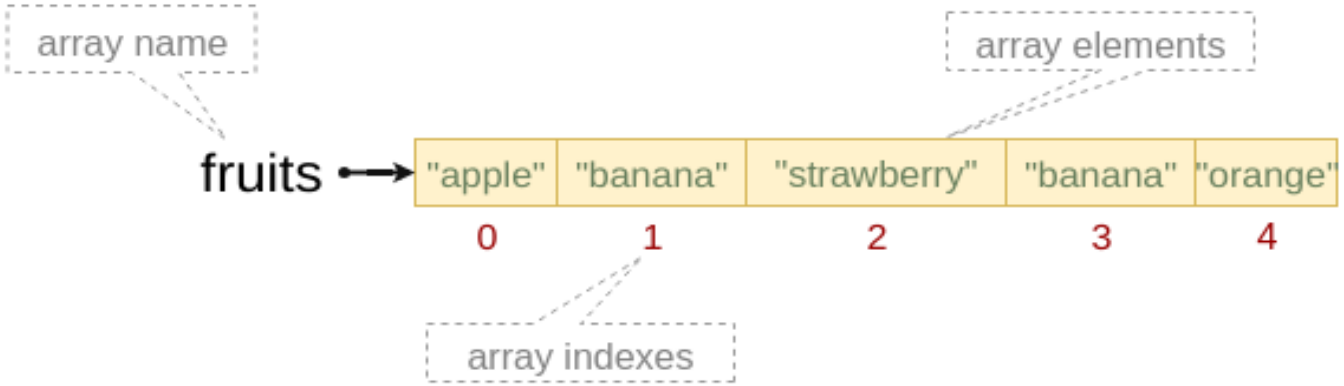
KỸ THUẬT LẬP TRÌNH

CHƯƠNG 4. MẢNG, CHUỖI



Liên hệ GV: thuong.bui@ut.edu.vn

4.1 Mảng (Array)

Mảng là một cấu trúc dữ liệu được sử dụng để lưu trữ một tập hợp các giá trị cùng loại (kiểu dữ liệu) dưới một tên biến duy nhất. Mỗi giá trị trong mảng được gọi là một phần tử và được xác định bởi một chỉ số (index) duy nhất.

Các đặc điểm chính của mảng:

- **Kích thước cố định:** Thông thường, kích thước của mảng được xác định khi khai báo và không thể thay đổi trong quá trình thực thi chương trình.
- **Các phần tử cùng kiểu:** Tất cả các phần tử trong mảng phải có cùng một kiểu dữ liệu (ví dụ: số nguyên, số thực, chuỗi).
- **Truy cập ngẫu nhiên:** Ta có thể truy cập trực tiếp đến bất kỳ phần tử nào trong mảng bằng cách sử dụng chỉ số của nó.

4.1 Mảng (Array)

Tại sao sử dụng mảng?

- **Tổ chức dữ liệu:** Tổ chức dữ liệu một cách hiệu quả, dễ dàng truy xuất và xử lý.
- **Tiết kiệm bộ nhớ:** Thay vì tạo nhiều biến riêng biệt, mảng giúp bạn sử dụng bộ nhớ một cách hiệu quả hơn.
- **Vòng lặp:** Mảng kết hợp rất tốt với các vòng lặp (for, while) để thực hiện các thao tác trên từng phần tử của mảng.

Mảng được sử dụng rộng rãi trong lập trình để giải quyết nhiều bài toán khác nhau, chẳng hạn như:

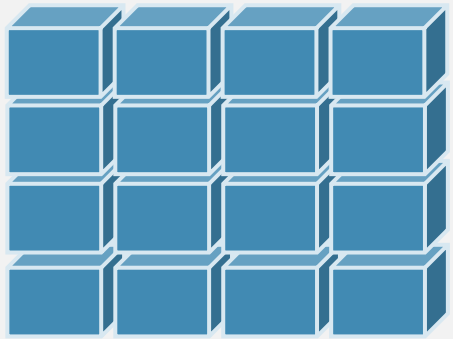
- **Xử lý dữ liệu:** Lưu trữ và xử lý các danh sách số, văn bản, hình ảnh,...
- **Tạo các cấu trúc dữ liệu phức tạp**: Mảng là nền tảng để xây dựng các cấu trúc dữ liệu khác như ma trận, danh sách liên kết,...
- **Thực hiện các thuật toán**: Nhiều thuật toán trong khoa học máy tính sử dụng mảng để lưu trữ và xử lý dữ liệu.

