Глава 5.
Объектная модель C++

МГТУ им. Н.Э. Баумана
Факультет Информатика и системы управления
Кафедра Компьютерные системы и сети
Лектор: д.т.н., проф.
Иванова Галина Сергеевна

5.1 Описание класса

Формат описания класса:

class <Имя класса>

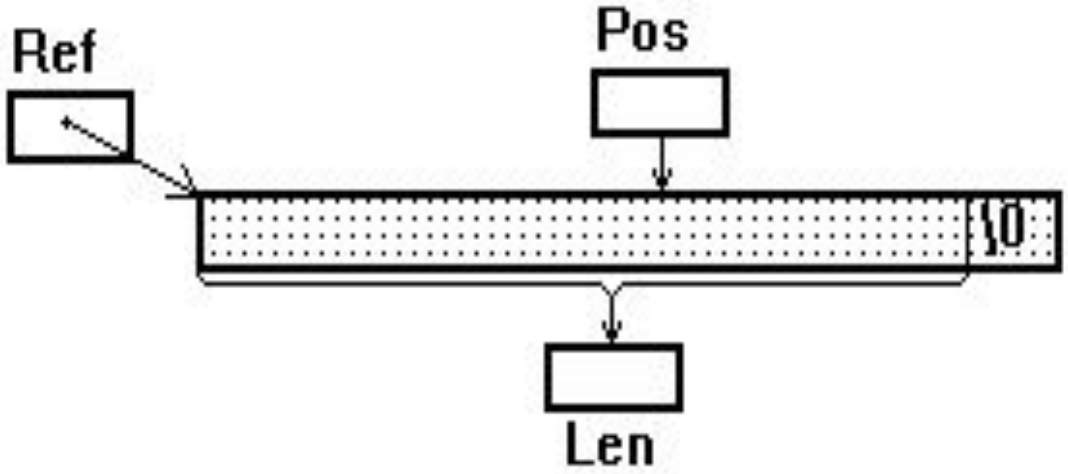
{ private: <Внутренние компоненты класса>;

protected: <Защищенные компоненты класса>;

public: <Общедоступные компоненты класса>;

};

Пример: Объект – строка (Ex5_01)



TSlovo
-Len -Ref #Pos
+TSlovo() +~TSlovo() +GetCh() +*PrintPos*()

Пример описания класса (файл Slovo.h)

```
class TSlovo
{ private:   int Len;
             char *Ref;
  protected: int Pos;
  public:
    TSlovo(char *ref,int pos=0);
    ~TSlovo(void) {delete Ref;}
    char GetCh(void)
    {   char Chr=Pos<Len?Ref[Pos]:'\0';
        if(!Chr)  Pos=-1;
        return Pos++,Chr;
    }
    virtual void PrintPos(void);
};
```

inline

inline

Конструктор

Деструктор

Пример описания методов (файл Slovo.cpp)

```
#include "stdafx.h"
#include "slovo.h"
#include <iostream>
#include <string.h>

TSlovo::TSlovo(char *ref,int pos):Pos(pos)
    {   Len=strlen(ref);
        Ref=new char[Len+1];
        strcpy(Ref,ref);
    }
void TSlovo::PrintPos(void)
    {
    std::cout<<Pos;
    }
```

5.2 Объявление объектов и обращение к полям

<Имя класса> <Список переменных и/или указателей>;

Примеры:

а) `TPoint a,*b, c[5];` /* класс описан без конструктора или с конструктором без параметров */

б) `TSlovo D("Это строка", 4);` // конструктор с параметрами

<Имя объекта>.<Имя поля или метода>

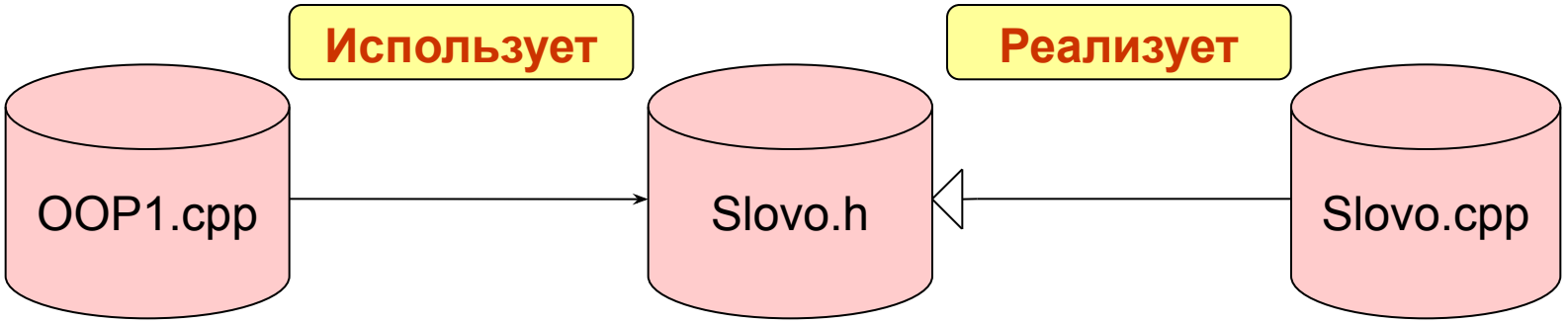
<Имя указателя на объект> -><Имя поля или метода>

<Имя массива>[<Индекс>] .<Имя поля или метода>

Ссылка **Self (Паскаль) ⇔ this (C++)** **Указатель**

Пример: `this->Pos`

Тестирующая программа




```
#include "stdafx.h"
#include <iostream.h>
#include "Slovo.h"

int main(int argc, char* argv[])
{   TSlovo Greet("Hello World",6);
    char Chr;
    while (Chr=Greet.GetCh())   cout<<Chr<<' ';
    return 0;
}
```

W o r l d

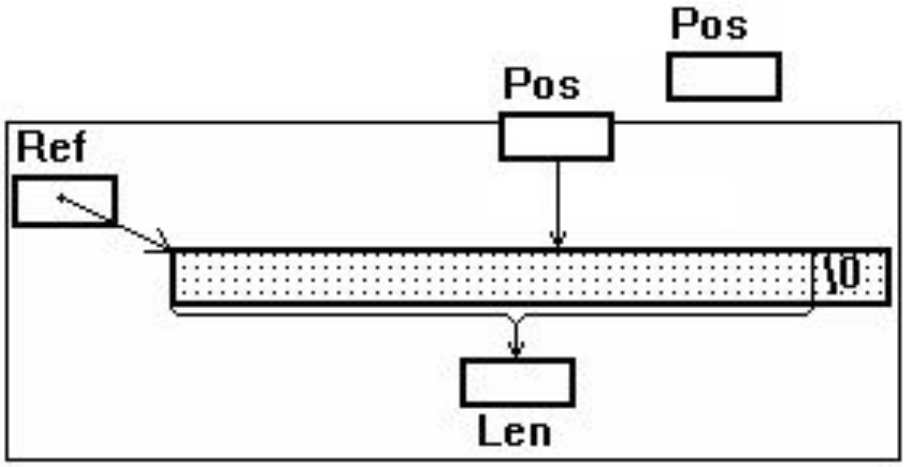
5.3 Наследование

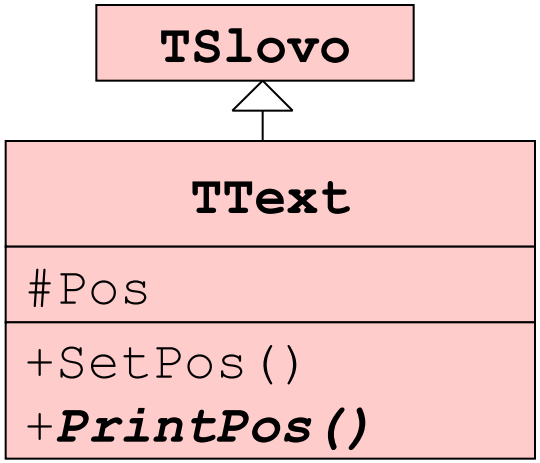
class <Имя производного класса>:

<Вид наследования> <Имя базового класса>{…};

Вид наследования	Объявление поля в базовом классе	Видимость в производном классе
private	private protected public	не видимо private private
protected	private protected public	не видимо protected protected
public	private protected public	не видимо protected public

