1. Default access: The default member access for struct is public, while the default member access for class is private. This means that member variables and member functions declared in a struct are accessible externally by default, but not in class.

Inheritance: In struct, the default inheritance method is public inheritance, while in class, the default inheritance method is private inheritance. This means that when inheriting with struct, the public members of the base class remain public in the derived class, while when inheriting with class, the public members of the base class become private in the derived class.

Type inference: After C++11, the type derivation behavior of struct and class is somewhat different. When using the auto keyword for type inference, the derivation result of struct retains the struct keyword, while class does not.

2. A constructor is a special type of member function that is used to initialize the data members of an object when it is created.Destructors are used to free up resources occupied by objects, such as freeing dynamically allocated memory, closing files, disconnecting from a network, and so on.There are default constructor,parameterized,copy,move.When creating an object,Object array initialization,Function parameter passing,Object Returns,Explicitly called.
3. The instantiation of classes is limited
4. Inference b should be initialized,constructor can’t use type,c is a const parameter,it can’t be modified,x(int i) and setA(int i) is private,it can’t be called out of class.
5. (1)a=4
   b=X::X(int,char,float)
   c=32
   a=0
   b=X::X()
   c=10
   a=10
   B=X::X(....)
   C=10
   A=0
   B=X::X(const X&other)
   C=10

(2)5
10

(3)X1 constructor called...
X2 constructor called...
X3 constructor called...
constructor a() called...
constructor a() called...
constructor a() called...
b3 constructor...
B3 destructor...
X0 destructor...
X0 destructor...

X0 destructor...
X3 destructor...
X2 destructor...
X1 destructor...
(4) 1:Ctor(char *)
1:Ctor(char *)
4:Ctor(Ctor&)
5:=(Ctor &)
3:=(Ctor&&)
2:Ctor(Ctor&&)
~Ctor
~Ctor
~Ctor
~Ctor
~Ctor
6.

```
#include<iostream>
using namespace std;

class Salary
{
private:
    double Wage, Subsidy, Rent, WaterFee, ElecFee;
public:
    Salary(double wage, double subsity, double rent, double waterfee, double elctfee)
    {
        Wage = wage; Subsidy = subsity; Rent = rent; WaterFee = waterfee; ElecFee = elctfee;
    }
    Salary():Wage(0),Subsidy(0),Rent(0),WaterFee(0),ElecFee(0){}

    void setwage(double f)
    {
        Wage = f;
    }
    double getwage()
    {
        return Wage;
    }

    void setsubsidy(double f)
    {
        Subsidy = f;
    }
    double getsubsity()
    {
```

```
        return Subsidy;
    }

    void setrent(double f)
    {
        Rent = f;
    }
    double getrent()
    {
        return Subsidy;
    }

    void setwaterfee(double f)
    {
        WaterFee = f;
    }
    double getwaterfee()
    {
        return WaterFee;
    }

    void setelecfee(double f)
    {
        ElecFee = f;
    }
    double getelecfee()
    {
        return ElecFee ;
    }

    double realSalary()
    {
        return Wage + Subsidy - Rent - WaterFee - ElecFee;
    }
};

int main()
{
    Salary test1;
    cout << test1.realSalary()<< endl;
    Salary test2(12000,23000,1300,1100,1100);
    test2.getelecfee();
    test2.setelecfee(3400);
    test2.getelecfee();
```

```
    cout << test2.realSalary() << endl;
}
```

7.

```
#include<iostream>
using namespace std;
#include<string>

static int number = 0;

class Salary
{
private:
    double Wage, Subsidy, Rent, WaterFee, ElecFee;
public:
    Salary(double wage, double subsity, double rent, double waterfee, double elctfee)
    {
        Wage = wage; Subsidy = subsity; Rent = rent; WaterFee = waterfee; ElecFee = elctfee;
    }
    Salary():Wage(0),Subsidy(0),Rent(0),WaterFee(0),ElecFee(0){}

    void setwage(double f)
    {
        Wage = f;
    }
    double getwage()
    {
        return Wage;
    }

    void setsubsidy(double f)
    {
        Subsidy = f;
    }
    double getsubsity()
    {
        return Subsidy;
    }

    void setrent(double f)
    {
        Rent = f;
    }
    double getrent()
    {
        return Subsidy;
```

```
    }

    void setwaterfee(double f)
    {
        WaterFee = f;
    }
    double getwaterfee()
    {
        return WaterFee;
    }

    void setelecfee(double f)
    {
        ElecFee = f;
    }
    double getelecfee()
    {
        return ElecFee ;
    }

    double realSalary()
    {
        return Wage + Subsidy - Rent - WaterFee - ElecFee;
    }
};

class worker
{
private:
    string name,dept;
    int age;
    Salary salary;
public:
    void count()
    {
        number++;
    }
};
int main()
{
    worker lihua;
    worker pp;
    pp.count();
    lihua.count();
```

```
    cout << number << endl;
}
```

