**//Nhập xuất mảng 1 chiều, 2 chiều**

#include <iostream>

using namespace std;

int n;

void InputArray1D(int\*& a, int& n)

{

a = new int[n];

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

{

cout << "Xin hay nhap phan tu thu " << i + 1 << ":";

cin >> a[i];

}

}

void OutputArray1D(int\*& a, int& n)

{

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

{

cout << "Day la phan tu thu " << i + 1 << " trong mang:" << \*(a + i) << endl;

}

}

void FreeArray1D(int\*& a)

{

delete[] a;

cout << "Mang vua duoc cap phat da bi huy!";

}

void InputArray2D(int\*\*& a, int n)

{

a = new int \*[n];

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

{

a[i] = new int[n];

for (int j = 0; j < n; j++)

{

cout << "Xin hay nhap phan tu a[" << i << "][" << j << "]: ";

cin >> a[i][j];

}

}

}

void OutputArray2D(int\*\*& a, int n)

{

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

{

for (int j = 0; j < n; j++)

{

cout << "a[" << i << "][" << j << "]: " << a[i][j]<<endl;

}

}

}

void FreeArray2D(int\*\*& a, int& n)

{

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

delete[] a[i];

delete[] a;

cout << "Mang 2 chieu vua duoc cap phat da bi huy!";

}

int main()

{

int\* a = NULL;

cout << "Day la bai 1:" << endl;

cout << "Moi thay nhap so phan tu n cua mang 1 chieu :";

cin >> n;

InputArray1D(a, n);

OutputArray1D(a, n);

FreeArray1D(a);

int\*\* b = NULL;

cout << endl;

cout << "Day la bai 2: " << endl;

cout << "Moi thay nhap so phan tu n cua ma tran vuong:";

cin >> n;

InputArray2D(b, n);

OutputArray2D(b, n);

FreeArray2D(b, n);

}

**//Nhập xuất mảng struct**

#include<iostream>

#include<stdlib.h>

using namespace std;

const int hsize = 2 \* sizeof(int);

struct Sophuc

{

float phanthuc, phanao;

};

istream& operator>>(istream& inDev, Sophuc& p)

{

inDev >> p.phanthuc >> p.phanao;

return inDev;

}

ostream& operator<<(ostream& os, const Sophuc& p)

{

os << p.phanthuc << " + " << p.phanao << "i ";

return os;

}

void\* origin\_addr(void\* aData) //địa chỉ lưu số lượng phần tử

{

if (aData != NULL)

return (char\*)aData - 2 \* sizeof(int);

return NULL;

}

void\* szItem\_addr(void\* aData) //địa chỉ lưu kích thước của mỗi phần tử

{

if (aData != NULL)

return (char\*)aData - sizeof(int);

return NULL;

}

void\* Data\_addr(void\* aOrigin) //địa chỉ của data

{

if (aOrigin != NULL)

return (char\*)aOrigin + 2 \* sizeof(int);

return NULL;

}

void set\_nItem(void\* aData, int nItem) //lưu số lượng phần tử của mảng vào origin\_addr

{

\*((int\*)origin\_addr(aData)) = nItem;

}

int ArrSize(void\* aData) // lấy số lượng phần tử trong mảng

{

if (aData != NULL)

return \*((int\*)origin\_addr(aData));

return 0;

}

void arrFree(void\* aData)

{

if (aData != NULL)

{

free(origin\_addr(aData));

}

}

void\* ArrResize(void\* aData, int nItem)

{

if (aData == NULL || nItem < 0) return NULL;

void\* aOrigin = origin\_addr(aData);

int szItem = \*((int\*)szItem\_addr(aData));

int sz = szItem \* nItem + hsize;

void\* anew = realloc(aOrigin, sz);

if (anew != NULL)

{

aData = Data\_addr(anew);

set\_nItem(aData, nItem);

\*((int\*)szItem\_addr(aData)) = szItem;

return aData;

}

return NULL;

}

bool ArrPushBack(void\*\* aData, void\* x)