Пример наследования Ex5_02 (Text.h)






```
#include "f:\iva\primer.vc\lection\oopr1\slovo.h"
#include <iostream.h>
class TText:public TSlovo
{   private:     int Pos;
    public:
inline  TText(char *ref,int pos=0):TSlovo(ref),Pos(pos) {}
inline  void SetPos()  { TSlovo::Pos=Pos; }
virtual void PrintPos();
};
void TText::PrintPos(){cout<<TSlovo::Pos<<':'<<Pos; }
```

Тестирующая программа

```
#include "stdafx.h"
#include "Text.h"

int main(int argc, char* argv[])
{   TText Greet("Hello World",6);
    char Chr;
    while (Chr=Greet.GetCh()) cout<<Chr;
    cout<<'\n';
    Greet.SetPos();
    while (Chr=Greet.GetCh()) cout<<Chr;
    cout<<'\n';
    Greet.PrintPos();
    return 0;
}
```

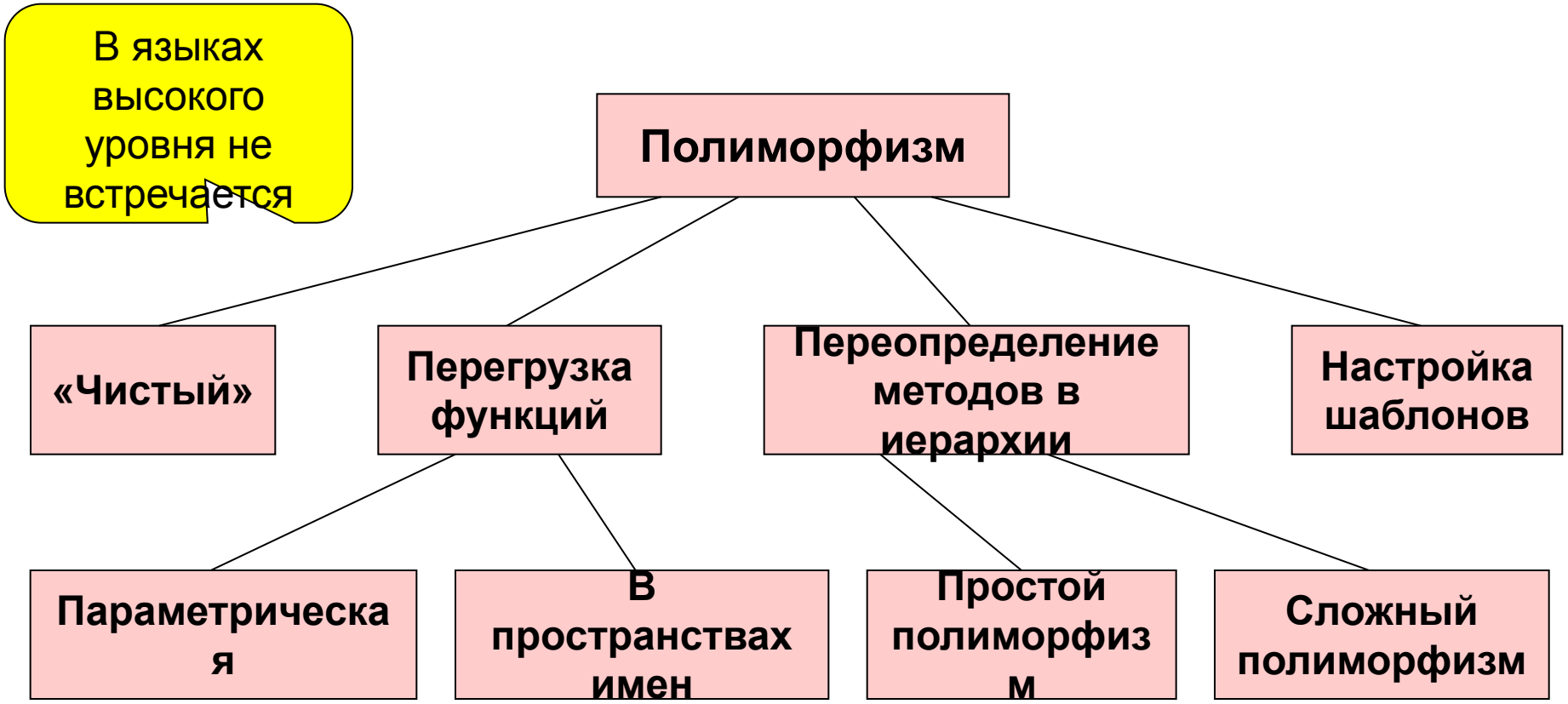
```
TText(char *ref,int pos=0):
   TSlovo(ref),Pos(pos)
```

```
void
SetPos(){TSlovo::Pos=Pos;}
```

```
Hello World
World
0:6
```

5.4 Полиморфизм

Полиморфизм – «многоформие», т.е. свойство изменения формы. В программировании встречаются следующие виды полиморфизма:



Полиморфизм

В Паскале:

простой полиморфизм

сложный полиморфизм

В C++:

переопределение методов

виртуализация методов

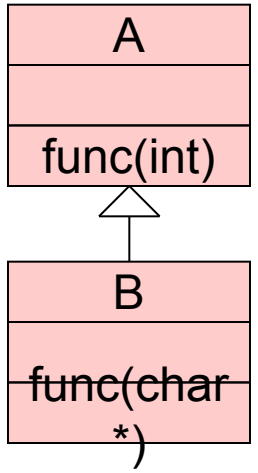
Пример использования сложного полиморфизма (Ex5_03):

```
#include "stdafx.h"
#include "f:\iva\primer.vc\lection\oopr11\Ttext.h"
int main(int argc, char* argv[])
{  TSlovo SGreet("HELLO!!!"), *Ref=&SGreet;
   TText Greet("Hello World",6);
   Ref->PrintPos();
   cout<<endl;
   Ref=&Greet;
   Ref->PrintPos();
   cout<<endl;
    return 0;
}
```

```
0
0:6
```

Использование пространств имен для перегрузки методов класса (Ex5_111)

```
#include "stdafx.h"
#include <conio.h>
class A
  {    public:    void func(int ch);
  };
class B : public A
  {    public:
          void func(char *str);
          using A::func;  // перегрузить B::func
  };
void A::func(int ch)     // метод базового класса
{ std::cout<<"Symbol\n"; }
void B::func(char *str) // метод производного класса
{ std::cout<<"String\n"; }
int main()
{   B b;
    b.func(25);      // вызов  A::func()
    b.func("ccc");   // вызов  B::func()
   getch();
   return 0;}
```



5.5 Инициализация общих полей объектов при отсутствии конструкторов

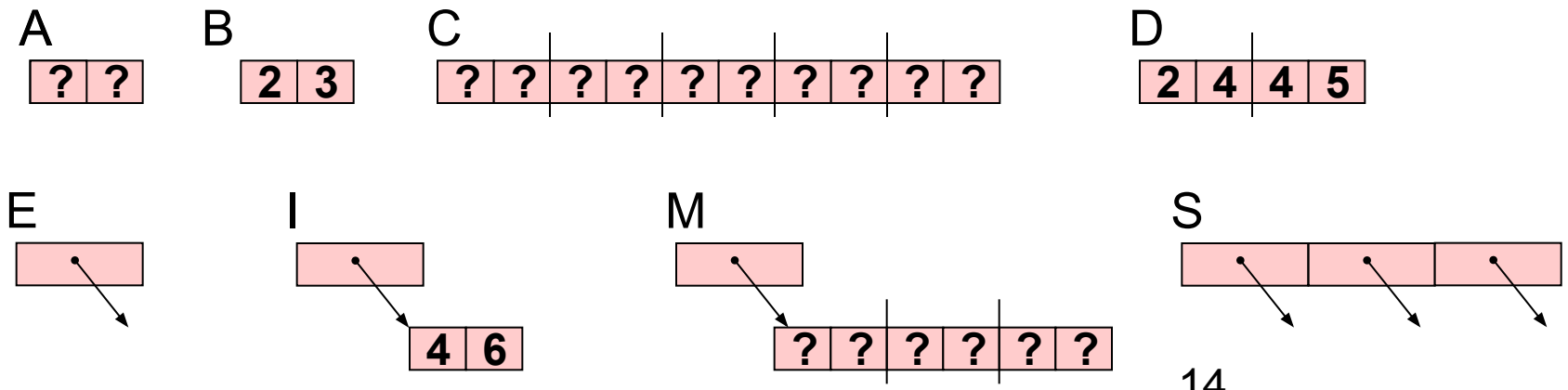
Пример:

```
class TPoint
        {public: int x,y;
         ...};

int main()
{ TPoint A = {2,3};
  TPoint *pB = {5,3};
  TPoint C[]= {{4,5},{2,8},{7,6}};
...
}
```