4.1 Mảng (Array)

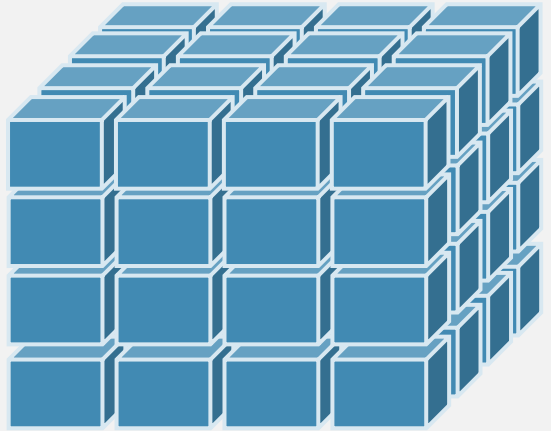
Phân loại mảng:

- ❖ Mảng có thể là mảng 1 chiều hay mảng nhiều chiều.
- ❖ Mảng n chiều (n>1) có thể được coi như mảng 1 chiều mà mỗi phần tử của nó là mảng n-1 chiều.
- ❖ Số phần tử của mảng nhiều chiều bằng tích của kích thước các chiều.

Mảng 1 chiều



Mảng 2 chiều



Mảng 3 chiều

4.1.1 Mảng một chiều (One-Dimensional Array)

Khai báo:

```
< Kiểu_dữ_liệu > <Tên_mảng>[Số_phần_tử ] ; // không khởi tạo
< Kiểu_dữ_liệu > <Tên_mảng>[Số_phần_tử ] = {Dãy_giá_trị } ;  /* có khởi tạo */
< Kiểu_dữ_liệu > <Tên_mảng>[ ] = {Dãy_giá_trị } ;  // có khởi tạo
```

- **<Kiểu_dữ_liệu>** có thể char, int, float,...
- **Tên_mảng:** theo quy tắc đặt tên.
- **Số_phần_tử** (kích thước của mảng):
  - Phải được xác định ngay tại thời điểm khai báo.
  - Phải là hằng số.
- **Dãy_giá_trị:** Giá trị các phần tử của mảng, cách nhau bởi dấu phẩy

4.1.1 Mảng một chiều (One-Dimensional Array)

```
#include<iostream>
using namespace std ;
int main(){
    int A[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};
    int B[]   = {1, 2, 3};
    int C[10] = {1, 2, 3};
    int D[5];
    D[1] = 3, D[3] =9;
    cout<< "A[0] = " <<A[0]<< "\tA[7] = " <<A[7]<<endl;
    cout<< "B[0] = " <<B[0]<< "\tB[2] = " <<B[2]<<endl;
    cout<< "C[0] = " <<C[0]<< "\tC[7] = " <<C[7]<<endl;
    cout<< "D[0] = " <<D[0]<< "\tD[1] = " <<D[1]<< "\tD[3] = " <<D[3]<<endl;
    system("pause");
}
```

Kết quả chạy chương trình:

```
A[0] = 1        A[7] = 8
B[0] = 1        B[2] = 3
C[0] = 1        C[7] = 0
D[0] = 6422124  D[1] = 3        D[3] = 9
```

4.1.1 Mảng một chiều (One-Dimensional Array)

Truy xuất các phần tử trong mảng:

Tên_mảng[chỉ_số]

Trong đó chỉ_số là số nguyên bắt đầu từ 0 đến n-1, với n là kích thước của mảng.

Ví dụ: Mảng A[] = {4, -7, 3, 2}

Chỉ số	0	1	2	3
Mảng A	4	-7	3	2
Phần tử	A[0]	A[1]	A[2]	A[3]

4.1.1 Mảng một chiều (One-Dimensional Array)

Ví dụ: Nhập xuất dữ liệu cho dãy số nguyên sử dụng mảng một chiều.

```
#include<iostream>
using namespace std ;
int main(){
    int n, i, A[100]; //Mảng A chứa tối đa 100 phần tử
    cout<< "Nhap so luong phan tu: ";
    cin>>n;
    cout<< "Nhap du lieu cho day so:\n";
    for(i=0; i<n; i++){
        cout<< "So thu "<<i+1<< ": ";
        cin>>A[i];
    }
    cout<< "Day so vua nhap la:\n";
    for(i=0; i<n; i++)
        cout<<A[i]<< ((i!=n-1) ? ", ":"");
       /*Các phần tử cách nhau bởi dấu , */
    cout<<endl;
    system("pause");
}
```

