &a;
      //khai báo con trỏ contro để lưu địa chỉ biến a
      //vì kiểu int nên 1 ô nhớ là 4 byte
cout << contro << endl;
      //xuất ra địa chỉ contro (0x6dfee8)
contro++;
      //dịch contro lên 1 ô nhớ tiếp theo
cout << contro << endl;
      // địa chỉ contro (0x6dfeec) tăng 4 byte so với phía trên
```

Con trỏ hằng có ba cách khai báo gần giống nhau nhưng cách dùng khác nhau:

```
int a = 10, b = 20;
const int * ptr = &a;
int const *ptr = &a;
      //hai cách này đều khai báo con trỏ có tính chất read-only
    Con trỏ ptr có thể thay đổi (trỏ đến biến khác) nhưng không thể thay đổi giá trị của
biến. ptr = &b;
      //không báo lỗi
 (*ptr)++;
      //error

int * const ptr = &a;
      //immutable pointer
      Không thể trỏ đến biến khác, nhưng giá trị con trỏ thay đổi được
 (*ptr)++;
      //không báo lỗi
ptr = &b;  //error
```

3. Cấp phát động với con trỏ:

Keyword để cấp phát bộ nhớ: **new**

Keyword để giải phóng bộ nhớ: **delete**

Việc cấp phát động này sẽ giúp bộ nhớ chỉ sử dụng những vùng nhớ cần thiết cho công việc tính toán mà không có thừa thãi.

Khai báo:

```
<type> *<pointer name> = new <type>(value);
int *ctro = new int;
        //Khai báo động một con trỏ kiểu int
int *ptr = new int(2020);
        //Khai báo động con trỏ ptr với giá trị ban đầu là 2020
```

Sau khi sử dụng con trỏ để tính toán (thay vì khai báo thông thường int a = … không thể xóa), chúng ta có giải phóng vùng nhớ của con trỏ này để trống bộ nhớ.

```
delete <pointer name>;
delete ctro;
delete ptr;
ctro = NULL;
ptr = NULL;
        //gán NULL để tránh hiện tượng 'con trỏ lạc'
```

Đối với khai báo mảng động, có chút khác biệt:

```
<type> *<pointer name> = new <type>[number of values]; double *ctro = new double[10];
```

Xóa mảng động:

```
delete[] <pointer name>;
<pointer name> = NULL;

delete[] ctro;
ctro = NULL;
```

Lưu ý: Cần nắm kĩ lý thuyết để tránh mắc bẫy những trường hợp kiểu thế này: `int *p1, *p2;`

```
p2 = p1;
        //Gán con trỏ cho con trỏ, p1 và p2 bây giờ sẽ cùng trỏ đến 1 địa chỉ
*p2 = *p1;
    //Gán giá trị của p1 cho giá trị của p2, hai con trỏ vẫn trỏ về 2 ô nhớ khác nhau
```

4. Bài tập tham khảo:

Bài 1: Dự đoán kết quả của chương trình sau:

```
#include <iostream>
using namespace std;
int main()
{
 int a = 32, * ptr = &a;
 char ch = 'A', & cho = ch;
 cho += a;
 *ptr += ch;
 cout << a << ", " << ch << endl;
 return 0;
}
```

Đáp án: 129, a

Giải thích: vì *cho* là tham biến nên khi giá trị biến cho thay đổi thì biến *ch* cũng thay đổi theo. Do vậy, khi cho giá trị biến *cho* cộng thêm a thì giá trị mới của biến *cho* là 65(A) + 32 = 97(a), và theo đó biến *ch* cũng mang giá trị là ch = 97 = 'a'. Sau đó khi cho *ptr cộng thêm vào một lượng *ch* thì giá trị mà biết *ptr* trỏ vào sẽ có giá trị là 32 + 97 = 129.

Bài 2: Dự đoán kết quả của chương trình sau:

```
#include <iostream>
using namespace std;
int main()
{
 int num[5];
 int* p;
 p = num;
 *p = 10;
```

```
p++;
 *p = 20;
 p = &num[2];
 *p = 30;
 p = num + 3;
 *p = 40;
 p = num;
 *(p + 4) = 50;
 for (int i = 0; i < 5; i++)
 cout << num[i] << ", ";
 return 0;
}
```

Đáp án: 10, 20, 30, 40, 50, (nhớ để ý thêm dấu ‘,’)

Bài 3: Dự đoán kết quả của chương trình sau:

```
#include <iostream>
using namespace std;
int main()
{
 const int i = 20;
 const int* const ptr = &i;
 (*ptr)++;
 int j = 15;
 ptr = &j;
 cout << i;
 return 0;
}
```

Đáp án: Compile Error

Giải thích: vì biến con trỏ ptr là con trỏ hằng trỏ vào một giá trị hằng nên giá trị và địa chỉ mà biến ptr trỏ vào là bất biến. Do đó hai dòng code "(*ptr)++;" và "ptr = &j;" đều mắc lỗi Compile Error.

Bài 4: Dự đoán kết quả của chương trình sau:

```
#include <iostream>
using namespace std;
int main()
{
 int arr[] = { 4, 5, 6, 7 };
 int* p = (arr + 1);
 cout << *arr + 10;
 return 0;
}
```

Đáp án: 14

Bài 5: Trong đoạn code dưới đây có một số lỗi nghiêm trọng, bạn có thể tìm ra chúng không?