5.6 Конструкторы. Список инициализации

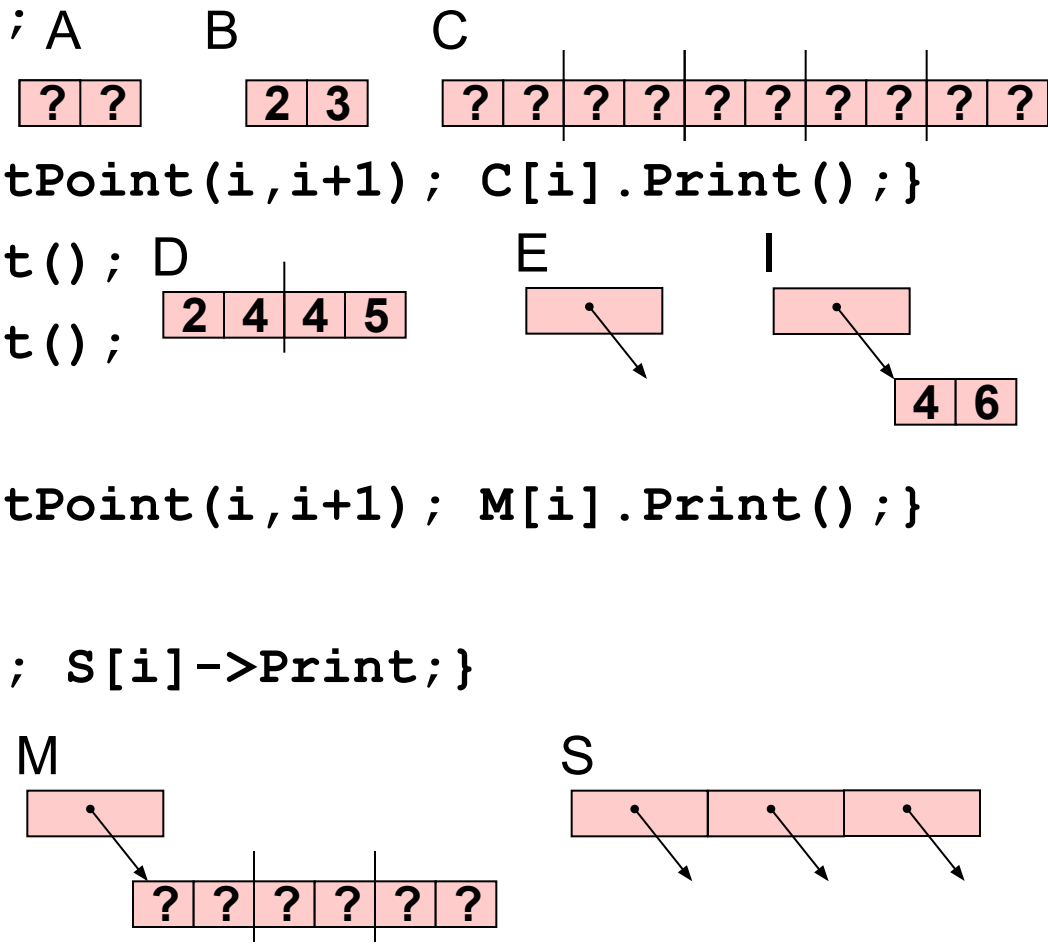
Пример (Ex5_4):

```
class TPoint
{ private: int x,y;
  public: TPoint(int ax,int ay){x=ax;y=ay;}
          TPoint(){}
          SetPoint(int ax,int ay) {x=ax;y=ay;}   ...};
int main()
{ TPoint A, B(2,3), C[5],  D[2] = {TPoint(2,4),TPoint(4,5)},
    *E,    *I = new TPoint(4,6), *M = new TPoint[3], *S[3];
```



Распределение/освобождение памяти и инициализация объектов в программе

```
A.SetPoint(2,3); A.Print();
B.Print();
for (i=0;i<5;i++) {C[i].SetPoint(i,i+1); C[i].Print();}
for(i=0;i<2;i++) d[i].Print();
E=new TPoint(3,4); E->Print();
I->Print();
for (i=0;i<3;i++) {M[i].SetPoint(i,i+1); M[i].Print();}
for (i=0;i<3;i++)
   {S[i]=new TPoint(i,i+1); S[i]->Print;}
delete E;
delete I;
delete [] M;
for (i=0;i<3;i++) delete S[i];
}
```



Список инициализации. Инициализация объектных полей

Формат элемента списка инициализации:
<Имя поля>(<Список выражений>)

Примеры:

а)
```
TPoint(int ax,ay): x(ax),y(ay) {}
```

б)
```
class TLine
{ private:
      const int x;
      int &y;
      TPoint t;
  public: TLine(int ax,int ay,int tx,int ty):
                x(ax),y(ay),t(tx,ty){}
          TLine(){}
  ...};
```

Автоматически вызывает конструктор объектного поля без параметров TPoint( ) !

5.7 Копирующий конструктор

Автоматически вызывается:

а) при использовании объявлений типа

```
TPoint A(2,5),B=A;
```

б) при передаче параметров-объектов по значению, например:

```
void Print(TPoint R) {…}
```

Формат:

```
<Имя конструктора>(const <Имя класса> &<Имя объекта>){…}
```

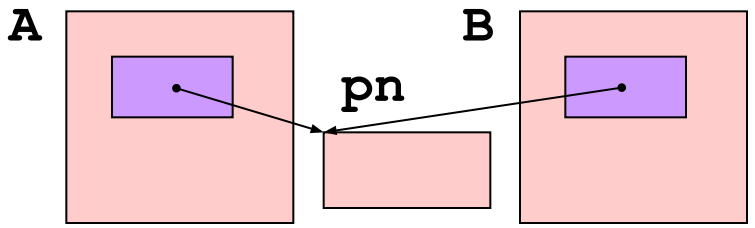
Примеры:

а)
```
TPoint(const TPoint &Obj)
    {x=Obj.x; y=Obj.y;}
```

б)
```
TPoint(const TPoint &Obj)
    {x=Obj.x; y=2*Obj.y;}
```

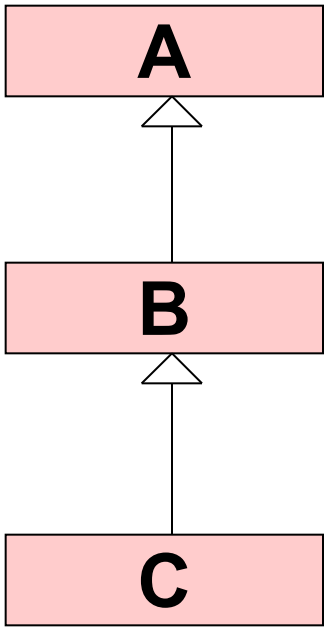
Строится автоматически

Пример обязательного определения копирующего конструктора (Ex5_05)




```
#include "stdafx.h"
#include <stdio.h>
class TNum
{ public:int *pn;
   TNum(int n){puts("new pn"); pn=new int(n);}
   TNum(const TNum &Obj)
        {puts("copy new pn"); pn=new int(*Obj.pn);}
   ~TNum(){puts("delete pn");delete pn;}
};
void Print(TNum b) { printf("%d ",*b.pn); }

int main(int argc, char* argv[])
{ TNum A(1);
  Print(A); return 0;
}
```

5.8 Конструкторы и деструкторы производных классов



`A(){…}`

`B():A(),`<Конструкторы полей>`{}`

`C():B(),`<Конструкторы полей>`{}`

При объявлении объектов производного класса всегда вызывается конструктор базового класса.