Kết quả chạy chương trình:

```
Nhap so luong phan tu: 5
Nhap du lieu cho day so:
So thu 1: 6
So thu 2: 5
So thu 3: 2
So thu 4: 9
So thu 5: -6
Day so vua nhap la:
6, 5, 2, 9, -6
```

4.1.1 Mảng một chiều (One-Dimensional Array)

Hàm và mảng một chiều:

```
#include<iostream>
using namespace std ;
void nhap(int arr[], int size){
    for(int i=0; i<size; i++){
        cout<<"Nhap phan tu thu "<<i+1<<": ";
        cin>>arr[i];
    }
}
void xuat(int arr[], int size){
    for(int i=0; i<size; i++)
        cout<<arr[i]<<" ";
}
int main(){
    int a[100], n;
    cout<<"Nhap so phan tu cua mang: "; cin>>n;
    nhap(a, n);
    cout<<"Mang vua nhap la: \n"; xuat(a, n);
    cout<<endl; system("pause");
}
```

Kết quả chạy chương trình:

```
Nhap so phan tu cua mang: 5
Nhap phan tu thu 1: 9
Nhap phan tu thu 2: 7
Nhap phan tu thu 3: 0
Nhap phan tu thu 4: 12
Nhap phan tu thu 5: 4
Mang vua nhap la:
9 7 0 12 4
```

Ghi chú: Mảng có thể thay đổi nội dung sau khi thực hiện hàm.

4.1.2 Mảng hai chiều (Two-Dimensional Array hay 2D Array)

Khái niệm:

Mảng 2 chiều là mảng trong đó mỗi phần tử của nó là 1 mảng 1 chiều, các mảng 1 chiều này ta gọi là các hàng (hay các dòng) của mảng 2 chiều, thứ tự trong các mảng 1 chiều đó ta gọi là cột của mảng 2 chiều.

Hình sau minh họa mảng 2 chiều A với 3 dòng và 4 cột:

	0	1	2	3
0	A[0][0]	A[0][1]	A[0][2]	A[0][3]
1	A[1][0]	A[1][1]	A[1][2]	A[1][3]
2	A[2][0]	A[2][1]	A[2][2]	A[2][3]

Ghi chú: Các phần tử của mảng 2 chiều được lưu trữ liên tiếp trong bộ nhớ, theo thứ tự hàng ưu tiên. Nghĩa là, các phần tử của hàng đầu tiên được lưu trữ trước, tiếp theo là các phần tử của hàng thứ hai, và cứ thế cho đến hết.

4.1.2 Mảng hai chiều (Two-Dimensional Array hay 2D Array)

Khai báo:

< Kiểu_dữ_liệu > <Tên_mảng>[Số_hàng] [Số_cột] ;

- Trong khai báo cũng có thể được khởi tạo bằng dãy các dòng giá trị, các dòng cách nhau bởi dấu phẩy, mỗi dòng được bao bởi cặp ngoặc {} và toàn bộ giá trị khởi tạo nằm trong cặp dấu {}.
- Sử dụng: Để truy nhập phần tử của mảng ta sử dụng tên mảng kèm theo 2 chỉ số chỉ vị trí hàng và cột của phần tử. Các chỉ số này có thể là các biểu thức thực, khi đó C++ sẽ tự chuyển kiểu sang nguyên.