{

int nItem = ArrSize(\*aData), szItem = \*((int\*)szItem\_addr(\*aData));

void\* anew = ArrResize(\*aData, nItem + 1);

if (anew != NULL)

{

memmove((char\*)anew + nItem \* szItem, x, szItem);

\*aData = anew;

return 1;

}

return 0;

}

void\* ArrPopBack(void\*\* aData)

{

int nItem = ArrSize(\*aData), szItem = \*((int\*)szItem\_addr(\*aData));

void\* x = malloc(szItem);

if (\*aData != NULL && nItem > 0)

{

nItem--;

memmove(x, (char\*)\*aData + nItem \* szItem, szItem);

void\* anew = ArrResize(\*aData, nItem);

if (anew != NULL) \*aData = anew;

}

return x;

}

void\* soPhucArrInput()

{

Sophuc\* a = (Sophuc\*)malloc(0);

if (a != NULL)

{

a = (Sophuc\*)Data\_addr(a);

\*((int\*)origin\_addr(a)) = 0;

\*((int\*)szItem\_addr(a)) = sizeof(Sophuc);

}

Sophuc x = { 0, 0 };

while (cin >> x)

{

ArrPushBack((void\*\*)&a, &x);

}

cin.clear();

return a;

}

void soPhucArrOut(Sophuc\* aData)

{

int nItem = ArrSize(aData), szItem = \*((int\*)szItem\_addr(aData));

cout << "Mang B gom cac so phuc la: ";

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

{

cout << \*(Sophuc\*)(aData + i);

}

cout << endl;

}

void main()

{

cout << " nhap cac phan tu so phuc\n";

Sophuc\* B = (Sophuc\*)soPhucArrInput();

Sophuc\* x = (Sophuc\*)ArrPopBack((void\*\*)&B);

cout << "\nAfter pop: \n";

soPhucArrOut(B);

cout << "\nPopped element: " << \*x << endl;

free(x);

arrFree(B);

}

**//So sánh tổng quát**

#include <iostream>

#include <memory.h>

using namespace std;

struct Fraction

{

float x;

float y;

};

ostream& operator << (ostream& os, Fraction m)

{

os << m.x << "/" << m.y;

return os;

}

template<class T, int n>

ostream& operator << (ostream& os,const T (&a)[n])

{

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

os << a[i] <<' ';

os << endl;

return os;

}

bool operator > (Fraction m, Fraction n) {

return ((m.x / m.y) > (n.x / n.y));

}

bool operator < (Fraction m, Fraction n) {

return ((m.x / m.y) < (n.x / n.y));

}

template<class T>

bool increase(T a,T b)

{

return (a > b);

}

template<class T>

bool decrease(T a, T b)

{

return (a < b);

}

template<class T>

void sort(T a[], int n, bool (\*SoSanh)(T, T))

{

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

{

for (int j = i + 1; j < n; j++)

{

if (SoSanh(a[i], a[j]))

{

T temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

}

}

int main()

{

string a[] = { "all", "taa", "bcc", "ddf" };

//float a[] = { 2,3,6,5 };

//Fraction a[] = { {2,5}, {3, 7}, {4, 6} };

int n = sizeof(a) / sizeof(a[0]);

sort(a, n, increase);

cout << a;

cout << endl;

sort(a, n, decrease);

cout << a;

return 0; }

**//Khảo sát dữ liệu double**

#include<iostream>

#include<stdlib.h>

using namespace std;

//#pragma GCC diagnostic ignored "-Wpedantic"

typedef union

{

double value;

unsigned long dword[2];

unsigned short word[4];

unsigned char bytes[8];

struct

{

unsigned long long m : 52;

unsigned long long ex : 11;

unsigned long long s : 1;

};

}doubleStruct;

void doubleDump(doubleStruct\* ds)

{

//cout << ds->bytes[0];

cout << "-------------------------------------------\n";

cout << "+ value: " << ds->value << endl;;

cout << "+ Dword: " << hex << ds->dword[0] <<" " <<ds->dword[1] << endl;

cout << "+ Word: ";

for (int i = 0; i < 4; i++)

cout << ds->word[i] << " ";

cout << "\n+ bytes: ";

for (int i = 0; i < 8; i++)