Если в списке инициализации конструктора производного класса вызов конструктора базового отсутствует, то автоматически вызывается конструктор базового класса без параметров

Вызов конструкторов и деструкторов для объектов производных классов (Ex5_06)

```
#include "stdafx.h"
#include <stdio.h>
class TNum
   { public:   int n;
       TNum(int an):n(an) {puts("TNum(an)");}
       TNum() {puts("TNum()");}
       ~TNum(){puts("~TNum");}
   };
class TNum2:public TNum
   { public:   int nn;
       TNum2(int an):nn(an) {puts("TNum2(an)");}
       ~TNum2(){puts("~TNum2");}
   };

int main(int argc, char* argv[])
{TNum2 A(1);return 0;}
```

Неявный вызов конструктора TNum

TNum()
TNum2(an)
~TNum2
~TNum

5.9 Абстрактные методы и классы

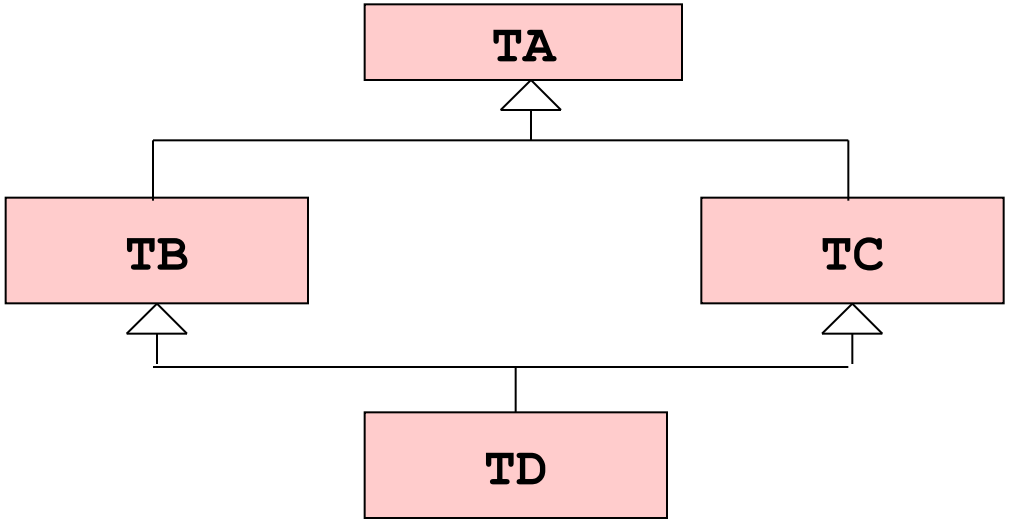
Абстрактный метод.

```
class AClass
     { …
        virtual int Fun(int,int)=0;
     }
```

Класс, содержащий абстрактный метод, называется абстрактным.

Объекты абстрактного класса создавать запрещено !

5.10 Множественное и виртуальное наследование



Проблема: дублирование полей базового класса

```
class <Имя>: virtual <Вид наследования> <Имя базового класса>
{ ...};
```

Порядок вызовов конструкторов:

- **конструктор виртуально наследуемого базового класса,**
- **конструкторы базовых классов в порядке их перечисления при объявлении производного класса,**
- **конструкторы объектных полей и конструктор производного класса.**

Деструкторы соответственно вызываются в обратном порядке.

Пример множественного виртуального наследования

```
#include "stdafx.h"
#include <iostream.h>
class TA
   { protected:   int Fix;
     public:TA() { cout<<"Inside A\n";}
        TA(int  fix):Fix(fix) { cout<<"Inside TA int\n";}
   };
class TB:virtual public TA
   { public:  int One;
        TB(void){ cout<<"Inside TB\n";}
   };
```

Пример множественного виртуального наследования (2)

```
class TC:virtual private TA
    { public:int Two;
          TC() { cout<<"Inside TC\n";}
    };
class TD:public TB,public TC
    { public:
            TD(int fix):TA(fix){cout<<"Inside TD\n";}
            void Out( ) {cout<<Fix; }
    };
main()
   { TD Var(10);
       Var.Out( );
       return 0;
   }
```

```
Inside TA int
Inside TB
Inside TC
Inside TD
10
```

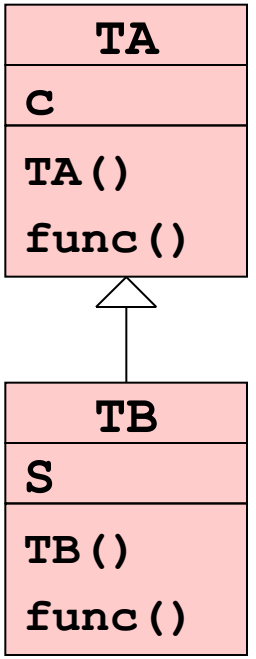
5.11 Приведение типов объекта

В C++ для приведения типов используют:

1) **(<Тип>)<Переменная> - используется в Си/C++ для любых типов, ничего не проверяет;**
2) **static_cast <Тип>(<Переменная>)- используется в C++ для любых типов, ничего не проверяет;**
3) **reinterpret_cast <Тип указателя> (<Указатель или интегральный тип>)- используется в C++ для указателей, ничего не проверяет;**
4) **dynamic_cast <Тип указателя на объект> (<Указатель на объект>) – используется в C++ для полиморфных классов, требует указания опции компилятора /GR (Project/Settings…), если приведение невозможно, то возвращает NULL.**

Пример приведения типов объектов (Ex5_07)

```
#include "stdafx.h"
#include <iostream.h>
#include <string.h>

class TA
{ protected:char c;
  public: TA(char ac):c(ac){}
   virtual void func(){cout<<c<<endl;}
};
class TB:public TA
{ char S[10];
public: TB(char *aS):TA(aS[0]){strcpy(S,aS);}
        void func(){cout<<c<<' '<<S<<endl;}
};
```



Пример приведения типов объектов(2)

```
int main(int argc, char* argv[])
{ TA *pA=new TA('A'),*pC=new TB("AB");
  TB *pB=new TB("AC");

  ((TA *)pB)->func();
  reinterpret_cast<TA *>(pB)->func();
  static_cast<TA *>(pB)->func();
  dynamic_cast<TA *>(pB)->func();

  ((TB *)pC)->func();
  reinterpret_cast<TB *>(pC)->func();
  static_cast<TB *>(pC)->func();
  dynamic_cast<TB *>(pC)->func();

  ((TB *)pA)->func();
  reinterpret_cast<TB *>(pA)->func();
  static_cast<TB *>(pA)->func();
 // dynamic_cast<TB *>(pA)->func();
  if (TB *pD=dynamic_cast<TB *>(pA)) pD->func();
  else cout<<"Cast Error"<<endl;
  return 0;}
```

Восходящее приведение

Нисходящее приведение

Ошибка! Приведение не корректно

5.12 Контейнер «Двусвязный список» (Ex5_08)