Ví dụ:

```
int A[3][4] = {1,2,3,4,5,6,7,8,9,0,10,11};
int B[3][4] = {{1,2,3},{4},{5,6,7,8}};
```

Ghi chú: phần tử nào không được khởi tạo sẽ có giá trị mặc định bằng 0

4.1.2 Mảng hai chiều (Two-Dimensional Array hay 2D Array)

Mảng 2 chiều và hàm:

```
#include<iostream>
using namespace std ;
void printArr(int a[][4], int size){
    for(int i = 0; i < size;i++){
        for(int j=0;j<4;j++){
            cout<<a[i][j]<<"\t";
        }
        cout<<endl;
    }
}
int main() {
    int A[3][4] = {1,2,3,4,5,6,7,8,9,0,10,11};
    int B[3][4] = {{1,2,3},{4},{5,6,7,8}};
    int C[3][4] = {{1,2,3},{4}};
    cout<< "Mang A:\n";  printArr(A,3);
    cout<< "Mang B:\n" ;  printArr(B,3);
    cout<< "Mang C:\n";  printArr(C,3);
    system("pause");
}
```

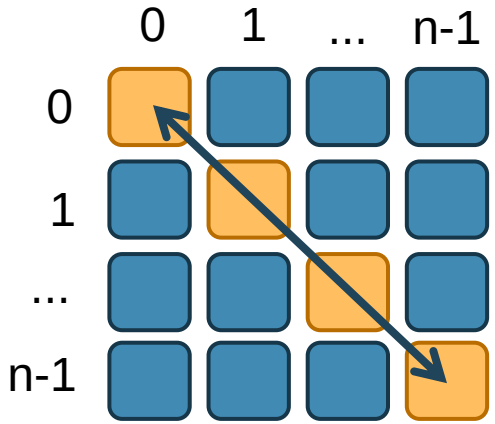
Kết quả chạy chương trình:

```
Mang A:
1       2       3       4
5       6       7       8
9       0       10      11
Mang B:
1       2       3       0
4       0       0       0
5       6       7       8
Mang C:
1       2       3       0
4       0       0       0
0       0       0       0
```

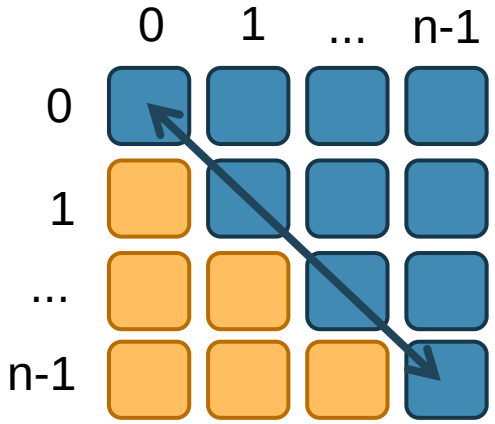
Lưu ý: *Khi khởi tạo giá trị cho mảng, ta có thể để trống chỉ số hàng nhưng phải chỉ ra chỉ số cột.*

4.1.2 Mảng hai chiều (Two-Dimensional Array hay 2D Array)

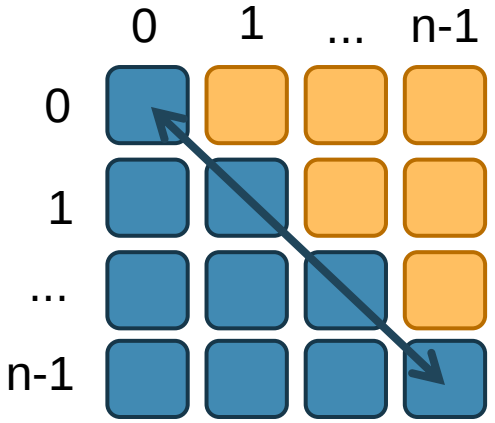
Liên hệ giữa mảng 2 chiều và ma trận vuông:



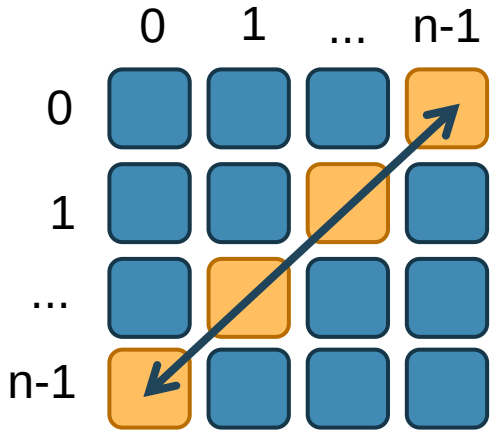
dòng = cột



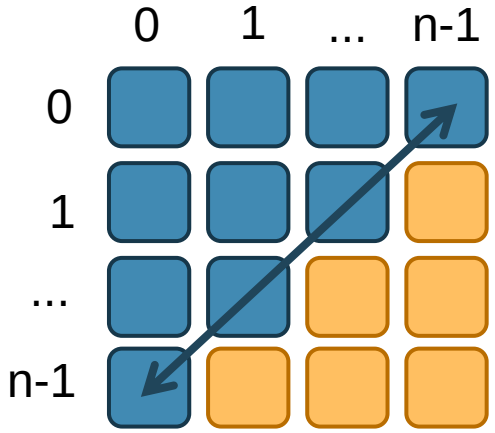
dòng > cột



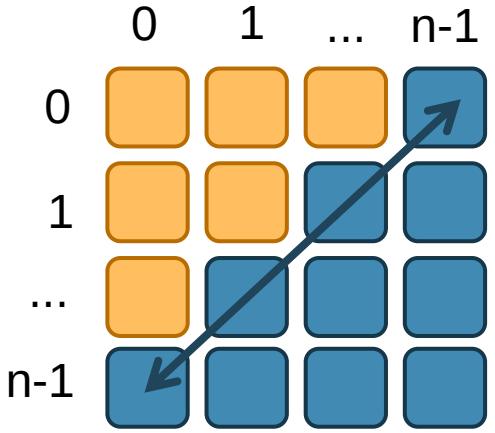
dòng < cột



dòng + cột = n-1



dòng + cột > n-1



dòng + cột < n-1

4.2 Ký tự và chuỗi kỹ tự

Khái niệm:

- Kiểu char chỉ chứa được một ký tự. Để lưu trữ một chuỗi (nhiều ký tự) ta sử dụng mảng (một chiều) các ký tự.
- Chuỗi ký tự kết thúc bằng ký tự '\0' (null).
- Độ dài chuỗi = kích thước mảng – 1

Ví dụ:

char Hoten[30]; *// Dài tối đa 29 ký tự*
char NgaySinh[9]; *//Dài tối đa 8 ký tự*

0	1	2	3	4	5	6	7
T	**I**	**N**	**H**	**O**	**C**	**\0**	
T	**I**	**N**	**\0**	H	O	C	\0
\0	**T**	**I**	**N**	**H**	**O**	**C**	\0

4.2 Ký tự và chuỗi kỹ tự

Khai báo chuỗi:

char <tên chuỗi>[độ dài] ; // không khởi tạo

char <tên chuỗi>[độ dài] = xâu kí tự ; // có khởi tạo

char <tên chuỗi>[] = xâu kí tự ; // có khởi tạo

Ví dụ: Khai báo và khởi tạo chuỗi

```
char s[10]={'c', 'h', 'a', 'o', ' ', 'b', 'a', 'n','\0'};
char s[10]="chao ban"; //Tự động thêm '\0'
char s[]={'c', 'h', 'a', 'o', ' ', 'b', 'a', 'n','\0'};
char s[]="chao ban"; //Tự động thêm '\0'
```

4.2 Ký tự và chuỗi kỹ tự

Nhập dữ liệu cho ký tự, chuỗi:

- Nhập ký tự: **cin.get(Biến ký tự);**
- Nhập chuỗi ký tự: **cin.getline(s, n);**

Ví dụ:

```
#include<iostream>
using namespace std ;
int main() {
    char c, Str[100];
    cout<<"Nhap mot ki tu: ";
    cin.get(c); cin.ignore(1);
    cout<<"Nhap mot chuoi: ";
    cin.getline(Str, 20);
    cout<<"======================\n";
    cout<<"Ki tu vua nhap la: "<<c<<"\n";
    cout<<"Chuoi vua nhap la: "<<Str<<"\n";
    system("pause");
}
```

Kết quả chạy chương trình:

```
Nhap mot ki tu: h
Nhap mot chuoi: Lap trinh C++
======================
Ki tu vua nhap la: h
Chuoi vua nhap la: Lap trinh C++
```

4.2 Ký tự và chuỗi kỹ tự

Hàm sao chép chuỗi strcpy(s, t): Sao chép t vào s kể cả ký tự '\0'

Ví dụ:

```
#include<iostream>
#include<string.h>
using namespace std;
int main(){
    char s[10], t[10];
    strcpy(t,"Face");//được, gán "Face" cho t
    strcpy(s,t);//được, sao chép t sang s
    cout << s << " to " << t<<'\n';
    system("pause");
}
```

Kết quả chạy chương trình:

```
Face to Face
```

4.2 Ký tự và chuỗi kỹ tự

Hàm sao chép n ký tự từ chuỗi strncpy(s, t, n): Sao chép n kí tự của t vào s. Hàm này chỉ làm nhiệm vụ sao chép, không tự động gắn kí tự kết thúc xâu cho s.