8.

```
// list.h
#ifndef LIST_H
#define LIST_H

#include <iostream>

class IntegerList {
private:
    struct Node {
        int data;
        Node* next;
        Node(int val) : data(val), next(nullptr) {}
    };
    Node* head;

public:
    IntegerList();
    ~IntegerList();
    void insert(int value);
    void Delete(int value);
    bool find(int value);
    void empty();
    void print();
};

#endif // LIST_H

// list.cpp
#include "list.h"

IntegerList::IntegerList() : head(nullptr) {}

IntegerList::~IntegerList() {
    clear();
}

void IntegerList::insert(int value) {
    Node* newNode = new Node(value);
    if (!head) {
        head = newNode;
    } else {
```

```
            Node* current = head;
            while (current->next) {
                current = current->next;
            }
            current->next = newNode;
        }
}

bool IntegerList::Delete(int value) {
        if (!head) return false;
        if (head->data == value) {
            Node* temp = head;
            head = head->next;
            delete temp;
            return true;
        }
        Node* current = head;
        while (current->next && current->next->data != value) {
            current = current->next;
        }
        if (current->next) {
            Node* temp = current->next;
            current->next = current->next->next;
            delete temp;
            return true;
        }
        return false;
}

bool IntegerList::find(int value) {
        Node* current = head;
        while (current) {
            if (current->data == value) {
                return true;
            }
            current = current->next;
        }
        return false;
}

void IntegerList::empty() {
        Node* current = head;
        while (current) {
            Node* temp = current;
```

```
            current = current->next;
            delete temp;
        }
        head = nullptr;
}

void IntegerList::print() {
        Node* current = head;
        while (current) {
            std::cout << current->data << " ";
            current = current->next;
        }
        std::cout << std::endl;
}            Node* current = head;
            while (current->next) {
                current = current->next;
            }
            current->next = newNode;
        }
}
```

9.

```
#include<iostream>
using namespace std;

class ATM
{
    int name;
    int account[10];
    double balance = 1000;
    int choose;
public:
    void checkid()
    {
        name = 0;
        cout << "please input your account:";
        bool flag = true;
        for (auto a:account)
        {
            cin >> a;
            if (a >10||a<0)
            {
                flag = false;
                break;
            }
            name = name * 10 + a;
```

```
    }
    if (!flag)
    {
        cout << "please input right account!" << endl;
        checkid();
    }
    else
    {
        cout << "welcome consume " << name << endl;
    }
}

void check_blance()
{
    cout << "your balance: "<<balance<<endl;
}

void withdraw()
{
    double wmoney;
    cout << "please input the account you want to withdraw:";
    cin >> wmoney;
    balance -= wmoney;
    cout << "succeed!" << endl;
}

void deposit()
{
    double dmoney;
    cout << "please input the account you want to deposit:";
    cin >> dmoney;
    balance += dmoney;
    cout << "succeed!" << endl;
}

void menu()
{
    cout << "Main Menu" << endl;
    cout << "1.Check balance" << endl;
    cout << "2.Withdraw" << endl;
    cout << "3.deposit" << endl;
    cout << "4.Exit" << endl;

    cin >> choose;
```

```
        switch (choose)
        {
        case 1:
            check_blance();
            break;

        case 2:
            withdraw();
            break;

        case 3:
            deposit();
            break;

        case 4:
            cout << "Thank you,Goodbye " << name << endl;
            break;
        }

        menu();
    }
};

int main()
{
    ATM a;
    a.checkid();
    a.menu();
}
```