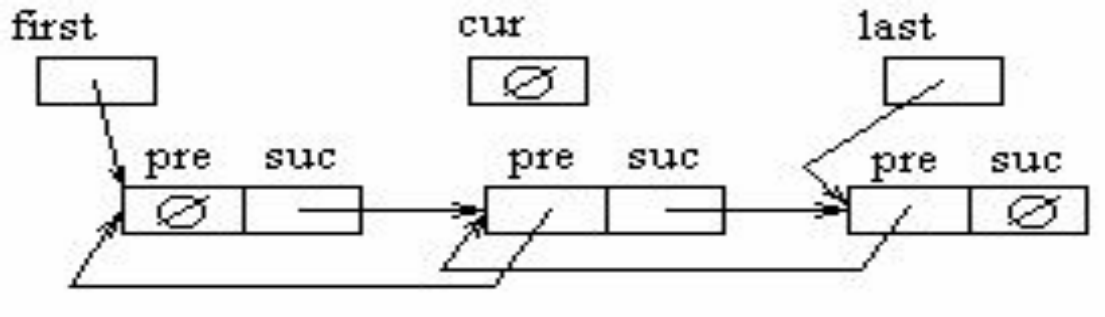
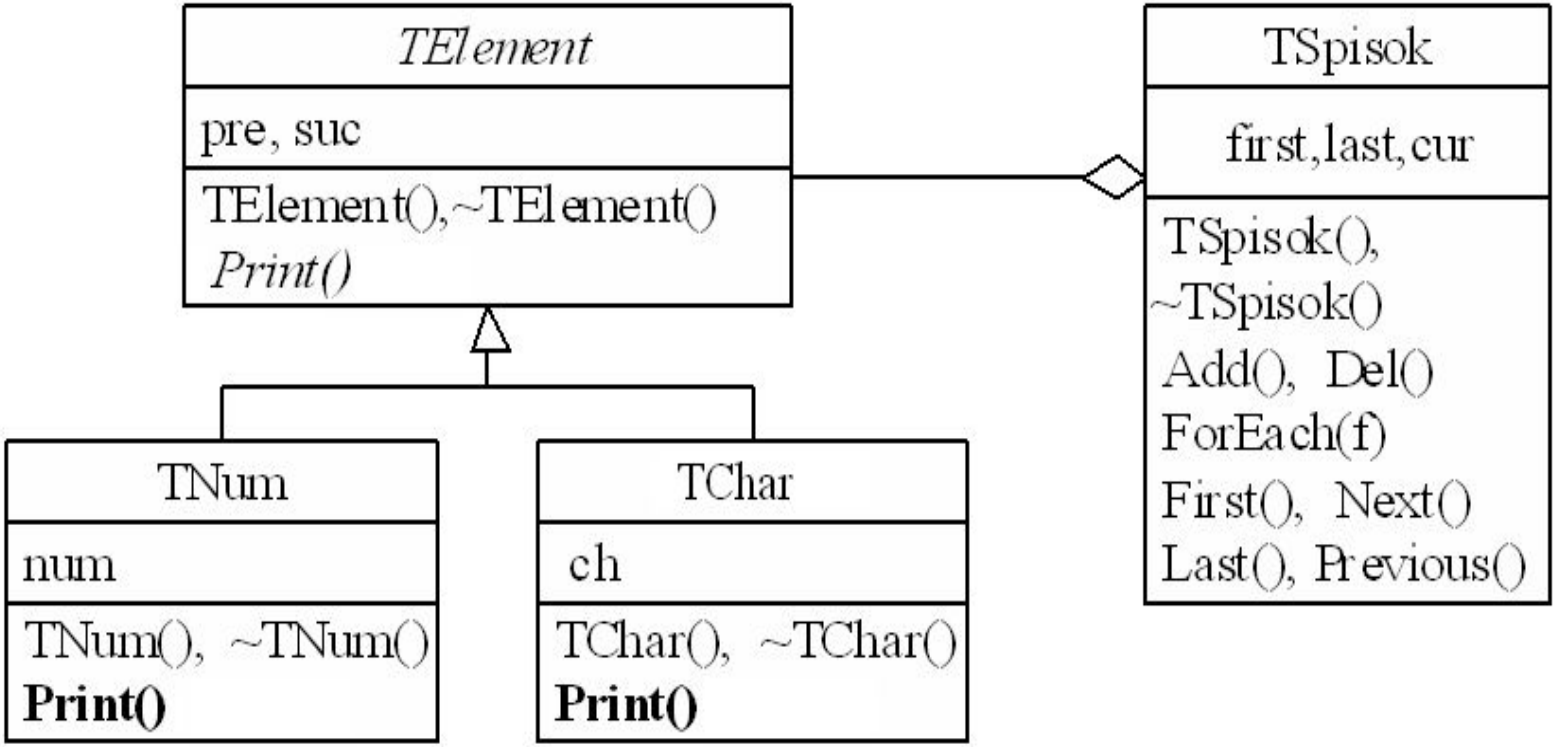
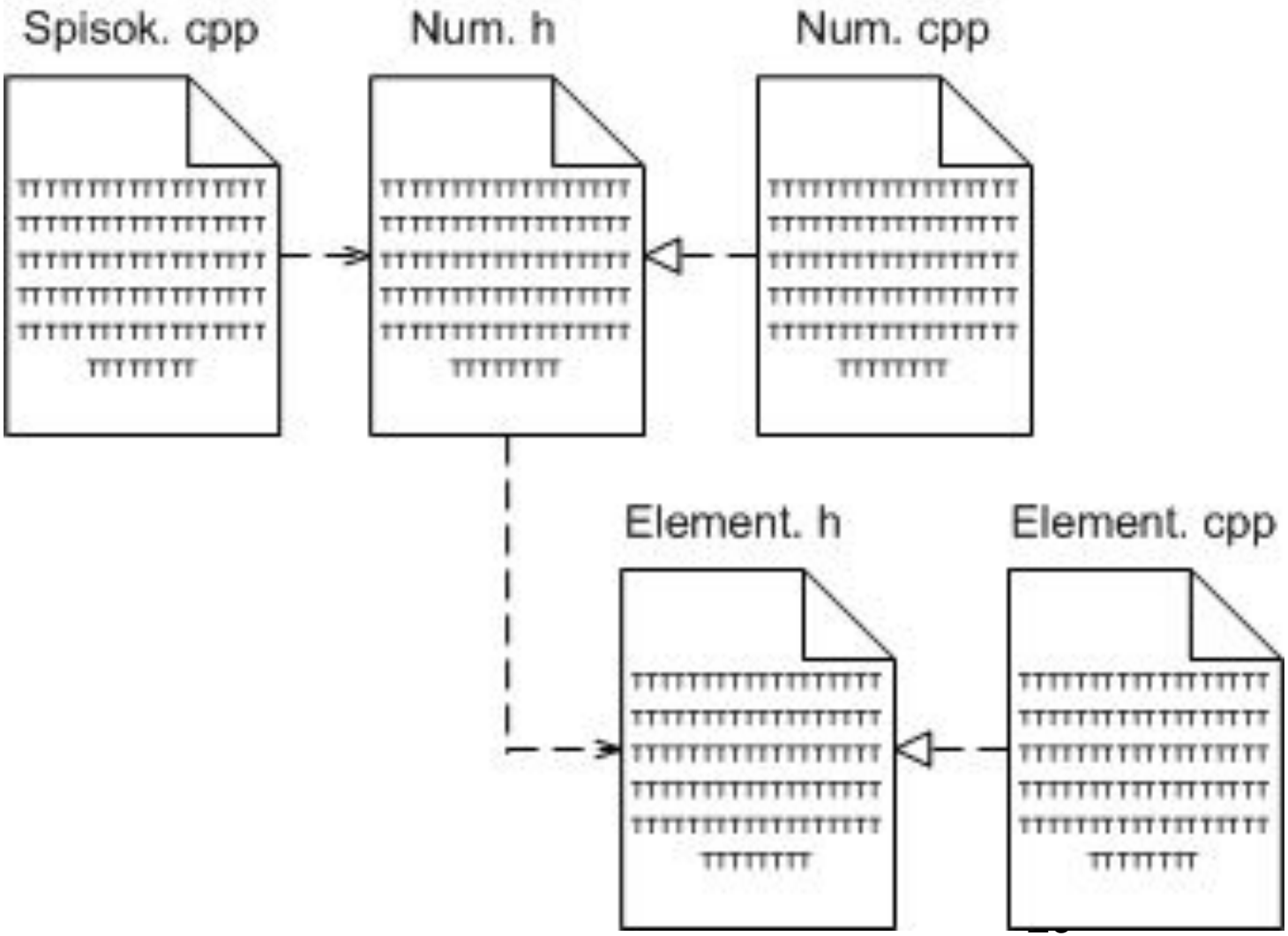


Диаграмма классов



Контейнер «Двусвязный список»(2)

Диаграмма компоновки



Файл Element.h

```
#include <stdio.h>
class TElement
  {  public:    TElement *pre,*suc;
     TElement() { pre=suc=NULL;}
     virtual ~TElement() { puts("Delete TElement.");}
     virtual void Print()=0;
  };
class TSpisok
  {  private: TElement *first,*last,*cur;
     public:  TSpisok() {first=last=cur=NULL;}
    ~TSpisok();
    void Add(TElement *e);
    TElement *Del();
    void ForEach(void (*f)(TElement *e));
    TElement *First(){return cur=first;}
    TElement *Next(){return cur=cur->suc;}
    TElement *Last(){return cur=last;}
    TElement *Previous(){return cur=cur->pre;}
  };
```

Файл Element.cpp

```
#include "stdafx.h"
#include "Element.h"
TSpisok::~TSpisok()
{  puts("Delete TSpisok");
   while ((cur=Del())!=NULL)  { cur->Print();
                                delete(cur);  }
}
void TSpisok::Add(TElement *e)
{  if (first==NULL) first=last=e;
   else  { e->suc=first;
           first->pre=e;
           first=e;  }
}
```