```
#include <iostream>
using namespace std;
void nhapMang(int a[], int n)
{
 for (int i = 0; i < n; i++)
 {
 cout << "a[" << i << "]= ";
 cin >> a[i];
 }
}
void xuatMang(int a[], int n)
{
```

```
for (int i = 0; i < n; i++)  {
 cout << a[i] << endl;
 }
}
int main()
{
 int n;
 int a = 10;
 int b = 12;
 cin >> n;
 int* A = new int[n];
 int* x,y;
 x = &a;
 y = &b;
 cout << x << endl;
 cout << y << endl;
 nhapMang(A, n);
 delete A;
 xuatMang(A, n);
 return 0;
}
```

Lời giải:

```
#include <iostream>
using namespace std;
void nhapMang(int a[], int n)
{
 for (int i = 0; i < n; i++)  {
 cout << "a[" << i << "]= ";  cin
>> a[i];
 }
}
void xuatMang(int a[], int n)
{
```



```
 for (int i = 0; i < n; i++)
 {
```

```
 cout << a[i] << endl;
 }
}
int main()
{
 int n;
 int a = 10;
 int b = 12;
 cin >> n;

 int* A = new int[n];
 int* x,y; // x là con trỏ, nhưng y lại là một biến bình thường x
= &a;
 y = &b; // y không phải biến con trỏ
cout << x << endl;
 cout << y << endl;
 nhapMang(A, n);
 delete A; // delete đặt sai vị trí và cú pháp sai
xuatMang(A, n);
 delete [] A; // (phần này đáp án mới xuất hiện)
return 0;
}
```

IV. Cấu trúc (struct)

1. Khái niệm:

Là cấu trúc cho phép định nghĩa các kiểu dữ liệu có cấu trúc: dữ liệu kèm theo các hàm xử lý dữ liệu. Cách khai báo:

```
struct <tên kiểu cấu trúc>
{
<kiểu dữ liệu> <tên thành phần 1>;
 …
<kiểu dữ liệu> <tên thành phần n>;
};
```

Ví dụ:

```
struct vector
{
 double x;
 double y;
};
```

Cách truy xuất dữ liệu:
<tên biến cấu trúc>.<tên thành phần>;

Gán dữ liệu kiểu cấu trúc:

```
<biến câu trúc đích>=<biến cấu trúc nguồn>;
<biến câu trúc đích>.<tên thành phần>=<giá trị>;
```

2. Ví dụ:

Viết chương trình tính tổng 2 vector trong hệ tọa độ Đề-Các (sử dụng cấu trúc struct). **Đầu vào**

Đầu vào từ bàn phím gồm 2 dòng. Mỗi dòng chứa 2 số thực biểu diễn tọa độ x và y của một vector. Các số nguyên trên cùng một dòng cách nhau bởi một dấu cách và có giá trị tuyệt đối không vượt quá *100*.

Đầu ra

In ra màn hình 2 số thực cách nhau bởi một dấu cách, biểu diễn tọa độ x và y của vector tổng của 2 vừa nhập.

Đầu vào	Đầu ra
2 4 3 5	5 9

Gợi ý giải:

```
#include <iostream>
using namespace std;

struct Toado
{
 double x;
 double y;
};
typedef struct Toado td;

void nhap (td &vt)
{
 cin>> vt.x;
 cin>>vt.y;
}
void tinh( td A, td B )
{
 double tong1= A.x+B.x;
 double tong2= A.y+B.y;
 cout << tong1 <<" "<< tong2;
}
int main()
{
 td A;
 td B;
 nhap(A);
 nhap(B);
 tinh(A,B);
 return 0;
}
```

3. Bài tập tham khảo:

Viết chương trình BookEntry với thông tin sách gồm : tên sách, tác giả. Xuất thông tin của các cuốn sách đã nhập vào.

```
#include<iostream>
#include<stdio.h>
using namespace std;

struct BookEntry {
      char tensach[50],
      char tacgia[50];
}a[50];

int main() {
     int i, n;
     cout << "So cuon sach [nho hon 50]: ";
     cin >> n;
     cout << "Nhap thong tin ve sach\n";
     cout << "----------------------\n";

     for (i = 0; i < n; i++) {
          cout << "Thong tin ve cuon sach so " << i + 1 << "\n";
          cout << "Ten Sach :";
          cin >> a[i].tensach;
          cout << "\nTac gia :";
          cin >> a[i].tacgia;
          cout << "----------------------\n";
     }

     cout <<
     "================================================\n"; cout
     << " STT\t| Ten Sach\t| Tac gia\n";
     cout << "============================";

     for (i = 0; i < n; i++) {
          cout << "\n " << i + 1 << "\t|" << a[i].tensach << "\t|
" << a[i].tacgia;
     }

     cout << "\n========================================";
     return 0;
}
```