

Полиморфизм

Лекция 6

```

class Table
{
    int* size;
    char* color;
    bool InitSize(int);
    bool InitColor(char*);
public:
    Table();
    Table(int, char*);
    Table(int);
    ~Table();
    void Show();
    int CalcVolume();
    void SetSize(int);
    void SetColor(char*);
    void GetSize(int&);
    void GetColor(char*);
};

```

```

class CompTable: public Table
{
    int* h;
    char* material;
public:
    CompTable();
    CompTable(int, char*, int, char*);
    ~CompTable();
    void ShowCT();
    int CalcVolumeCT();
};

```

```
class CompTable: public Table
{
    int* h;
    char* material;
public:
    CompTable();
    CompTable(int, char*, int, char*);
    ~CompTable();
    int CalcVolume();
    void Show();
};
```

Метод `CompTable::CalcVolume`

```
int CompTable::CalcVolume ()  
{  
    return (*h) * (*h) * (*h) + Table::CalcVolume ();  
}
```

Метод CompTable::Show

```
void CompTable::Show()  
{  
    cout<<"CompTable info:\n";  
    cout<<"\t h="<<*h<<endl;  
    cout<<"\t material:"<<material<<endl;  
    Table::Show();  
}
```

Вызов обновленных методов -1

```
cout<<"\t-----\n";
```

```
CompTable* CT;
```

```
CT=new CompTable(10,"black",2,"wood");
```

```
CT->Show();
```

```
cout << "CompTable Volume = "  
      << CT->CalcVolume() << endl;
```

```
delete CT; CT=NULL;
```

```
cout<<"\t-----\n";
```

```
-----  
Table::Table(int size1, char* color1)  
void Table::InitSize(int size1)  
bool Table::InitColor(char* color1)  
CompTable::CompTable(int, char*, int, char*)  
CompTable info:  
    h=2  
    material:wood  
Table info:  
    size: 10  
    color: black  
CompTable Volume = 1008  
CompTable::~~CompTable()  
Prс Table::~~Table()  
-----
```

Вызов обновленных методов -1

```
cout<<"\t-----\n";
```

```
CompTable* CT;
```

```
CT=new CompTable(10,"black",2,"wood");
```

```
CT->Show();
```

```
cout << "CompTable Volume = "  
    << CT->CalcVolume() << endl;
```

```
delete CT; CT=NULL;
```

```
cout<<"\t-----\n";
```

```
-----  
Table::Table(int size1, char* color1)  
void Table::InitSize(int size1)  
bool Table::InitColor(char* color1)  
CompTable::CompTable(int, char*, int, char*)
```

```
CompTable info:
```

```
h=2
```

```
material:wood
```

```
Table info:
```

```
size: 10
```

```
color: black
```

```
CompTable Volume = 1008
```

```
CompTable::~~CompTable()
```

```
Прс Table::~~Table()
```

```
-----
```

Вызов обновленных методов -1

```
cout<<"\t-----\n";
```

```
CompTable* CT;
```

```
CT=new CompTable(10,"black",2,"wood");
```

```
CT->Show();
```

```
cout << "CompTable Volume = "
```

```
<< CT->CalcVolume() << endl;
```

```
delete CT; CT=NULL;
```

```
cout<<"\t-----\n";
```

```
-----  
Table::Table(int size1, char* color1)  
void Table::InitSize(int size1)  
bool Table::InitColor(char* color1)  
CompTable::CompTable(int, char*, int, char*)
```

```
CompTable info:  
    h=2  
    material:wood  
Table info:  
    size: 10  
    color: black
```

```
CompTable Volume = 1008
```

```
CompTable::~~CompTable()
```

```
Prс Table::~~Table()
```

```
-----
```


Вызов обновленных методов -1

```
cout<<"\t-----\n";
```

```
CompTable* CT;
```

```
CT=new CompTable(10,"black",2,"wood");
```

```
CT->Show();
```

```
cout << "CompTable Volume = "  
      << CT->CalcVolume() << endl;
```

```
delete CT; CT=NULL;
```

```
cout<<"\t-----\n";
```

```
-----  
Table::Table(int size1, char* color1)  
void Table::InitSize(int size1)  
bool Table::InitColor(char* color1)  
CompTable::CompTable(int, char*, int, char*)  
CompTable info:  
    h=2  
    material:wood  
Table info:  
    size: 10  
    color: black  
CompTable Volume = 1008  
CompTable::~~CompTable()  
Table::~~Table()  
-----
```