Файл Element.cpp (2)

```
TElement *TSpisok::Del(void)
{  TElement *temp=last;
   if (last!=NULL)
      {last=last->pre;
       if (last!=NULL) last->suc=NULL;
      }
   if (last==NULL) first=NULL;
   return temp;
}
void TSpisok::ForEach(void (*f)(TElement *e))
{  cur=first;
   while (cur!=NULL)
      {(*f)(cur);
        cur=cur->suc;
      }
}
```

Файл Num.h

```
#include "Element.h"
class TNum:public TElement
{ public: int num;
          TNum(int n):TElement(),num(n) {}
          ~TNum() { puts("Delete TNum.");}
          void Print() { printf("%d ",num); }
};
class TChar:public TElement
{ public: char ch;
          TChar(char c):TElement(),ch(c) {}
          ~TChar() { puts("Delete TChar.");}
          void Print() { printf("%c ",ch);}
};
void Show(TElement *e);
```

Файл Num.cpp

```
#include "stdafx.h"
#include "Num.h"
void Show(TElement *e)
{ e->Print();}
```

Тестирующая программа

```
#include "stdafx.h"
#include "Num.h"
#include <string.h>
#include <stdlib.h>
TSpisok N;
int main(int argc, char* argv[])
{  char str[10];
   int k,i;
   TElement *p;
   while(printf("Input numbers, strings or <end>:"),
         scanf("%s",str),strcmp(str,"end"))
   { k=atoi(str);
     if (k||(strlen(str)==1 && str[0]=='0'))  p=new TNum(k);
     else p=new TChar(str[0]);
     N.Add(p);
   }
   puts("All list:");
   N.ForEach(Show);
```

Тестирующая программа(2)

```
   p=N.First(); k=0;
   while (p!=NULL)
    { if (TNum *q=dynamic_cast<TNum *>(p)) k+=q->num;
   // установить создание RTTI (/GR в Project\Settings...)
      p=N.Next();
    }
   printf("Summa= %d\n",k);
   p=N.Last();
   i=0;
   while (p!=NULL)
    {if (TChar *q=dynamic_cast<TChar *>(p)) str[i++]=q->ch;
      p=N.Previous();
    }
   str[i]='\0';
   printf("String= %s\n",str);
   return 0;
}
```

5.13 Статические компоненты класса

Объявляются с описателем **`Static`**

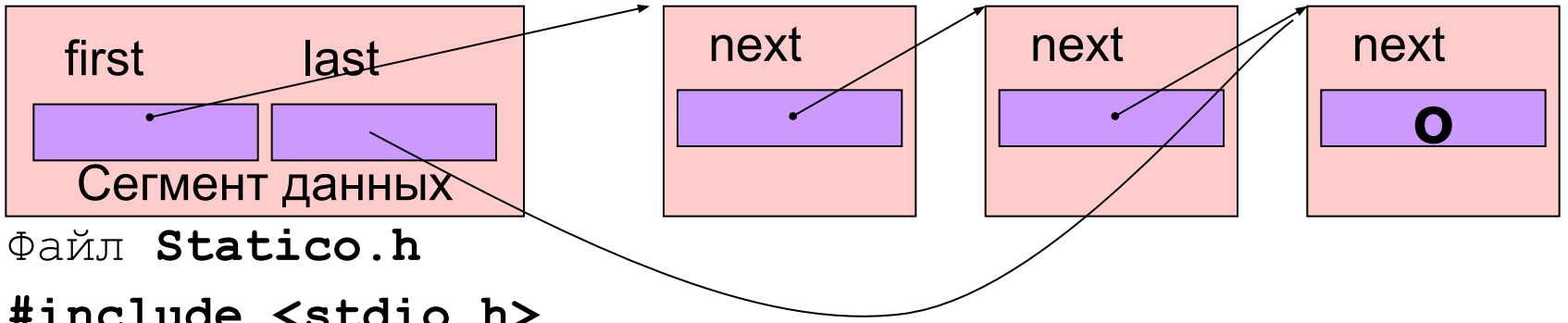
Статические поля являются общими для всех объектов класса и существуют даже при отсутствии объектов. Инициализация статических полей в определении класса не допустима.

Статические методы не получают параметра this и, следовательно, не могут без указания объекта обращаться к нестатическим полям.

Для доступа к статическим компонентам вне компонентных функций используют квалификатор **<класс>::**

Статические компоненты класса (Ex5_09)

Пример. Создать список объектов



Файл **Statico.h**

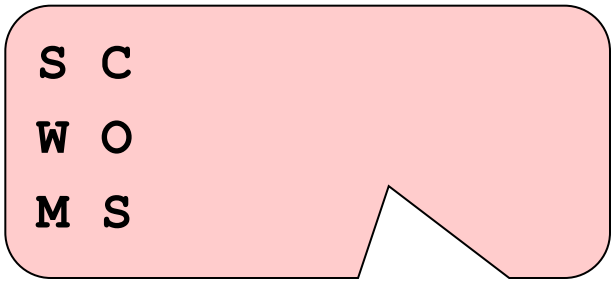
```
#include <stdio.h>
#include <conio.h>
class TPoint
  { public: char ch1,ch2;
        static TPoint *first, *last;
        TPoint *next;
        TPoint(char ach1,char ach2);
        void Draw()
         { printf("%c   %c  \n",ch1,ch2); }
        static void DrawAll();
  };
```

Файл Statico.cpp

```
#include "stdafx.h"
#include "statico.h"

TPoint *TPoint::first=NULL,*TPoint::last=NULL;

TPoint::TPoint(char ach1,char ach2)
         { ch1=ach1; ch2=ach2;
           next=NULL;
           if(first==NULL)first=this;
           else last->next=this;
           last=this;
         }
void TPoint::DrawAll()
      { TPoint *p=first;
        if(p==NULL)return;
        do {p->Draw();
     p=p->next;}
        while(p!=NULL);
      }
```

Тестирующая программа

```
#include "stdafx.h"
#include "statico.h"

int main(int argc, char* argv[])
{ TPoint A('S','C'),B('W','O'),C('M','S');
  if(TPoint::first!=NULL) TPoint::DrawAll();
  return 0;
}
```



5.14 Дружественные функции и классы

Описываются с описателем **friend**, **что обеспечивает доступ к внутренним компонентам класса**

Пример:

```
class TPoint
    {private: int x,y;
     public:...
     friend void Show(TPoint A); // функция
    };
void Show(TPoint A){cout<<A.x<<' '<<A.y;}

int main()
{ TPoint W(2,3);
  Show(W);
  ... }

friend void TLine::Show(TPoint A); // метод

friend class TLine;                // класс
```

5.15 Переопределение операций
Операции

Типы функций-операций:

1. Независимая функция-операция

а) **<Тип результата> operator@(<Операнд>)**

б) **<Тип результата> operator@(<Операнд1>,<Операнд2>)**

2. Компонентная функция-операция

а) **<Тип результата> operator@( )** // Операнд = Объект

б) **<Тип результата > operator@(<Операнд2>)** // Операнд1 = Объект

Формы вызова

а) стандартная	б) операторная
operator@(<Арг>)	**@<Арг>**
operator@(<Арг1>,<Арг2>)	**<Арг1>@<Арг2>**
<Арг>. operator@( )	**@<Арг>**
<Арг1>. operator@(<Арг2>)	**<Арг1>@<Арг2>**

Переопределение операций

1. Можно переопределять только операции, параметры которых – объекты.
2. Не разрешается переопределение *, sizeof, ? :, #, ##, ::, Class::.
3. Операции =, [ ], ( ) можно переопределять только в составе класса
4. При переопределении операций нельзя изменить ее приоритет и ассоциативность.