{

//printf("0x%02X ", ds->bytes[i]);

cout << (unsigned short)ds->bytes[i] << " ";

//cout << endl;

}

cout << "\nIEEE stored parts: ";

cout << "\nSign: " << ds->s;

cout << "\nExpo: " << ds->ex;

cout << "\nMantissa: " << ds->m;

}

void main()

{

double x = -12.09;

doubleStruct\* p = (doubleStruct\*)&x;

doubleDump(p);

}

**//Thêm node nhưng vẫn dữ thứ tự cho trước**

#include<iostream>

#include <string.h>

using namespace std;

struct node

{

int value;

node\* next;

};

node\* getNode(int k)

{

node\* p = new node;

if (p == NULL) return NULL;

else

{

p->value = k;

p->next = NULL;

return p;

}

}

bool ascending(int a, int b)

{

return a < b;

}

bool descending(int a, int b)

{

return a > b;

}

void ThemDS(node\* head, int val, bool (\*compare)(int, int))

{

node\* p1 = head;

node\* p2 = p1->next;

//node\* p = new node;

while (p2 && compare(p2->value, val))

{

p1 = p2;

p2 = p1->next;

}

node\* p = getNode(val);

p1->next = p;

p->next = p2;

}

void XuatDS(node\* head)

{

while (head != NULL)

{

cout << head->value << " ";

head = head->next;

}

cout << endl;

}

void DeleteList(node\* head)

{

node\* p;

while (head != NULL)

{

p = head;

head = head->next;

delete p;

}

}

int main()

{

node\* head = new node;

if(head != NULL)head->next = NULL;

int cs;

cout << "Nhap 1 de sap xep tang dan, nhap 2 de sap xep giam dan: ";

cin >> cs;

int tmp;

do

{

cout << "Hay nhap mot so, nhap 0 de dung.";

cin >> tmp;

if (tmp != 0)

{

switch (cs)

{

case 1:

ThemDS(head, tmp, ascending); break;

case 2:

ThemDS(head, tmp, descending); break;

}

XuatDS(head->next);

}

else break;

} while (1);

DeleteList(head);

}

**//File nhị phân cơ bản**

#include <stdio.h>

#include <conio.h>

#include <iostream>

using namespace std;

struct Diem {

float x;

float y;

};

void main()

{

FILE\* f = NULL;

/\*fopen\_s(&f, "Demo.bin", "wb");

if (f == NULL) return;

Diem\* toaDo = (Diem\*)malloc(10 \* sizeof(Diem));

for (int i = 0; i < 10; i++) {

cin >> toaDo[i].x >> toaDo[i].y;

fwrite(&toaDo[i], sizeof(Diem), 1, f);

}

fclose(f);\*/ //Xuất ra file nhị phân

fopen\_s(&f, "Demo.bin", "rb");

cout << "\n" << "Diem doc duoc la: ";

fseek(f, 0, 2);

long n = ftell(f);

rewind(f);

Diem\* DS = (Diem\*)malloc(n);

fread(DS, sizeof(Diem), n / sizeof(Diem), f);

for (int i = 0; i < n / sizeof(Diem); i++) {

cout << "(" << DS[i].x << ' ' << DS[i].y << ")";

}

fclose(f);

}

// Run program: Ctrl + F5 or Debug > Start Without Debugging menu

// Debug program: F5 or Debug > Start Debugging menu

// Tips for Getting Started:

// 1. Use the Solution Explorer window to add/manage files

// 2. Use the Team Explorer window to connect to source control

// 3. Use the Output window to see build output and other messages

// 4. Use the Error List window to view errors

// 5. Go to Project > Add New Item to create new code files, or Project > Add Existing Item to add existing code files to the project

// 6. In the future, to open this project again, go to File > Open > Project and select the .sln file

**//Các hàm xử lý chuỗi (thư viện string.h)**

**Cộng chuỗi**

**Hàm strcat()**

**Cú pháp:** char \*strcat(char \*des, const char \*source)

Hàm này có tác dụng Nối xâu ký tự source vào sau xâu ký tự des.