Вызов обновленных методов -1

```
cout<<"\t-----\n";
```

```
CompTable* CT;
```

```
CT=new CompTable(10,"black",2,"wood");
```

```
CT->Show();
```

```
cout << "CompTable Volume = "  
      << CT->CalcVolume() << endl;
```

```
delete CT; CT=NULL;
```

```
cout<<"\t-----\n";
```

```
-----  
Table::Table(int size1, char* color1)  
void Table::InitSize(int size1)  
bool Table::InitColor(char* color1)  
CompTable::CompTable(int, char*, int, char*)  
CompTable info:  
    h=2  
    material:wood  
Table info:  
    size: 10  
    color: black  
CompTable Volume = 1008  
CompTable::~~CompTable()  
Table::~~Table()  
-----
```

Вызов обновленных методов -2

```
cout<<"\t-----\n";  
Table *T;  
T=new Table(20,"white");  
T->Show();  
cout << "Table Volume = "  
      << T->CalcVolume() << endl;  
delete T; T=NULL;  
cout<<"\t-----\n";
```

```
Table::Table(int size1, char* color1)  
void Table::InitSize(int size1)  
bool Table::InitColor(char* color1)  
Table info:  
        size: 20  
        color: white  
Table Volume = 8000  
Table::~~Table()
```

Вызов обновленных методов -2

```
cout<<"\t-----\n";  
Table *T;  
T=new Table(20,"white");  
T->Show();  
cout << "Table Volume = "  
    << T->CalcVolume() << endl;  
delete T; T=NULL;  
cout<<"\t-----\n";
```

```
Table::Table(int size1, char* color1)  
void Table::InitSize(int size1)  
bool Table::InitColor(char* color1)  
Table info:  
    size: 20  
    color: white  
Table Volume = 8000  
Table::~~Table()
```

Вызов обновленных методов -2

```
cout<<"\t-----\n";  
Table *T;  
T=new Table(20,"white");  
T->Show();  
cout << "Table Volume = "  
    << T->CalcVolume() << endl;  
delete T; T=NULL;  
cout<<"\t-----\n";
```

```
Table::Table(int size1, char* color1)  
void Table::InitSize(int size1)  
bool Table::InitColor(char* color1)  
Table info:  
    size: 20  
    color: white  
Table Volume = 8000  
Table::~~Table()
```

Вызов обновленных методов -2

```
cout<<"\t-----\n";  
Table *T;  
T=new Table(20,"white");  
T->Show();
```

```
cout << "Table Volume = "  
      << T->CalcVolume() << endl;
```

```
delete T; T=NULL;
```

```
cout<<"\t-----\n";
```

```
Table::Table(int size1, char* color1)
```

```
void Table::InitSize(int size1)
```

```
bool Table::InitColor(char* color1)
```

```
Table info:
```

```
    size: 20
```

```
    color: white
```

```
Table Volume = 8000
```

```
Table::~~Table()
```

Вызов обновленных методов -2

```
cout<<"\t-----\n";  
Table *T;  
T=new Table(20,"white");  
T->Show();  
cout << "Table Volume = "  
      << T->CalcVolume() << endl;  
delete T; T=NULL;  
cout<<"\t-----\n";
```

```
Table::Table(int size1, char* color1)  
void Table::InitSize(int size1)  
bool Table::InitColor(char* color1)  
Table info:  
        size: 20  
        color: white  
Table Volume = 8000  
Table::~~Table()
```

Связывание

Связывание — это процесс, который используется для конвертации идентификаторов (таких как имена переменных или функций) в адреса.

Все функции имеют свой уникальный адрес.

Когда компилятор (или линкер) встречает вызов функции, он заменяет его инструкцией машинного кода, которая сообщает процессору перейти к адресу функции.

Раннее связывание -1

Раннее связывание (или «**статическая привязка**») означает, что компилятор (или линкер) может напрямую связать имя идентификатора с машинным адресом.

В терминах ООП **раннее связывание** означает, что объект и вызов функции связываются между собой на этапе компиляции.

Раннее связывание -2

Достоинство: выполняется быстрее и обычно требует меньше памяти, чем позднее связывание.

Недостаток: невысокая гибкость.

Раннее связывание -3

```
cout<<"\t-----\n";  
CompTable* CT;  
CT=new CompTable(10,"black",2, "wood");  
CT->Show();  
cout << "CompTable Volume = "  
    << CT->CalcVolume() << endl;  
delete CT; CT=NULL;  
cout<<"\t-----\n";
```

```
cout<<"\t-----\n";  
Table *T;  
T=new Table(20,"white");  
T->Show();  
cout << "Table Volume = "  
    << T->CalcVolume() << endl;  
delete T; T=NULL;  
cout<<"\t-----\n";
```

Позднее связывание -1

Позднее связывание (или «динамическая привязка») означает, что объект и вызов функции связываются между собой на этапе запуска программы.