Пример 1. Класс «Точка» (Ex5_10)

Файл Tpoint.h

```
#include <iostream.h>
class TPoint{
   private:      float x,y;
   public:
      TPoint(float ax,float ay):x(ax),y(ay)
           {cout<<"Constructor\n";}
      TPoint(){cout<<"Constructor without parameters\n";}
      TPoint(TPoint &p){ cout<<"Copy Constructor\n";
                    x=p.x; y=p.y;
                  }
      ~TPoint(){cout<<"Destructor\n";}
      void Out(void)     { cout<<"\n{"<<x<<","<<y<<"}\n"; }
      TPoint& operator+=(TPoint &p);  // a+=b;
      TPoint operator+(TPoint &p);    // a+b;
      TPoint& operator=(TPoint &p);   // a=b;
};
```

Файл Tpoint.cpp

```
#include "stdafx.h"
#include  "Tpoint.h"

TPoint& TPoint::operator+=(TPoint &p)
    { x+=p.x; y+=p.y;      cout<<"operator+=\n";
      return *this;
    }
 TPoint TPoint::operator+(TPoint &p)
    { TPoint pp(x,y);       cout<<"operator+\n";
      return pp+=p;
    }
 TPoint& TPoint::operator=(TPoint &p)
    { x=p.x; y=p.y;        cout<<"operator=\n";
      return *this;
    }
```

Тестирующая программа

```
#include "stdafx.h"
#include "Tpoint.h"
int main(int argc, char* argv[])
{    TPoint p(2,3),q(4,5),r(7,8);
     p+=r;
     p.Out();
     q=p+r;

     q.Out();
     return 0;
}
```

Constructor
Constructor
Constructor

Operator +=

Constructor (pp)
Operator +
Operator +=
Copy constructor
Destructor (pp)
Operator =
Destructor

TPoint pp(x,y);
cout<<"operator+\n";
pp+=p;
return
}

x+=p.x; y+=p.y;
cout<<"+=\n";
return *this;

Destructor
Destructor
Destructor

Пример 2. Класс «Строка»(Ex5_11)

Файл S.h:

```
#include <string.h>
#include <stdio.h>
#include <iostream.h>
#include <conio.h>
class String
 { private:     char *str,name;        int  len;
   public:  String(int Len,char Name);
            String(char *vs,char Name);
            String(String &S);
            ~String();
            int Length(){return len;}
            char operator[](int n)
              {return ((n>=0)&&(n<len))?str[n]:'\0';}
            void print() { cout<<"Str: "<<name<<":";
               cout<<str<<" Length: "<<len<<endl; }
            String  operator+(String &A);
            String  operator+(char c);
            String& operator=(String &S);
  };
```

Файл S.cpp

```
#include "stdafx.h"
#include "s.h"
String::String(int Len,char Name){ len=Len;
    str=new char[len+1]; str[0]='\0'; name=Name;
    cout<<"Constructor with length "<<name<<"\n";
    }
String::String(char *vs,char Name) { len=strlen(vs);
    str=new char[len+1];  strcpy(str,vs);  name=Name;
    cout<<"Constructor "<<name<<"\n";
    }
```

Файл S.cpp (2)

```
String::String(String &S)
    {
    len=S.Length();
    str=new char[len+1];
    strcpy(str,S.str);
    name='K';
       cout<<"Copy from "<<S.name<<" to "<<name<<"\n";
        }
String::~String()
        {
    delete [] str;
    cout<<"Destructor "<<name<<"\n";
     }
```

Файл S.cpp (3)

```
String  String::operator+(String &A)
      { cout<<"Operation +"<<"\n";   int j=len+A.Length();
        String S(j,'S');        strcpy(S.str,str);
        strcat(S.str,A.str);    cout<<"Operation +"<<"\n";
        return S;
    }
String  String::operator+(char c)
    { cout<<"Operation +c"<<"\n";  int j=len+1;
         String S(j,'Q');        strcpy(S.str,str);
      S.str[len]=c;      S.str[len+1]='\0';
         cout<<"Operation +c"<<"\n";  return S;
    }
```

Файл S.cpp (3)

```
String& String::operator=(String &S)
    {
    cout<<"Operation ="<<"\n";
   len=S.Length();
        if (str!=NULL) delete[]str;
        str=new char[len+1];
       strcpy(str,S.str);
       cout<<"Operation ="<<"\n";
    return *this;
        }
```

Тестирующая программа

```
#include "stdafx.h"
#include "s.h"

int main(int argc, char* argv[])
{   String A("ABC",'A'),B("DEF",'B'),C(6,'C');
    C.print();
    C=A+B;
    C.print();
     C=C+'a';
    C.print();
    return 0;
}
```

Выполнение операций

	Выполняемые операторы	Результаты

C=A+B;	String operator+(String &A)
	{ cout<<"Operation +"<<"\n";
	int j=len+A.Length();
	String S(j,'S');
	strcpy(S.str,str);
	strcat(S.str,A.str);
	cout<<"Operation +"<<"\n";
	return S;
	}
	String& operator=(String &S)
	{ cout<<"Operation ="<<"\n";
	len=S.Length();
	if (str!=NULL) delete[]str;
	str=new char[len+1];
	strcpy(str,S.str);
	cout<<"Operation ="<<"\n";
	return *this;
	}
		Operation +
		Constructor with length S
		Operation +
		Copy from S to K
		Destructor S
		Operation =
		Operation =
		Destructor K

5.16 Параметризованные классы (шаблоны)

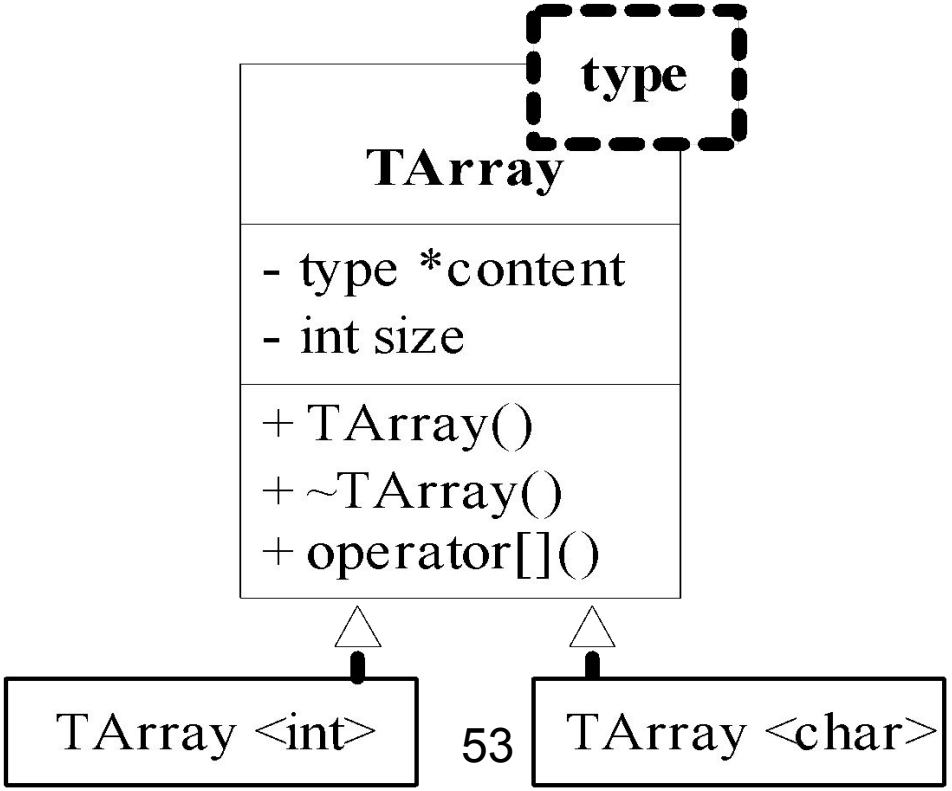
Формат описания шаблона класса:

```
template <Список параметров><Описание класса>
```

Формат объявления объектов:

```
<Имя класса> <Список аргументов>
      <Имя объекта>(<Параметры конструктора>)
```

Пример. Динамический массив (`Ex5_12`)



Файл A.h

```
#include <iostream.h>
template <class type>
class TArray
{   type * content;
    int size;
 public:
    TArray(int asize)
        { content = new type [size=asize];}
    ~TArray (){delete [] content;}
    type & operator[] (int x)
        { if ((x<0)||(x>=size))
                { cerr << "Index Error"; x=0; }
          return content[x];
        }
};
```

Тестирующая программа

```
#include "stdafx.h"
#include "A.h"
#include <stdio.h>

int main(int argc, char* argv[])
{int i;
 TArray<int> int_a(5);
 TArray<char> char_a(5);
 for (i=0;i<5;i++)
     { int_a[i]=i*3+2*(i+1); char_a[i]='A'+i;}
 puts("Massivs ");
 for (i=0;i<5;i++)
     { printf("%5d   %2c\n",int_a[i],char_a[i]);}
 return 0;
}
```