Ví dụ:

```
#include<iostream>
#include<string.h>
using namespace std;
int main(){
    char s[25], t[25] = "Ky thuat lap trinh C++";
    strncpy(s, t, 8); //copy 8 kí tự "Ky thuat" vào s
    s[8]=' '; //Gán kí tự khoảng trống cho s[8]
    strncpy(s+9, t+9, 9); //copy "lap trinh" vào s từ vị trí thứ 10
    s[18] = '\0'; //kết thúc xâu s
    cout <<"Chuoi s la: " << s<<'\n'; //Xuất ra “Ky thuat lap trinh”
    system("pause");
}
```

Kết quả chạy chương trình:

```
Chuoi s la: Ky thuat lap trinh
```

4.2 Ký tự và chuỗi kỹ tự

Hàm nối chuỗi strcat(s, t): Nối một bản sao của t vào sau s (thay cho phép +). Hiển nhiên hàm sẽ loại bỏ kí tự kết thúc xâu s trước khi nối thêm t. Việc nối sẽ đảm bảo lấy cả kí tự kết thúc của xâu t vào cho s (nếu s đủ chỗ) .

Ví dụ:

```
#include<iostream>
#include<string.h>
using namespace std;
int main(){
    char a[100] = "Ky thuat", b[6] = "C++";
    strcat(a, " lap trinh ");
    strcat(a, b);
    cout<<a<<endl;
    system("pause");
}
```

Kết quả chạy chương trình:

```
Ky thuat lap trinh C++
```

4.2 Ký tự và chuỗi kỹ tự

Một số hàm xử lý chuỗi khác trong thư viện <string.h>.

- **strncat(s, t, n):** Nối bản sao n kí tự đầu tiên của xâu t vào sau xâu s. Hàm tự động đặt thêm dấu kết thúc xâu vào s sau khi nối xong. Tương tự, có thể sử dụng cách viết strncat(s, t+k, n) để nối n kí tự từ vị trí thứ k của xâu t cho s.
- **strcmp(s, t):** Hàm so sánh 2 xâu s và t (thay cho các phép toán so sánh). Giá trị trả lại là hiệu 2 kí tự khác nhau đầu tiên của s và t. Từ đó, nếu s < t thì hàm trả lại giá trị âm, bằng 0 nếu s==t, và dương nếu s > t. Trong trường hợp chỉ quan tâm đến so sánh bằng, nếu hàm trả lại giá trị 0 là 2 xâu bằng nhau và nếu giá trị trả lại khác 0 là 2 xâu khác nhau.

4.2 Ký tự và chuỗi kỹ tự

Một số hàm xử lý chuỗi khác trong thư viện <string.h>.

- **strncmp(s, t, n):** Giống hàm strcmp(s, t) nhưng chỉ so sánh tối đa n kí tự đầu tiên của hai xâu.
- **stricmp(s, t):** Như strcmp(s, t) nhưng không phân biệt chữ hoa, thường.
- **strupr(s):** Hàm đổi xâu s thành in hoa, và cũng trả lại xâu in hoa đó.
- **strlwr(s):** Hàm đổi xâu s thành in thường, kết quả trả lại là xâu s.
- **strlen(s):** Hàm trả giá trị là độ dài của xâu s.
- **strrev(s):** Đảo ngược thứ tự các ký tự trong xâu s (Trừ ký tự '\0')

4.3 Mảng chuỗi

Mảng chuỗi là mảng trong đó các phần tử của nó là một chuỗi.

Cú pháp khai báo :

```
char Tên_mảng[Kích_thước][Độ_dài_chuỗi];
```

Trong đó:

- **Kích_thước** *là số lượng phần tử tối đa của mảng*
- **Độ_dài_chuỗi** *là kích thước của mỗi phần tử.*

Để truy cập đến các phần tử ta chỉ cần viết tên mảng kèm chỉ số của nó ở trong cặp dấu **[]**.