Позднее связывание достигается в C++ с помощью использования виртуальных функций и производных классов.

Достоинство: высокая гибкость

Недостаток: снижение производительности

Позднее связывание -2

Оно может использоваться для поддержки общего интерфейса, позволяя различным объектам иметь свою собственную реализацию этого интерфейса!

Позднее связывание -3

CompTable * CT;

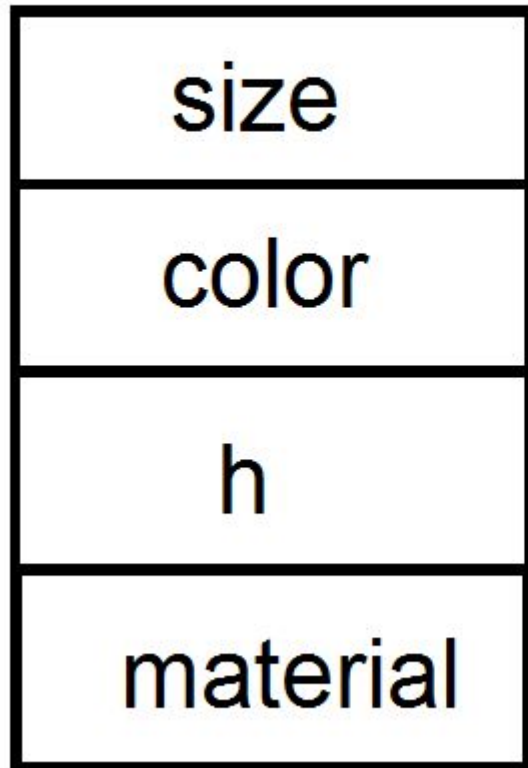
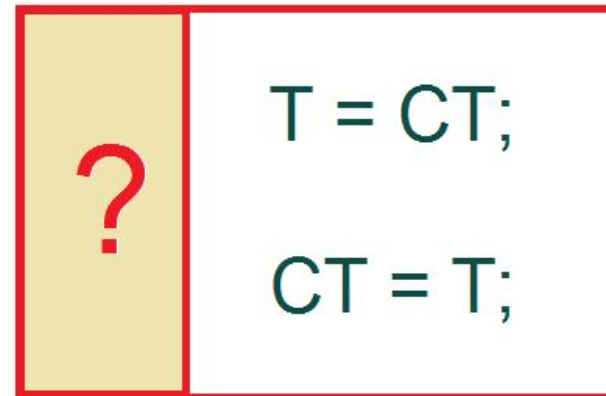


Table * T;



Позднее связывание -3

CompTable * CT;

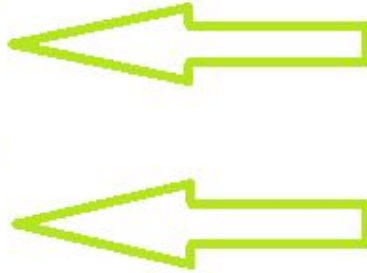
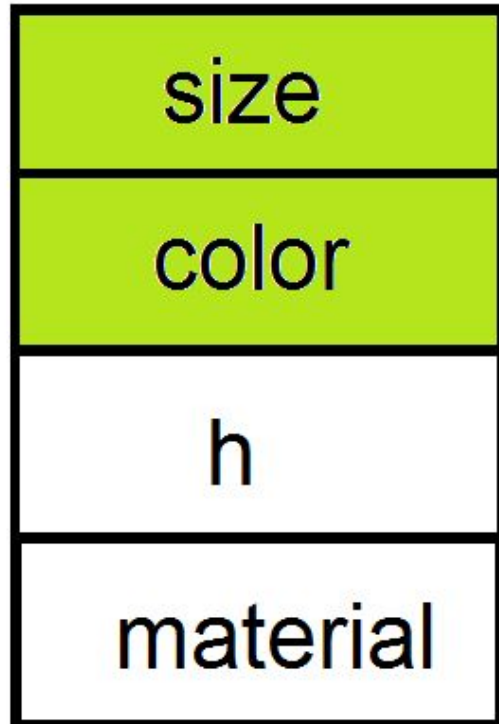


Table * T;



CT=T

Позднее связывание -3

CompTable * CT;

