Program 1

Calculate the sum and the average price of a kind of goods.

Program code

#include<iostream>

using namespace std;

class Sale

{

private:

int num;

int quantity;

double price;

public:

Sale(){};

Sale(int pn, int pq,double pp)

{

n=n+pq;

if(pq>10)

sum=sum+pq\*pp\*discount;

else

sum=sum+pq\*pp;

}

static double discount;

static double sum;

static int n;

static double average();

};

double Sale::discount=0.98;

double Sale::sum=0;

int Sale::n=0;

double Sale::average()

{

return sum\*1.0/n;

}

int main()

{

Sale s1(101,5,23.50);

Sale s2(102,12,24.56);

Sale s3(103,100,21.50);

cout<<s3.sum<<endl;

cout<<s3.average()<<endl;

}

Program analysis

1. Static members are shared by all the objects.
2. The static members should be initialized.

Program result

Program 2

Define a class Manager and Employee. Let Manager to be the friend class of Employee.

Program code

#include<iostream>

using namespace std;

class Manager

{

private:

int id;

char \*name;

int age;

friend class Employee;

public:

Manager(){}

Manager(int pid, char\* pn,int pa)

{

id=pid;

name= new char[strlen(pn)+1];

if(name!=0)

strcpy(name,pn);

age=pa;

}

~Manager()

{

delete[] name;

}

};

class Employee

{

private:

int id;

char \*name;

int age;

public:

Employee(){}

Employee(int pid, char\* pn,int pa)

{

id=pid;

name= new char[strlen(pn)+1];

if(name!=0)

strcpy(name,pn);

age=pa;

}

~Employee()

{

delete[] name;

}

void printManager(Manager& m1)

{

cout<<"id "<<m1.id<<endl;

cout<<"name "<<m1.name<<endl;

cout<<"age "<<m1.age<<endl;

}

};

int main()

{

Manager m1(1001,"张三",38);

Employee e1(1002,"李四",25);

e1.printManager(m1);

return 0;

}

Program analysis

1.since Manager is the friend class of Engineer, so member function in Manager can access the member in the Engineer

Program result

Program 3

Based on the thoughts of combining to create a class Computer by Chip, HardDisk, Monitor and Keyboard.

Program code

#include<iostream>

using namespace std;

class Chip

{

private:

int id;

char \*name;

public:

Chip(){cout<<"Chip constructed"<<endl;}

Chip(int pid, char\* pn)

{

id=pid;

name= new char[strlen(pn)+1];

if(name!=0)

strcpy(name,pn);

cout<<"Chip constructed"<<endl;

}

void setChip(int pid, char\* pn)

{

id=pid;

name= new char[strlen(pn)+1];

if(name!=0)

strcpy(name,pn);

}

~Chip()

{

delete[] name;

cout<<"Chip deconstructed"<<endl;

}

};

class HardDisk

{

private:

int id;

char \*name;

public:

HardDisk(){cout<<"HardDisk constructed"<<endl;}

HardDisk(int pid, char\* pn)

{

id=pid;

name= new char[strlen(pn)+1];

if(name!=0)

strcpy(name,pn);

cout<<"HardDisk constructed"<<endl;

}

void setHardDisk(int pid, char\* pn)

{

id=pid;

name= new char[strlen(pn)+1];

if(name!=0)

strcpy(name,pn);

}

~HardDisk()

{

delete[] name;

cout<<"HardDisk deconstructed"<<endl;

}

};

class Monitor

{

private:

int id;

char \*name;

public:

Monitor(){cout<<"Monitor constructed"<<endl;}

Monitor(int pid, char\* pn)

{

id=pid;

name= new char[strlen(pn)+1];

if(name!=0)

strcpy(name,pn);

cout<<"Monitor constructed"<<endl;

}

void setMonitor(int pid, char\* pn)

{

id=pid;

name= new char[strlen(pn)+1];

if(name!=0)

strcpy(name,pn);

}

~Monitor()

{

delete[] name;

cout<<"Monitor deconstructed"<<endl;

}

};

class Keyboard

{

private:

int id;

char \*name;

public:

Keyboard(){cout<<"Keyboard constructed"<<endl;}

Keyboard(int pid, char\* pn)

{

id=pid;

name= new char[strlen(pn)+1];

if(name!=0)

strcpy(name,pn);

cout<<"Keyboard constructed"<<endl;

}

void setKeyboard(int pid, char\* pn)

{

id=pid;

name= new char[strlen(pn)+1];

if(name!=0)

strcpy(name,pn);

}

~Keyboard()

{

delete[] name;

cout<<"Keyboard deconstructed"<<endl;

}

};

class Computer

{

public:

Chip mc;

HardDisk mh;

Monitor mm;

Keyboard mk;

};

int main()

{

Computer c1;

c1.mh.setHardDisk(1001,"a");

c1.mk.setKeyboard(1002,"b");

c1.mm.setMonitor(1003,"c");

c1.mc.setChip(1004,"d");

return 0;

}

Program analysis

1. Computer is combined by other classes, so other classes are members in Computer.
2. the member classes are constructed in the order of they are defined. And the deconstructed order is opposite.

Program result

Program 4

Using template to find the maximum among 4 numbers

Program code

#include<iostream>

using namespace std;

template <class T>

class Max

{

public:

Max(T pfirst,T psecond,T pthird, T pfourth);

T getMax();

private:

T first;

T second;

T third;

T fourth;

};

template <class T>

Max<T>::Max(T pfirst,T psecond,T pthird,T pfourth):

first(pfirst),second(psecond),third(pthird),fourth(pfourth)

{}

template <class T>

T Max<T>::getMax()

{

T value;

value=first>second?first:second;

value=value>third?value:third;

value=value>fourth?value:fourth;

return value;

};

int main()

{

Max<int> nMax(15,27,59,63);

cout<<"nMax is : "<<nMax.getMax()<<endl;

Max<float> fMax(1.1f,2.2f,2.1f,3.2f);

cout<<"fMax is : "<<fMax.getMax()<<endl;

Max<char> cMax('a','f','e','g');

cout<<"cMax is : "<<cMax.getMax()<<endl;

Max<double> dMax(2.6,3.6,5.95,4.596);

cout<<"dMax is : "<<dMax.getMax()<<endl;

return 0;

}

Program analysis

Class T can be changed into int, double ,char.

And the class Max rank the order.

Program result