**题四：**

#include<iostream>

using namespace std;

template<class T> class Chain;

template<class T>

class ChainNode {

friend class Chain<T>;

public:

T data;

ChainNode<T>\* link;

};

template<class T>

class Chain {

public:

Chain(){first = 0;}

ChainNode<T>\* getHead(){return first;}

bool Insert(int k,const T&x){

ChainNode<T>\* p = first;

if (k < 0)

return false;

for (int i = 1; i < k && p != NULL; i++)

p = p->link;

if (k > 0 && p == NULL)

return false;

ChainNode<T>\* newNode = new ChainNode<T>;

newNode->data = x;

if (k) {

newNode->link = p->link;

p->link = newNode;

}

else {

newNode->link = first;

first = newNode;

}

return true;

}

private:

ChainNode<T>\* first;

};

template<class T>

class ChainIterator {

public:

ChainIterator(ChainNode<T>\*startNode=0){

current = startNode;

}

ChainNode <T>\* getCurrent()const { return current; }

int getData()const { return current->data; }

ChainIterator operator++(int){//后置递增

ChainIterator old = \*this;

current = current->link;

return old;

}

private:

ChainNode <T>\* current;

};

int main()

{

Chain<int>list;

int i, n, x, result = 0;

cout << "输入n的值(n需大于5):" << endl; cin >> n;

while (n <= 5){

cout << "n需大于5\n请重新输入n的值" << endl;

cin >> n;

}

cout << "输入x1, x2, ..., xn:" << endl;

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

cin >> x;

list.Insert(i, x);

}

ChainIterator<int>a(list.getHead());ChainIterator<int>b(list.getHead());

for (i = 0; i < 5; i++)

b++;

while (a.getCurrent() != NULL && b.getCurrent() != NULL)

result += a++.getData() \* b++.getData();

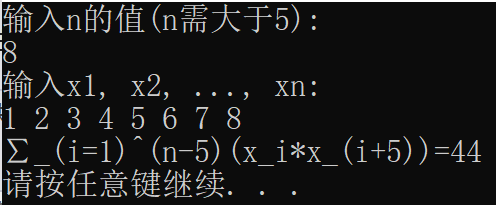
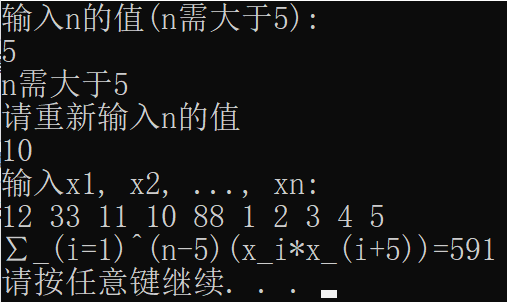
cout << "∑\_(i=1)^(n-5)(x\_i\*x\_(i+5))="<<result << endl;

system("pause");

return 0;

}

**测试结果：**



**12\*1+33\*2+11\*3+10\*4+88\*5=591； 1\*6+2\*7+3\*8=44；**

**题五：**

#include<iostream>

using namespace std;

template<class T> class Chain;

template<class T>

class ChainNode {

friend class Chain<T>;

public:

T data;

ChainNode<T>\* link;

};

template<class T>

class Chain {

public:

Chain() { first = last = 0; }//constructor

~Chain()//destructor

{

ChainNode<T>\* p;

while (first)

{

p = first;

first = first->link;

delete p;

}

}

void Output()

{

ChainNode<T>\* current;

for (current = first; current; current = current->link)

{

cout<< current->data;

if (!current->link)

cout << "" << endl;

else

cout << ",";

}

}

ChainNode<T>\* getHead()

{

return first;

}

ChainNode<T>\* getTail()

{

return last;

}

T Front()

{

return first->data;

}

T Back()

{

return last->data;

}

int Length()

{

ChainNode<T>\* current = first;

int length = 0;

while (current)

{

length++;

current = current->link;

}

return length;

}

bool IsEmpty()

{

return first == last;

}

void EnHead(T x) //在表头增加x

{

ChainNode<T>\* p = new ChainNode<T>;

p->data = x;

p->link = first;

first = p;

}

void EnTail(T x) //在表尾增加x

{

ChainNode<T>\* p = new ChainNode<T>;

p->data = x;

p->link = NULL;

last->link = p;

last = p;

}

bool DeHead(const T& x) //在表头删去一个元素

{

if (IsEmpty())return false;

else {

ChainNode<T>\* p;

p = first;

x = p->data;

first = first->link;

delete p;

return true;

}

}

bool DeTail(const T& x) //在表尾删去一个元素

{

if (IsEmpty())return false;

else {

ChainNode<T>\* p;

ChainNode<T>\* current = first;

int k = 1;

while (k < Length() - 1)

{

current = current->link;

k++;

}

p = last;

x = p->data;

current->link = p->link;

delete p;

return true;

}

}

bool Insert(int k,const T&x) //在第k个位置插入元素x（索引从0开始）

{

ChainNode<T>\* p = first;

if (k < 0) return false;

for (int i = 1; i < k && p != NULL; i++)

p = p->link;

if (k > 0 && p == NULL) return false;

ChainNode<T>\* newNode = new ChainNode<T>;

newNode->data = x;

if (k) {

newNode->link = p->link;

p->link = newNode;

}

else {

newNode->link = first;

first = newNode;

}

if(!newNode->link)

last = newNode;

return true;

}

T Get(int k) //返回表中第i个元素

{

ChainNode<T>\* p = first;

for (int i = 0; i < k - 1; i++)

p = p->link;

return p->data;

}

bool DeInsert(int i, const T& x) //将表中第i个元素修改为x

{

ChainNode<T>\* p;

if (i <= 1) {

p = first;

first = first->link;

}

else {

ChainNode<T>\* current = first;

int k = 1;

while (k < i - 1&&current!=NULL)

{

current = current->link;

k++;

}

if (current == NULL || current->link == NULL)

{

cerr << "无效修改位置" << endl;

return false;

}

p = current->link;

current->link = p->link;

}

delete p;

Insert(i-1, x);

return true;

}

private:

ChainNode<T>\* first;

ChainNode<T>\* last;

};

template<class T>

class ChainIterator {

public:

ChainIterator(ChainNode<T>\*startNode=0)

{

current = startNode;

}

ChainNode <T>\* getCurrent()const { return current; }

int getData()const { return current->data; }

ChainIterator operator++(int)//后置递增

{

ChainIterator old = \*this;

current = current->link;

return old;

}

ChainIterator& operator++()//前置递增

{

current = current->link; return \*this;

}

private:

ChainNode <T>\* current;

};

int main()

{

Chain<int>list;

int i, result=0;

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

list.Insert(i, i+1);

cout << "List: "; list.Output();

cout << "Length = " << list.Length() << endl;

cout << "The first element of the list is: " << list.Front() << endl;

cout << "The last element of the list is: " << list.Back() << endl;

list.EnHead(0);

cout << "After insert 0 at the front of the list, List is: "; list.Output();

list.EnTail(11);

cout << "After insert 11 at the back of the list, List is: "; list.Output();

cout << "Now the 6th element of the list is: " << list.Get(6) << endl;

list.DeInsert(5, 88);

cout << "After insert 88 as the 5th element of the list and delete the 5th element, List is: "; list.Output();

ChainIterator<int>c(list.getHead());

while (c.getCurrent() != NULL)//use a forward iterator

result += c++.getData();

cout << "Now the sum of the elements of the list is:" << result << endl;

system("pause");

return 0;

}

**测试结果：**