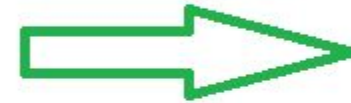
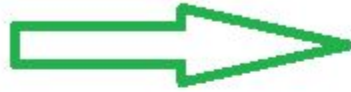
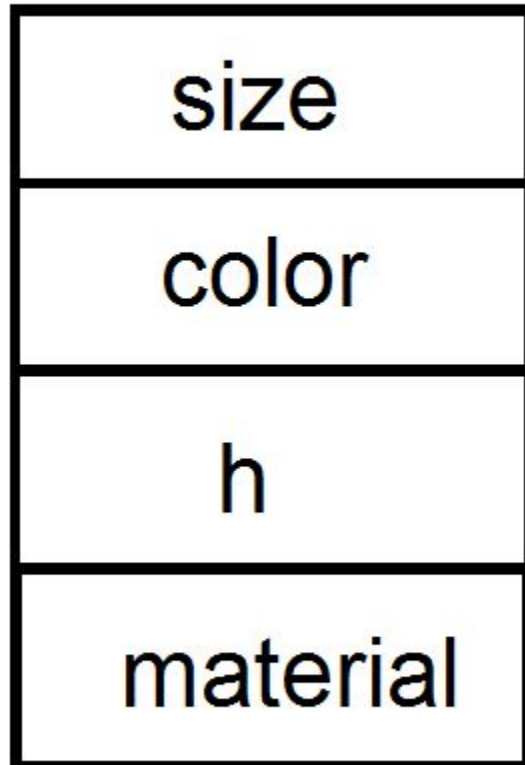


Table * T;



T=CT

Пример - 1

```
int key;
Table *T;

do {
    cout<<"Enter object type:\n";
    cout<<"\t - 1 -  Table\n";
    cout<<"\t - 2 -  CompTable\n";
    cout<<endl;
    cin>>key;
    if (key!=1 && key!=2)
        cout<<"-->>Bad choice, press 1 or 2\n";
}while (key!=1 && key!=2);
```

Пример - 2

```
switch (key)
{
    case 1:
        T=new Table(20, "white");
        break;
    case 2:
        T=new CompTable(10, "black", 2, "metal");
        break;
    default: cout<<"Error, unknown object\n";
}
```

Пример - 3

```
T->Show();  
cout<<"Table Volume = "<<T->CalcVolume()<<endl;  
delete T; T=NULL;
```

Пример – 4 (запуск программы)

```
Enter object type:  
    - 1 - Table  
    - 2 - CompTable
```

1

```
Table::Table(int size1, char* color1)  
void Table::InitSize(int size1)  
bool Table::InitColor(char* color1)  
Table info:  
    size: 20  
    color: white  
Table Volume = 8000  
Table::~~Table()
```

Пример – 5 (запуск программы)

```
Enter object type:
```

- 1 - Table
- 2 - CompTable

```
2
```

```
Table::Table(int size1, char* color1)  
void Table::InitSize(int size1)  
bool Table::InitColor(char* color1)  
CompTable::CompTable(int, char*, int, char*)
```

```
Table info:  
    size: 10  
    color: black  
Table Volume = 1000  
Table::~~Table()
```

```
class Table
{
    int* size;
    char* color;
    bool InitSize(int);
    bool InitColor(char*);
public:
    Table();
    Table(int, char*);
    Table(int);
    ~Table();
    void Show();
    int CalcVolume();
    void SetSize(int);
    void SetColor(char*);
    void GetSize(int&);
    void GetColor(char*);
};
```

Зачем нужен virtual?

```
class Table
{
    int* size;
    char* color;
    bool InitSize(int);
    bool InitColor(char*);
public:
    Table();
    Table(int, char*);
    Table(int);
    virtual ~Table();
    virtual void Show();
    virtual int CalcVolume();
    void SetSize(int);
    void SetColor(char*);
    void GetSize(int&);
    void GetColor(char*);
};
```

virtual нужен для полиморфизма

Виртуальные методы

```
virtual ~Table();  
virtual void Show();  
virtual int CalcVolume();
```


Соответствующие методы потомка автоматически становятся Виртуальными!

```
class CompTable: public Table
{
    int* h;
    char* material;
public:
    CompTable();
    CompTable(int, char*, int, char*);
    virtual ~CompTable();
    virtual int CalcVolume();
    virtual void Show();
};
```

Как это работает

```
Enter object type:
```

- 1 - Table
- 2 - CompTable

```
2
```

```
Table::Table(int size1, char* color1)  
void Table::InitSize(int size1)  
bool Table::InitColor(char* color1)  
CompTable::CompTable(int, char*, int, char*)
```

```
CompTable info:
```

```
    h=2  
    material:metal
```

```
Table info:
```

```
    size: 10  
    color: black
```

```
Table volume = 1008
```

```
CompTable::~~CompTable()
```

```
Table::~~Table()
```