4.3 Mảng chuỗi

```
#include<iostream>
using namespace std ;
int main(){
    char sinhvien[50][33]; int i,n;
    cout<< "Nhap so luong sinh vien: ";
    cin>>n; cin.ignore(1);
    for(i=0; i<n; i++){
        cout<< "Ho va ten cua sinh vien thu
"<<i+1<<" :\n";
        cin.getline(sinhvien[i], 33);
    }
    cout<< "======================\n";
    cout<< "Danh sach sinh vien vua nhap la:\n";
    cout<< "STT\tHo va ten\n";
    for(i=0; i<n; i++)
        cout<<i+1<< "\t"<<sinhvien[i]<<endl;
    system("pause");
}
```

Kết quả chạy chương trình:

```
Nhap so luong sinh vien: 4
Ho va ten cua sinh vien thu 1 :
Nguyen Minh Tuan
Ho va ten cua sinh vien thu 2 :
Cao Van Dat
Ho va ten cua sinh vien thu 3 :
Lam Nhat Anh
Ho va ten cua sinh vien thu 4 :
Tran Vu Minh
======================
Danh sach sinh vien vua nhap la:
STT     Ho va ten
1       Nguyen Minh Tuan
2       Cao Van Dat
3       Lam Nhat Anh
4       Tran Vu Minh
```

4.4 Lớp string

Trong C++, lớp string cung cấp một cách hiệu quả và linh hoạt để làm việc với các chuỗi ký tự.

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string str1;
    string str2 = "Hello";
    string str3(10, 'a');
    string str4 = str2 + " World!";
    string str5("Ky thuat lap trinh");
    cout << "str1: " << str1 << endl;
    cout << "str2: " << str2 << endl;
    cout << "str3: " << str3 << endl;
    cout << "str4: " << str4 << endl;
    cout << "str5: " << str5 << endl;
    system("pause");
}
```

Kết quả chạy chương trình:

```
str1:
str2: Hello
str3: aaaaaaaaaa
str4: Hello World!
str5: Ky thuat lap trinh
```

4.4 Lớp string

Nhập dữ liệu cho string:

Nhập từ bàn phím: getline(cin, đối tượng, ký tự kết thúc); //Nhận cả ký tự trống(space, tab). Mặc định là ký tự xuống dòng (‘\n’)
getline(cin,đối tượng): khi gặp ký tự ‘\n’ kết thúc chuỗi.
getline(cin,đối tượng,’.’): khi gặp ký tự ‘.’ kết thúc chuỗi.

Truy cập từng phần tử (ký tự):

tên_chuỗi[chỉ số]: như mảng ký tự.
tên_chuỗi.at(chỉ số);

4.4 Lớp string

Ví dụ về nhập/xuất string:

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string str;
    cout<<"Nhap chuoi: ";
    getline(cin, str);
    cout<<"Ky tu thu nhat cua chuoi: \""<<str<<"\" la: "<<str[0]<<endl;
    cout<<"Ky tu thu hai cua chuoi: \""<<str<<"\" la: "<<str.at(1)<<endl;
    system("pause");
}
```

Kết quả chạy chương trình:

```
Nhap chuoi: abcdef123
Ky tu thu nhat cua chuoi: "abcdef123" la: a
Ky tu thu hai cua chuoi: "abcdef123" la: b
```

4.4 Lớp string

Một số phương thức cơ bản của lớp string:

- length(), size(): độ dài của chuỗi (số lượng ký tự).
- push_back(): thêm 1 ký tự vào cuối chuỗi. Trả về string
- pop_back(): xóa 1 ký tự ở cuối chuỗi. Trả về string
- append(chuỗi nối vào): nối chuỗi. s.append(s1): nối s1 vào s
- compare(chuỗi so sánh): trả về int. s.compare(s1): so sánh s với s1(trả về 0: s==s1, 1: s>s1, -1: s<s1)
- insert(vị trí, chuỗi cần chèn): string. Chèn chuỗi cần chèn vào chuỗi từ vị trí.
- find(chuỗi cần tìm): Trả về vị trí đầu tiên của chuỗi cần tìm.
- substr(vị trí, độ dài chuỗi con): string. Cho chuỗi con từ vị trí có độ dài chuỗi con.
- clear(): void. Xóa chuỗi.
- empty(): bool. true nếu chuỗi rỗng, ngược lại false.
- erase(vị trí, số ký tự cần xóa): string
- c_str(): const char* . Chuyển chuỗi string sang mảng chuỗi.

