Định nghĩa toán tử so sánh xâu ==, cộng xâu +, chỉ số [] trên lớp String. Có các hàm khởi tạo và in xâu trong lớp.

```
#include<iostream>
#include<string.h>
using namespace std;
class String
{
int L; // L<127
char st[127];
public:
String(char *str="")
{
strcpy(st, str); // st = str
L = strlen(st);
}
void Print() const
{
cout<<st<<endl;
}
String operator+(String &Str)
{
char dst[127];
strcpy(dst, st);
strcat(dst, Str.st);
return String(dst);
}
bool operator==(String &Str)
{
if (strcmp(st, Str.st) == 0) // st == str
return true;
else
return false;
}
char& operator[](int i)
{
if(i<0) // error input
return st[i];
if(i>L-1) // error input
return st[L-1];
return st[i];
}
};
int main()
{
```

```
char st[127];
cout<<"Nhap String 1: "; cin.getline(st, 30);
String Str1(st);
cout<<"String 1: "; Str1.Print();
int i;
cout<<"Vi tri thu i : "; cin>>i;
cout<<"Ky tu tai vi tri "<<i+1<<" trong String 1: "<<Str1[i]<<endl;
fflush(stdin); // xoa bo dem ban phim
cout<<"Nhap String 2: "; cin.getline(st, 30);
String Str2(st);
bool r = (Str1 == Str2);
if(r == true)
cout<<"String 1 == String2"<<endl;
else
cout<<"String 1 # String2"<<endl;
String Str;
Str = Str1 + Str2;
cout<<"String 1 + String 2 = "; Str.Print();
return 0;
}
```

Định nghĩa toán tử >>, <<, ++, -- trên lớp số phức.
Tăng/giảm trước: chức năng không đối số
Tăng/giảm sau: chức năng có đối số giả

```
#include<iostream>
#include<math.h>
class Complex
{
float real, image;
friend istream& operator>>(istream&is, Complex&c);
friend ostream& operator<<(ostream&os, Complex&c);
public:
Complex(float r=0.0, float i=0.0);
Complex& operator++(); // tang truoc
Complex operator--(int arg); //giam sau
};
istream& operator>>(istream&is, Complex&c)
{
cout<<"real: "; is>>c.real;
cout<<"image: "; is>>c.image;
return is;
}
ostream& operator<<(ostream&os, Complex&c)
{
```

```
os<<c.real<<((c.image>=0)? " +j": " -j")<<fabs(c.image)<<endl;
return os;
}
Complex::Complex(float r, float i)
{
real = r; image = i;
}
Complex& Complex::operator++()
{
real = real + 1;
image = image + 1;
return *this;
}
Complex Complex::operator--(int arg)
{
Complex tmp;
tmp = *this;
real = real - 1;
image = image - 1;
return tmp;
}
int main()
{
float real, image;
Complex p;
cout<<"Nhap so phuc p: "<<endl; cin>>p;
cout<<"So phuc p: "<<p;
Complex q;
q = ++p;
cout<<"So phuc p: "<<p;
cout<<"So phuc q: "<<q;
Complex r;
r = q--;
cout<<"So phuc q: "<<q;
cout<<"So phuc r: "<<r;
return 0;
}
```

Khai báo lớp học viên có các thuộc tính (họ tên, mã sinh viên, ngành học, điểm tổng kết), chức năng nhập dữ liệu cho các đối tượng, định nghĩa hàm bạn tìm mã học viên có điểm tổng kết cao nhất.

```
#include<iostream>
#include<string.h>
using namespace std;
class Student
```

```
{
char hoten[30];
char mahv[10];
char nganh[10];
float dht;
public:
void Init()
{
cout<<"Ho ten: "; cin.getline(hoten, 30);
cout<<"Ma hoc vien: "; cin.getline(mahv, 10);
cout<<"Nganh hoc: "; cin.getline(nganh, 10);
fflush(stdin);
cout<<"Diem hoc tap: "; cin>>dht;
}
void Print() const
{
cout<<"Ho ten: "<<hoten<<"\tMa hoc vien: "<<mahv<<"\tNganh hoc: "<<nganh
<<"\tDiem hoc tap: "<<dht<<endl;
}
friend char* Search(Student* hv, int n);
};
char* Search(Student* hv, int n)
{
int i,k;
float max = hv[0].dht;
k = 0;
char tmp[10];
for(i=1; i<n; i++)
if (max < hv[i].dht)
{
max = hv[i].dht;
k = i;
}
return hv[k].mahv;
}
int main()
{
Student hv[10];
int i, n;
cout<<"So luong hoc vien: "; cin>>n; // n<10
cout<<"Nhap thong tin hoc vien: "<<endl;
for(i=0; i<n; i++)
{
fflush(stdin);
hv[i].Init();
}
cout<<"Thong tin hoc vien: "<<endl;
for(i=0; i<n; i++)
hv[i].Print();
cout<<"Ma so hoc vien co diem hoc tap tot nhat: "<<Search(hv, n)<<endl;
return 0;
}
```

Khai báo và định nghĩa lớp bạn

```
#include <iostream>
#include <string.h>
using namespace std;
const int sz = 20;
class Holder
{
private:
int a[sz];
public:
void initialize();
friend class Pointer;
};
class Pointer {
private:
Holder* h;
int* p;
public:
void initialize(Holder* h);
// Move around in the array:
void next();
void previous();
void top();
void end();
// Access values:
int read();
void set(int i);
};
void Holder::initialize() {
memset(a, 0, sz * sizeof(int));
}
void Pointer::initialize(Holder* rv) {
h = rv;
p = rv->a;
}
void Pointer::next() {
if(p < &(h->a[sz - 1])) p++;
}
void Pointer::previous() {
if(p > &(h->a[0])) p--;
}
void Pointer::top() {
p = &(h->a[0]);
}
```

```
void Pointer::end() {
p = &(h->a[sz - 1]);
}
int Pointer::read() {
return *p;
}
void Pointer::set(int i) {
*p = i;
}
int main()
{
Holder h;
Pointer hp, hp2;
int i;
h.initialize();
hp.initialize(&h);
hp2.initialize(&h);
for(i = 0; i < sz; i++)
{
hp.set(i); hp.next();
}
hp.top();
hp2.end();
for(i = 0; i < sz; i++)
{
cout << "hp = " << hp.read()<< ", hp2 = " << hp2.read() << endl;
hp.next();
hp2.previous();
}
return 0;
}
```