**第四次作业**

**要求：**

编程实现有序单链表的插入算法，要求实例测试，是在链表表头和表中间插入的测试。

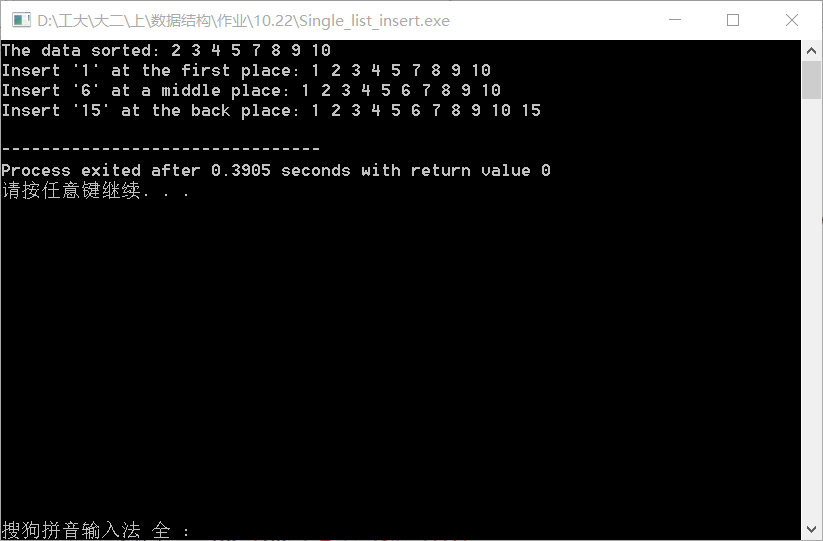
**分析：**

当原链表为空链表时，此时插入的数据即为最小值，将其作为头结点，next指向NULL。

当插入的数据小于链表开始的最小值时，只要将这个数据的结点作为头结点，并且next指针指向原先的头结点。

当插入的数据在中间值时，需要用到后继指针，找到第一个比输入数据大的数据，在其之前的数据就是后继指针指向的数据，将输入数据的next指向原后继指针指向的next，然后将后继指针的next指向输入数据的结点。即将数据插入到了链表中。

**运行截图：**



**源代码：**

//Single\_linked\_list inserting project\_Task5.

//Created by Yule, Oct\_23\_2015

//==============================================================================

#include<iostream>

using namespace std;

template <typename T>

class node

{

public:

T nodeValue; // data held by the node

node<T> \*next; // next node in the list

node() : next(NULL) {}

node(const T& item, node<T> \*nextNode = NULL) : nodeValue(item), next(nextNode) {}

};

// Create data by manual inputting.

node<int>\* create()

{

int temp[8] = {2, 3, 4, 5, 7, 8, 9, 10};

node<int> \*first = NULL, \*p[2];

// return first;

// using the subscript complementation way to avoid

// transmitting points addresses and accelerate the

// speed.

first = new node<int>(temp[0], NULL);

p[0] = first;

for (int i = 1; i<8; i++){

p[i%2] = new node<int>(temp[i], NULL);

p[(i-1)%2]->next = p[i%2];

}

return first;

}

// Output the list with given head pointer.

void print(node<int> \*first)

{

node<int> \*ser;

ser = first;

while(ser!=NULL){

cout<<ser->nodeValue<<' ';

ser=ser->next;

}

cout<<endl;

}

// Order inserting function

node<int>\* order\_insert(int num, node<int> \*first)

{

node<int> \*ser = NULL, \*sub = NULL, \*temp = NULL;

sub = ser = first;

// If there is no node in the original list, or the number

// is smaller than the first value.Create a new node point to

// the first, and let this new node be the head pointer.

if (ser == NULL || (first->nodeValue > num)){

sub = new node<int>(num, ser);

return sub;

}

// If the number is in a middle of the list, use subscript pointer

// to find the insert node.

while(ser!=NULL){

if (ser->nodeValue > num){

sub->next = new node<int>(num, sub->next);

return first;

}

sub = ser;

ser = ser->next;

}

// if the number is lager than any one in the list, let this number

// be the last node.

sub->next = new node<int>(num, NULL);

return first;

}

int main()

{

// Declarations of required function.

node<int>\* create(); // can be changed into manual input when needed.

void print(node<int> \*);

node<int>\* order\_insert(int, node<int>\*); //Function of single linked list insertion.

// data input section.

node<int> \*first = NULL;

first = create();

cout<<"The data sorted: ";

print(first);

// data insert section in order.

cout<<"Insert '1' at the first place: ";

first = order\_insert(1, first);

print(first);

cout<<"Insert '6' at a middle place: ";

first = order\_insert(6, first);

print(first);

cout<<"Insert '15' at the back place: ";

first = order\_insert(15, first);

print(first);

return 0;

}