**Chú ý:** des phải đủ lớn để chứa source.

**Xác định độ dài chuỗi**

**Hàm strlen()**

Cú pháp: int strlen(const char\* s)

**Đổi một ký tự thường thành ký tự hoa**

– **Hàm toupper()**: Hàm toupper() (trong ctype.h) được dùng để chuyển đổi một ký tự thường thành ký tự hoa.

Cú pháp: char toupper(char c)

Đổi chuỗi chữ thường thành chuỗi chữ hoa

– **Hàm strupr()**: Hàm struppr() được dùng để chuyển đổi chuỗi chữ thường thành chuỗi chữ hoa, kết quả trả về của hàm là một con trỏ chỉ đến địa chỉ chuỗi được chuyển đổi.

Cú pháp: char \*strupr(char \*s)

**Đổi chuỗi chữ hoa thành chuỗi chữ thường**

– **Hàm strlwr()**: Muốn chuyển đổi chuỗi chữ hoa thành chuỗi toàn chữ thường, ta sử dụng hàm strlwr(), các tham số của hàm tương tự như hàm strupr().

Cú pháp: char \*strlwr(char \*s)

**Sao chép chuỗi**

– **Hàm strcpy()**: Hàm này được dùng để sao chép toàn bộ nội dung của chuỗi nguồn vào chuỗi đích.

Cú pháp: char \*strcpy(char \*Des, const char \*Source)

**Sao chép một phần chuỗi**

**– Hàm strncpy()**: Hàm này cho phép chép n ký tự đầu tiên của chuỗi nguồn sang chuỗi đích.

Cú pháp: char \*strncpy(char \*Des, const char \*Source, size\_t n)

**- memcpy():** Cú pháp của hàm như sau:

void \* memcpy(void \* destination, const void \* source, size\_t num);

Tác dụng của hàm là sao chép num byte từ vị trí mà source trỏ tới đến vị trí mà destination trỏ tới.

**Trích một phần chuỗi**

**Hàm strchr()**: Để trích một chuỗi con của một chuỗi ký tự bắt đầu từ một ký tự được chỉ định trong chuỗi cho đến hết chuỗi, ta sử dụng hàm strchr().

Cú pháp :  char \*strchr(const char \*str, int c)

Lưu ý :

- Nếu ký tự đã chỉ định không có trong chuỗi, kết quả trả về là NULL ;

- Kết quả trả về của hàm là một con trỏ, con trỏ này chỉ đến ký tự c được tìm thấy đầu tiên trong chuỗi str.

**Tìm kiếm nội dung chuỗi**

– **Hàm strstr()**: Hàm strstr() được sử dụng để tìm kiếm sự xuất hiện đầu tiên của chuỗi s2 trong chuỗi s1.

Cú pháp: char \*strstr(const char \*s1, const char \*s2)

**So sánh chuỗi**

**Hàm strcmp()**: Để so sánh hai chuỗi theo từng ký tự trong bảng mã Ascii, ta có thể sử dụng hàm strcmp().

Cú pháp: int strcmp(const char \*s1, const char \*s2)

Hai chuỗi s1 và s2 được so sánh với nhau, kết quả trả về là một số nguyên (số này có được bằng cách lấy ký tự của s1 trừ ký tự của s2 tại vị trí đầu tiên xảy ra sự khác nhau).

- Nếu kết quả là số âm (-1), chuỗi s1 nhỏ hơn chuỗi s2;

- Nếu kết quả là 0, hai chuỗi bằng nhau;

- Nếu kết quả là số dương (1), chuỗi s1 lớn hơn chuỗi s2.

**memcmp():** Cú pháp của hàm này như sau:

int memcmp(const void \* ptr1, const void \* ptr2, size\_t num);

Tác dụng: so sánh giá trị các vùng nhớ mà ptr1 và ptr2 trỏ tới theo từng byte, sẽ dừng lại khi so sánh đủ num byte. Khác với strcmp() luôn dừng lại khi gặp kí tự ‘\0’.

