**1.**

#include <iostream>

#include <cstring>

#pragma warning(disable:4996)

using namespace std;

class String

{

public:

String(const char\* src\_str = NULL);

String(const String& src\_str);

void Set(const char\* new\_str);

void Set(const String& new\_str);

void Print() const { cout << str; }

String operator+(const String& addend) const;

String& operator=(const String& new\_str);

~String() { delete[] str; }

private:

char\* str;

};

String::String(const char\* src\_str)

{

if (src\_str == NULL)

{

str = new char[1];

\*str = '\0';

}

else

{

str = new char[strlen(src\_str) + 1];

strcpy(str, src\_str);

}

}

String::String(const String& src\_str)

{

if (&src\_str == this)

{

str = new char[1];

\*str = '\0';

return;

}

str = new char[strlen(src\_str.str) + 1];

strcpy(str, src\_str.str);

}

void String::Set(const char\* new\_str)

{

delete[] str;

if (new\_str == NULL)

{

str = new char[1];

\*str = '\0';

}

else

{

str = new char[strlen(new\_str) + 1];

strcpy(str, new\_str);

}

}

void String::Set(const String& new\_str)

{

if (&new\_str == this) return;

delete[] str;

str = new char[strlen(new\_str.str) + 1];

strcpy(str, new\_str.str);

}

String String::operator+(const String& addend) const

{

char\* res = new char[strlen(str) + strlen(addend.str) + 1];

strcpy(res, str);

strcat(res, addend.str);

String \_res(res);

delete[] res;

return \_res;

}

String& String::operator=(const String& new\_str)

{

if (&new\_str == this) return \*this;

delete[] str;

str = new char[strlen(new\_str.str) + 1];

strcpy(str, new\_str.str);

return \*this;

}

int main()

{

String s1("abcd"), s2("efghij"), s3("lmnop"), s4, s5;

s4 = s1 + s2 + s3; s4.Print();

cout << endl;

s5 = s5 + s3 + s4; s5.Print();

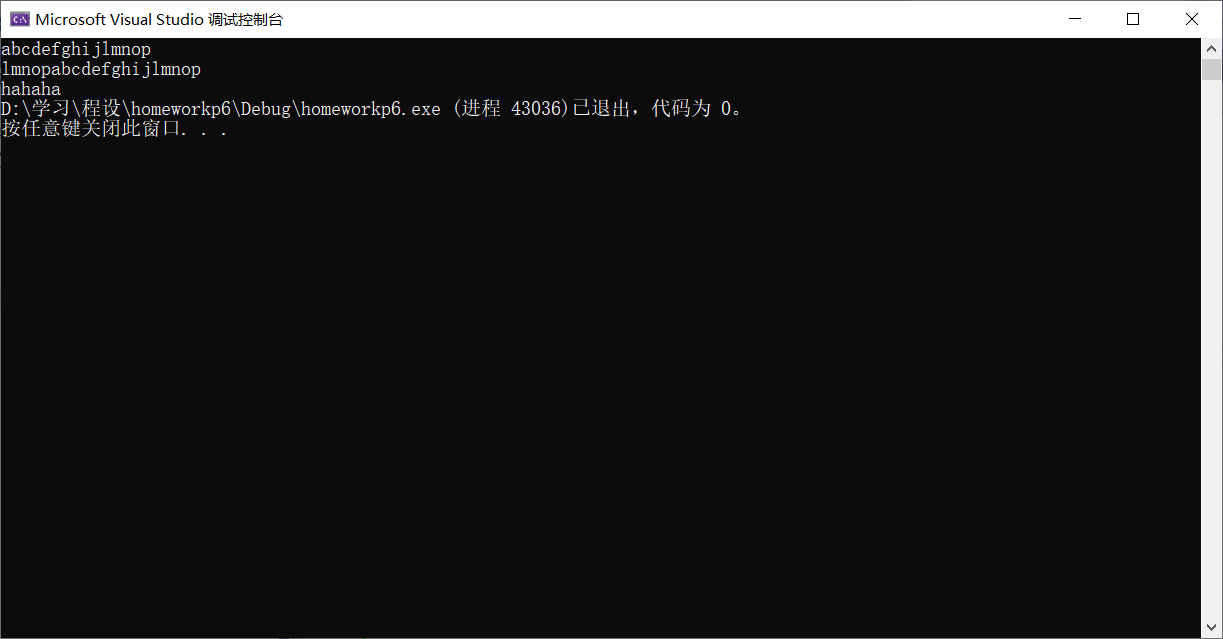
cout << endl;

s4.Set("hahaha");

s4.Print();

return 0;

}



**2.**

#include <iostream>

#include <cstring>

#pragma warning(disable:4996)

using namespace std;

class String

{

public:

String(const char\* src\_str = NULL);

String(const String& src\_str);

void Set(const char\* new\_str);

void Set(const String& new\_str);

void Print() const { cout << str; }

String operator+(const String& addend) const;

String& operator=(const String& new\_str);

String& operator+=(const String& addend);

~String() { delete[] str; }

private:

char\* str;

};

String::String(const char\* src\_str)

{

if (src\_str == NULL)

{

str = new char[1];

\*str = '\0';

}

else

{

str = new char[strlen(src\_str) + 1];

strcpy(str, src\_str);

}

}

String::String(const String& src\_str)

{

if (&src\_str == this)

{

str = new char[1];

\*str = '\0';

return;

}

str = new char[strlen(src\_str.str) + 1];

strcpy(str, src\_str.str);

}

void String::Set(const char\* new\_str)

{

delete[] str;

if (new\_str == NULL)

{

str = new char[1];

\*str = '\0';

}

else

{

str = new char[strlen(new\_str) + 1];

strcpy(str, new\_str);