4.4 Lớp string

Một số phương thức cơ bản của lớp string:

- capacity(): int. Trả về số bytes của string
- assign(str): Gán một chuỗi mới cho chuỗi hiện tại.
- clear(): Xóa toàn bộ nội dung của chuỗi.
- replace(pos, len, str): Thay thế một đoạn chuỗi bắt đầu từ vị trí pos và có độ dài len bằng chuỗi str.
- rfind(str): Tìm vị trí xuất hiện cuối cùng của chuỗi con str.
- find_first_of(str): Tìm vị trí xuất hiện đầu tiên của bất kỳ ký tự nào trong chuỗi str.
- find_last_of(str): Tìm vị trí xuất hiện cuối cùng của bất kỳ ký tự nào trong chuỗi.
- front(): Trả về ký tự đầu tiên.
- back(): Trả về ký tự cuối cùng.
- Các toán tử ==, !=, <, >, <=, >=: So sánh hai chuỗi theo thứ tự từ điển.

BÀI TẬP CHƯƠNG 4

Bài 1. Viết chương trình nhập vào 1 dãy số nguyên A gồm n chữ số (n<=100):

a) Xuất dãy A ra màn hình
b) Tìm phần tử có giá trị lớn nhất và nhỏ nhất của A
c) Tìm các số là số chính phương có trong A
d) Tìm các số là số nguyên tố có trong A
e) Sắp xếp dãy A theo thứ tự tăng dần
f) Tìm những phần tử trong A có giá trị bằng y và xóa chúng khỏi A nếu có
g) Thêm vào dãy A phần tử x sao cho sau khi thêm thì A vẫn tăng dần mà không cần phải sắp xếp lại

Bài 2. Viết chương trình nhập dữ liệu cho 2 mảng số nguyên A và B đều có m hàng, n cột (m, n<=40):

a) Xuất mảng A, B và mảng tổng A+B ra màn hình
b) Tìm giá trị lớn nhất của mảng A
c) Tìm giá trị lớn nhất trong tất cả các phần tử của 2 mảng
d) Tìm tất cả các phần tử xuất hiện chung ở cả 2 mảng
e) Tính tổng các phần tử là số lẻ ở trên mảng A
f) Tìm tất cả số chẵn xuất hiện ở A nhưng không xuất hiện ở B.

BÀI TẬP CHƯƠNG 4

Bài 3. Nhập dữ liệu cho chuỗi S bất kì:

a) Xuất chuỗi S theo chiều ngược lại bằng 2 cách.
b) Đếm xem chuỗi S gồm bao nhiêu kí tự là chữ cái, bao nhiêu kí tự là chữ số
c) Chuỗi S có phải là “Ngon ngu lap trinh C++” hay không ?
d) Hãy xóa tất cả các kí tự không phải là chữ cái khỏi S
e) Thay thế các chữ in hoa trong S bằng kí tự dấu *
f) Thêm vào cuối chuỗi S chuỗi “Hello”

Bài 4. Nhập vào 1 mảng với các phần tử là 1 chuỗi lưu trữ tên các thành phố.

a) Xuất tên các thành phố đó ra màn hình
b) Tìm thành phố có tên dài nhất
c) Tìm xem trong mảng có thành phố nào có tên là “Ha Noi” hay không? (không phân biệt chữ in hoa hoặc chữ in thường)
d) Sắp xếp các thành phố theo thứ tự tên gọi từ A->Z

BÀI TẬP CHƯƠNG 4

Bài 5. Nhập vào năm dương lịch, xuất ra năm âm lịch và các năm kỵ, năm hợp tương ứng.

Biết rằng năm âm lịch gồm 12 chi:

(Tý, Sửu, Dần, Mão, Thìn, Tỵ, Ngọ, Mùi, Thân, Dậu, Tuất, Hợi)/

và 10 can:

(Giáp, Ất, Bính, Đinh, Mậu, Kỷ, Canh, Tân, Nhâm, Quý).

Năm kỵ và hợp có chu kỳ lần lượt là 3 và 4 năm.

Ví Dụ:

Năm 0 là Canh Thân, năm 1 là Tân Dậu, năm 2 là Nhâm Tuất..)