Giá trị trả về:

- (-1): Khi giá trị của byte đầu tiên không trùng của ptr1 nhỏ hơn ptr2.

- 0: Khi giá trị của 2 vùng nhớ bằng nhau

- (1): Khi giá trị của byte đầu tiên không trùng của ptr1 lớn hơn ptr2.

**So sánh chuỗi**

– **Hàm stricmp()**: Hàm này thực hiện việc so sánh trong n ký tự đầu tiên của 2 chuỗi s1 và s2, giữa chữ thường và chữ hoa không phân biệt.

Cú pháp: int stricmp(const char \*s1, const char \*s2)

Kết quả trả về tương tự như kết quả trả về của hàm strcmp()

Khởi tạo chuỗi

– **Hàm memset()**: Hàm này được sử dụng để đặt n ký tự đầu tiên của chuỗi là ký tự c.

Cú pháp: memset(char \*Des, int c, size\_t n)

**Đổi từ chuỗi ra số**

**hàm atoi(), atof(), atol()** (trong stdlib.h): Để chuyển đổi chuỗi ra số, ta sử dụng các hàm trên.

Cú pháp :

**int atoi(const char \*s)** : chuyển chuỗi thành số nguyên

**long atol(const char \*s)** : chuyển chuỗi thành số nguyên dài

**float atof(const char \*s)** : chuyển chuỗi thành số thực

# **Chuyến số thành chuỗi**

Hàm **itoa()** để chuyển số thành chuỗi trong C như sau:

**itoa (num, target, base)**

**Trong đó:**

- num là số nguyên cần chuyển thành chuỗi

- target là chuỗi đích để chứa num sau khi chuyển thành chuỗi

- base là cơ số chuyển đổi, nhằm xác định một giá trị số nguyên, chuyển đổi nó thành một giá trị cơ bản và lưu trữ nó trong một bộ đệm. Nếu cơ số là 10 và giá trị là âm, thì chuỗi kết quả được đặt trước dấu trừ ( -).

**Bảng định dạng xuất trong C**

Dưới đây là bảng các định dạng xuất thông dụng được sử dụng trong các hàm xuất dữ liệu trong C như **printf(),fprintf(),sprintf().**

| **Chuỗi định dạng** | **Đại diện cho kiểu ký tự** | **Ý nghĩa** | **Ví dụ** |
| --- | --- | --- | --- |
| %c | char | Xuất ra một ký tự | "%c" |
| %s | char \* | Xuất ra một chuỗi ký tự | "%8s", "%-10s" |
| %d | int, short | Xuất ra một số nguyên dưới dạng thập phân | "%-2d","%03d" |
| %u | unsigned int, unsigned short | Xuất ra một số nguyên dưới dạng thập phân không dấu | "%2u","%02u" |
| %x | int, short, unsigned int, unsigned short | Xuất ra một số nguyên dưới dạng thập lục phân | %04x" |
| %o | int, short,unsigned int, unsigned short | Xuất ra một số nguyên dưới dạng bát phân | "%06o","%03o" |
| %f | float | Xuất ra một số thực | "%5.2f" |
| %e | float | Xuất ra một số thực dưới dạng số mũ | "%5.3e" |
| %g | float | Xuất ra một số thực dưới dạng phù hợp nhất | "%g" |
| %ld | long | xuất ra số nguyên chính xác kép ở dạng thập phân | "%-10ld" |
| %lu | unsigned long | xuất ra số nguyên chính xác kép ở dạng thập phân không dấu | "%10lu" |
| %lo | long, unsigned long | Xuất ra số nguyên chính xác kép trong hệ bát phân | "%12lo" |
| %lx | long, unsigned long | xuất ra số nguyên chính xác kép ở hệ thập lục phân | "%08lx" |
| %lf | double, unsigned long | xuất ra số thực chính xác gấp đôi | "%8.3lf" |
| %a | double | Xuất ra một số thực chính xác kép thập lục phân | "%a" |