}

}

void String::Set(const String& new\_str)

{

if (&new\_str == this) return;

delete[] str;

str = new char[strlen(new\_str.str) + 1];

strcpy(str, new\_str.str);

}

String String::operator+(const String& addend) const

{

char\* res = new char[strlen(str) + strlen(addend.str) + 1];

strcpy(res, str);

strcat(res, addend.str);

String \_res(res);

delete[] res;

return \_res;

}

String& String::operator=(const String& new\_str)

{

if (&new\_str == this) return \*this;

delete[] str;

str = new char[strlen(new\_str.str) + 1];

strcpy(str, new\_str.str);

return \*this;

}

String& String::operator+=(const String& addend)

{

char\* res = new char[strlen(str) + strlen(addend.str) + 1];

strcpy(res, str);

strcat(res, addend.str);

delete[] str;

str = res;

return \*this;

}

int main()

{

String s1("abcd"), s2("efghij"), s3("lmnop");

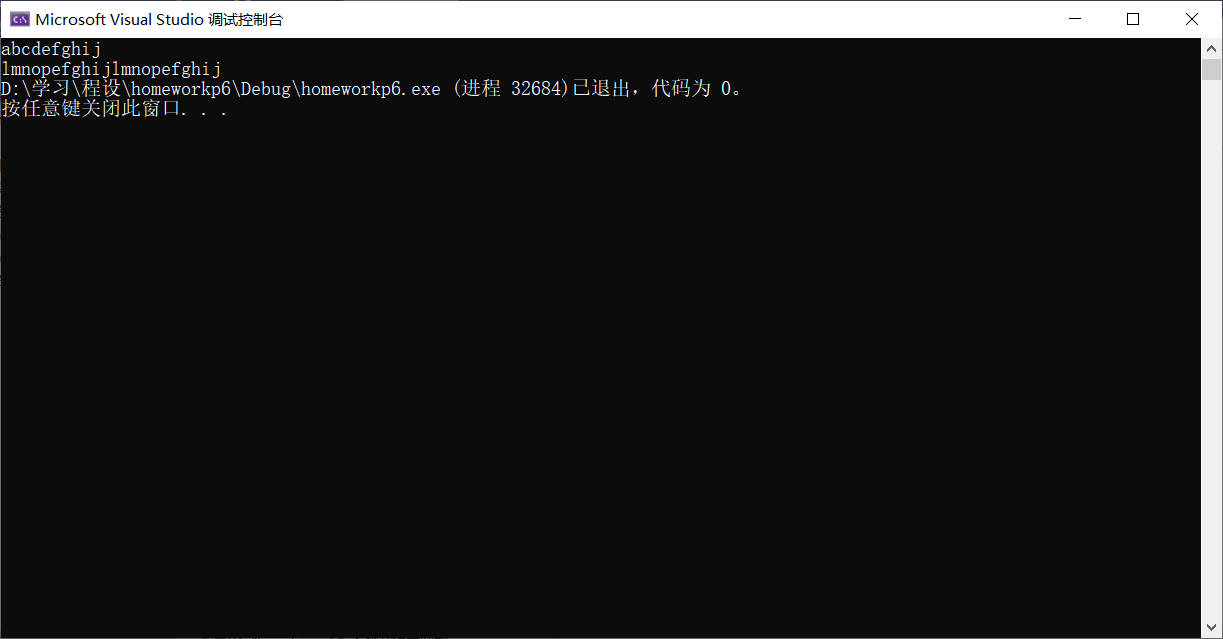
s1 += s2; s1.Print();

cout << endl;

s3 += s3 += s2; s3.Print();

return 0;

}



**3\*.**

#include <iostream>

#include <cstring>

#pragma warning(disable:4996)

using namespace std;

class String

{

public:

String(const char\* src\_str = NULL);

String(const String& src\_str);

void Set(const char\* new\_str);

void Set(const String& new\_str);

void Print() const { cout << str; }

friend String operator+(const String& \_Left, const String& \_Right);

String& operator=(const String& new\_str);

~String() { delete[] str; }

private:

char\* str;

};

String::String(const char\* src\_str)

{

if (src\_str == NULL)

{

str = new char[1];

\*str = '\0';

}

else

{

str = new char[strlen(src\_str) + 1];

strcpy(str, src\_str);

}

}

String::String(const String& src\_str)

{

if (&src\_str == this)

{

str = new char[1];

\*str = '\0';

return;

}

str = new char[strlen(src\_str.str) + 1];

strcpy(str, src\_str.str);

}

void String::Set(const char\* new\_str)

{

delete[] str;

if (new\_str == NULL)

{

str = new char[1];

\*str = '\0';

}

else

{

str = new char[strlen(new\_str) + 1];

strcpy(str, new\_str);

}

}

void String::Set(const String& new\_str)

{

if (&new\_str == this) return;

delete[] str;

str = new char[strlen(new\_str.str) + 1];

strcpy(str, new\_str.str);

}

String operator+(const String& \_Left, const String& \_Right)

{

char\* res = new char[strlen(\_Left.str) + strlen(\_Right.str) + 1];

strcpy(res, \_Left.str);

strcat(res, \_Right.str);

String \_res(res);

delete[] res;

return \_res;

}

String& String::operator=(const String& new\_str)

{

if (&new\_str == this) return \*this;

delete[] str;

str = new char[strlen(new\_str.str) + 1];

strcpy(str, new\_str.str);

return \*this;

}

int main()

{

String s1("abcd"), s2("efghij"), s3("lmnop"), s4, s5;

s4 = s1 + s2 + s3; s4.Print();

cout << endl;

s5 = s5 + s3 + s4; s5.Print();

cout << endl;

s4.Set("hahaha");

s4.Print();

return 0;

}