5.17 Параметризованные функции

Формат описания шаблона функции:

template <Список параметров><Описание функции>

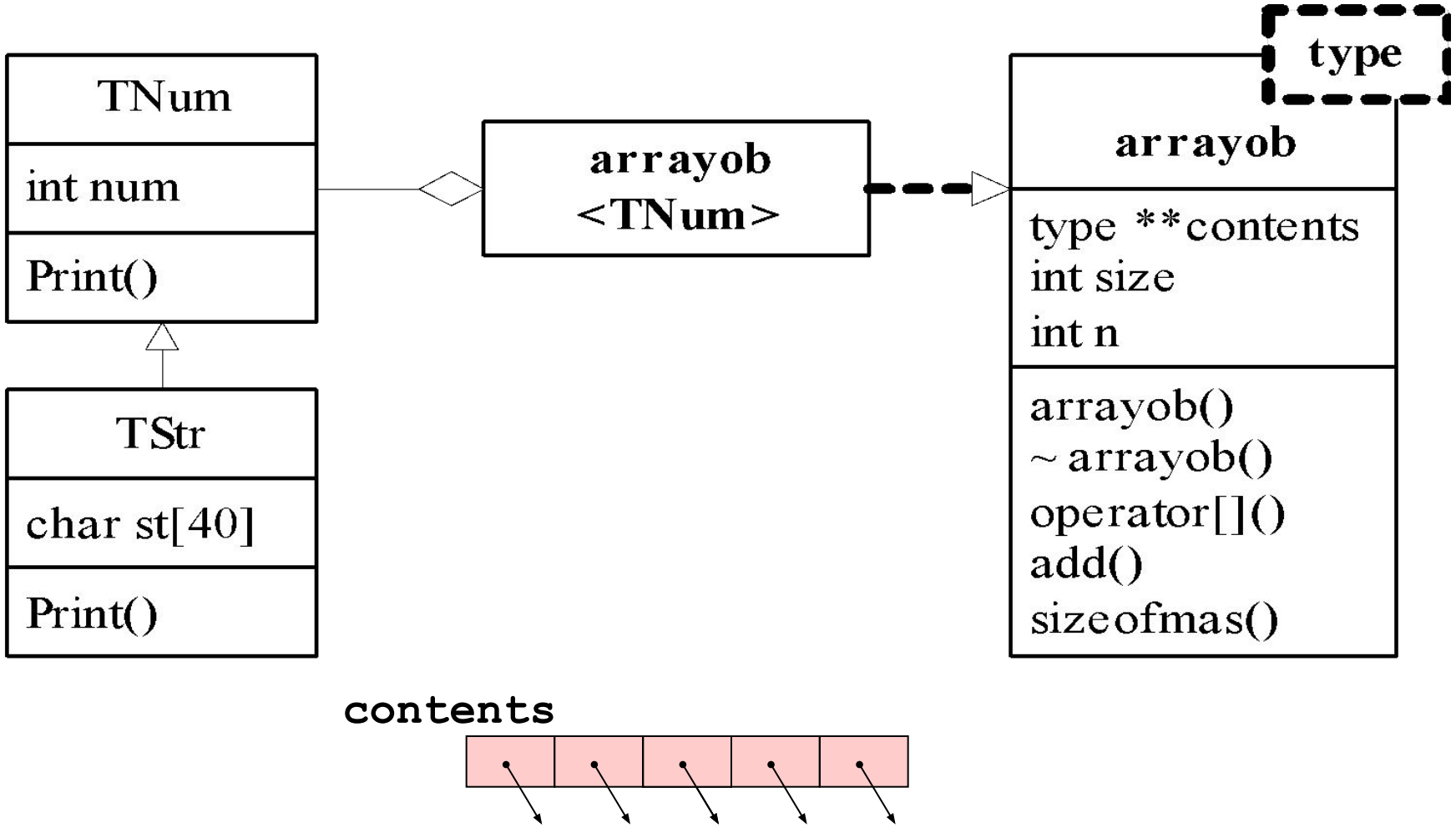
Пример. Шаблон функции определения максимального (Ex5_13)

```
#include "stdafx.h"
#include <string.h>
#include <iostream.h>
template <class T>
      T max(T x, T y){ return(x>y)?x:y;}
char * max(char * x, char * y)
      { return strcmp(x,y) > 0? x:y;}
```

Тестирующая программа

```
int main(int argc, char* argv[])
{   int a=1,b=2;
    char c='a', d='m';
    float e=123, f=456;
    double p=234.567,t=789.23;
    char str1[]="AVERO", str2[]="AVIER";

    cout << "Integer max=   "<<max(a,b)<< endl;
    cout << "Character max= "<<max(c,d)<< endl;
    cout << "Float max=     "<<max(e,f)<< endl;
    cout << "Double max=    "<<max(p,t)<< endl;
    cout << "String max=    "<<max(str,str2)<< endl;
    return 0;
}
```

Контейнер на основе шаблона (Ex5_14)

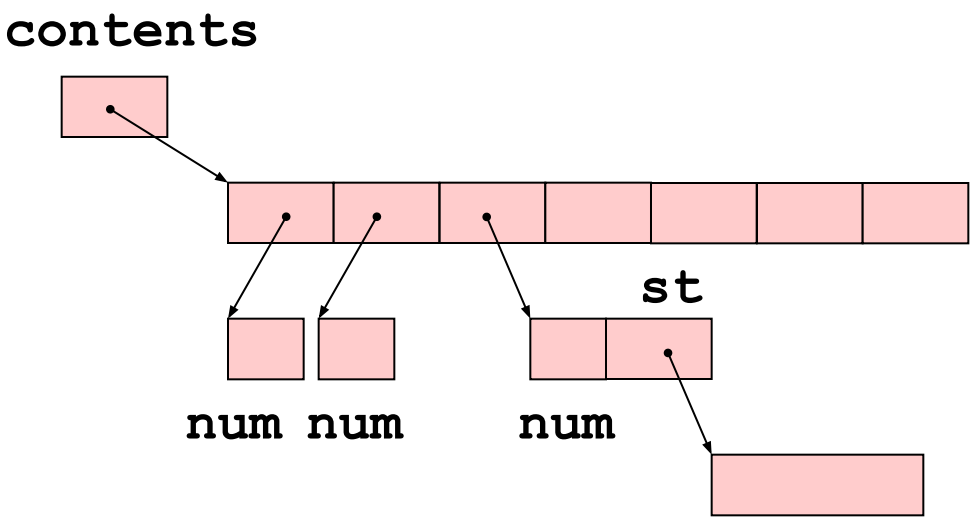


Объявление шаблона класса

```
#include <iostream.h>
template <class type>
class arrayob
{  type **contents;  int size;  int n;
 public:
  arrayob(int number){contents=new type *[size=number];}
  ~arrayob ();
  int sizeofmas(){return n;}
  void add(type *p) { if(n==size)cerr<<"Out of range";
                        else contents[n++]=p;
                     }
  type & operator [] (int x)
    { if  ((x<0)||(x>=size))
            { cerr <<"Error "<<x<<endl;x=0;}
      return *contents[x]; }
};
```

Объявление шаблона функции

```
template <class type>
   arrayob <type>::~arrayob ()
        {for(int i=0;i<size;i++) delete contents[i];
         delete [] contents;
        }
```



Файл N.h

```
#include <iostream.h>
#include <string.h>

class TNum
{ public:int num;
   TNum(int n):num(n) {}
   virtual ~TNum(void) {cout<<"Destructor TNum "<<endl;}
   virtual void Print(void) { cout<<num<<" "; }
};
class TStr:public TNum
{ public: char *st;
   TStr(char *s):TNum(strlen(s))
     {st=new char[num+1];strcpy(st,s);  st[num]='\0';   }
    ~TStr(void)
     { cout<<"Destructor TStr.";  delete [] st;}
   void Print(void)
     {TNum::Print(); cout<<st<<" ";}
};
```

Тестирующая программа

```
#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include "A.h"
#include "N.h"
arrayob <TNum>  ob_a(5);
int main(int argc, char* argv[])
{ int n,i;
  char S[10];
  for(i=0;i<5;i++)
   { puts("Input number or string");  gets(S);
     n=atoi(S);
     if (n==0&&S[0]=='0'||n!=0)
          ob_a.add(new TNum(n));
     else ob_a.add(new TStr(S));}
  cout<<" Contents of array"<<'\n';
  for (i=0;i<ob_a.sizeofmas();i++) ob_a[i].Print();
  return 0;
}
```