**一.插入排序**

#include <iostream>

using namespace std;

void swap(int zym[],int i,int j)

{

int tmp=zym[i];

zym[i]=zym[j];

zym[j]=tmp;

}

void InsertSort(int zym[],int n)

{

for(int i=1;i<n;i++)

{

for(int j=i;j>0;j--)

{

if(zym[j]>zym[j-1])

swap(zym,j,j-1);

else

break;

}

}

}

int main()

{

int zym[]={3,1,2,5,4,8,9,12,6};

InsertSort(zym,9);

for(int i=0;i<9;i++)

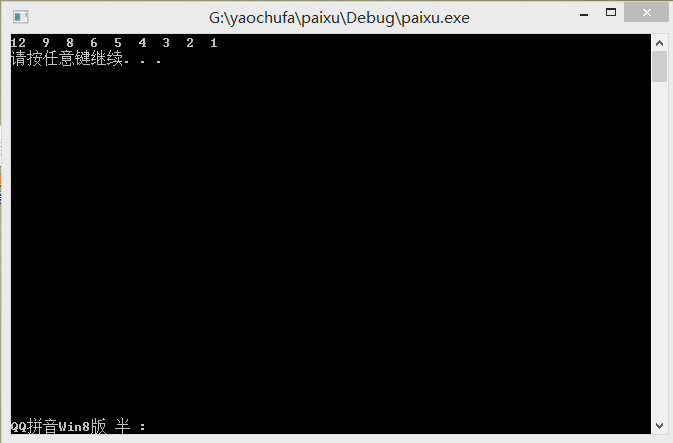
cout<<zym[i]<<" ";

cout<<endl;

system("pause");

return 0;

}



1. **冒泡排序**

#include <iostream>

using namespace std;

void swap(int zym[],int i,int j)

{

int tmp=zym[i];

zym[i]=zym[j];

zym[j]=tmp;

}

void BubbleSort(int zym[],int n)

{

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

{

for(int j=n-1;j>i;j--)

{

if(zym[j]<zym[j-1])

swap(zym,j,j-1);

}

}

}

int main()

{

int zym[7]={3,1,2,5,4,16,9};

BubbleSort(zym,7);

for(int i=0;i<7;i++)

cout<<zym[i]<<" ";

cout<<endl;

system("pause");

return 0;

}



**三.选择排序**

#include <iostream>

using namespace std;

void swap(int zym[],int i,int j)

{

int tmp=zym[i];

zym[i]=zym[j];

zym[j]=tmp;

}

void SelectionSort(int zym[],int n)

{

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

{

int smallest=i;

for(int j=i+1;j<n;j++)

{

if(zym[smallest]>zym[j])

smallest=j;

}

swap(zym,i,smallest);

}

}

int main()

{

int zym[6]={3,1,2,5,4,6};

SelectionSort(zym,6);

for(int i=0;i<6;i++)

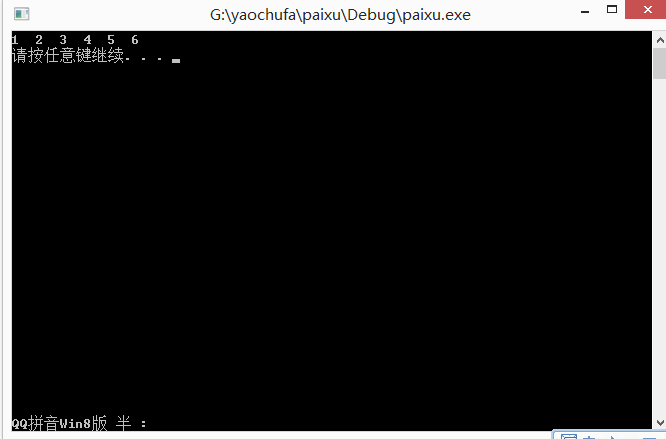
cout<<zym[i]<<" ";

cout<<endl;

system("pause");

return 0;

}



‘

**四.快速排序**

#include <iostream>

using namespace std;

void swap(int zym[],int i,int j)

{

int tmp=zym[i];

zym[i]=zym[j];

zym[j]=tmp;

}

int partition(int zym[],int left,int right)

{

int mid=(left+right)/2;

int tmp=zym[mid];

swap(zym,mid,right);

int i=left;

int j=right;

while(1)

{

while(1)

{

if(i==j)

{

zym[i]=tmp;

return i;

}

if(zym[i]>tmp)

{

zym[j]=zym[i];

j--;

break;

}

i++;

}

while(1)

{

if(i==j)

{

zym[j]=tmp;

return j;

}

if(zym[j]<tmp)

{

zym[i]=zym[j];

i++;

break;

}

j--;

}

}

}

void quickSort(int zym[],int left,int right)

{

if(right<=left)

return;

int pivot=partition(zym,left,right);

quickSort(zym,left,pivot-1);

quickSort(zym,pivot+1,right);

}

int main()

{

int zym[8]={6,8,7,3,1,2,5,4};

quickSort(zym,0,7);

for(int i=0;i<8;i++)

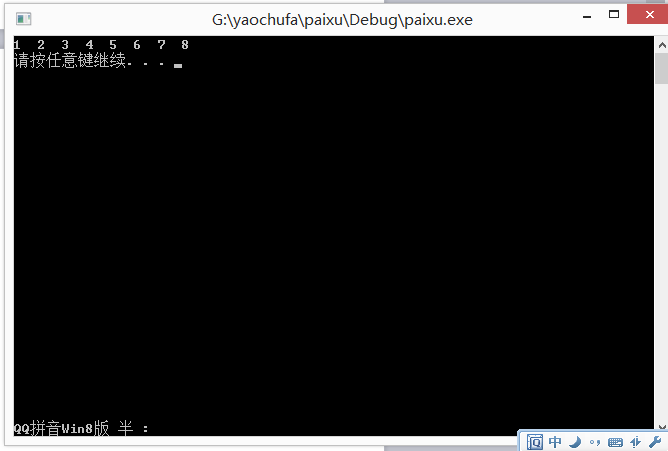
cout<<zym[i]<<" ";

cout<<endl;

system("pause");

return 0;

}



**五.归并排序**

#include <iostream>

using namespace std;

int KeySize(int zym[],int size)

{

int key\_size=1;

for(int i=0;i<size;i++)

{

int temp=1;

int n=10;

while(zym[i]/n>0)

{

temp++;

n\*=10;

}

key\_size=(temp>key\_size)?temp:key\_size;

}

return key\_size;

}

void RadixSort(int zym[],int size)

{

int bucket[10][10]={0};

int order[10]={0};

int key\_size=KeySize(zym,size);

for(int n=1;key\_size>0;n\*=10,key\_size--)

{

for(int i=0;i<size;i++)

{

int lsd=(zym[i]/n)%10;

bucket[lsd][order[lsd]]=zym[i];

order[lsd]++;

}

int k=0;

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

{

if(order[i]!=0)

{

for(int j=0;j<order[i];j++)

{

zym[k]=bucket[i][j];

k++;

}

order[i]=0;

}

}

}

}

int main()

{

int zym[10]={1,4,5,7,8,9,2,3,6,10};

int size=sizeof(zym)/sizeof(int);

RadixSort(zym,size);

for(int i=0;i<size;i++)

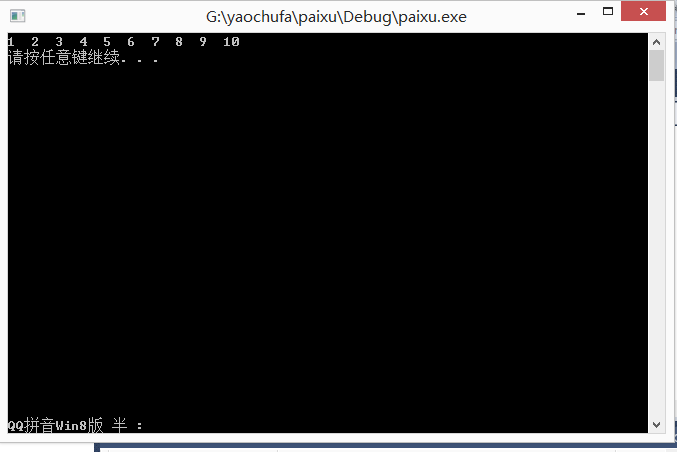
cout<<zym[i]<<" ";

cout<<endl;

system("pause");

return 0;

}



## ****适配器模式****

#include<iostream>

using namespace std;

class Target //目标

{

public:

virtual void Request(){};

};

class Adaptee //源

{

public:

void SpecificRequest()

{

cout<<"yao chu fa"<<endl;

}

};

class Adapter : public Adaptee, public Target //适配器

{

public:

void Request()

{

this->SpecificRequest();

}

};

int main()

{

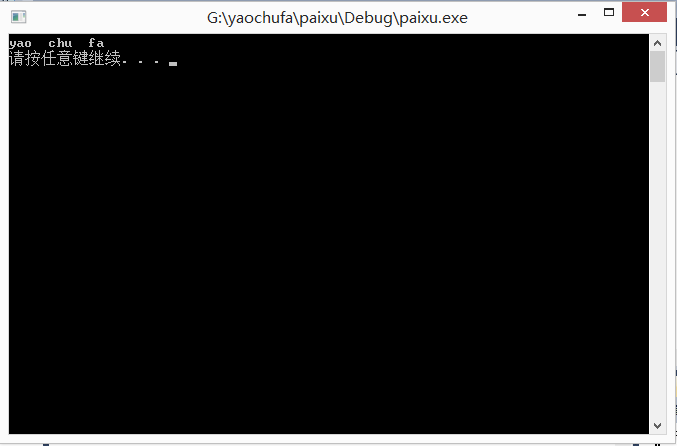
Target \*t = new Adapter();

t->Request();

system("pause");

return 0;

}



**观察者模式**

#include <iostream>

#include <vector>

#include <string>

using namespace std;

class teacher;

class student

{

public:

student(string strName, teacher\* strSub)

{

name = strName;

sub = strSub;

}

void Update();

private:

string name;

teacher\* sub;

};

class teacher

{

public:

string action;

void Add(student ob) { observers.push\_back(ob); }

void Remove(int addIndex)

{

if(addIndex >=0 && addIndex < observers.size())

observers.erase(observers.begin() + addIndex);

}

void Notify()

{

vector<student>::iterator it;

for (it=observers.begin(); it!=observers.end(); ++it)

{

(\*it).Update();

}

}

private:

vector<student> observers;

};

void student::Update()

{

cout << name << " : " << sub->action << ", begin to study" << endl;

}

int main()

{

teacher\* p = new teacher();

student\* s1 = new student("teacher", p);

student\* s2 = new student("student", p);

p->Add(\*s1);

p->Add(\*s2);

p->action = "The Principal is coming...";

p->Notify();

p->Remove(0);

p->Notify();

system("pause");

return 0;

}

