11. Шаблоны классов

11.1. Контейнеры

Контейнер – хранилище некоторой совокупности данных

Операции с контейнером не зависят от типа данных, размещенных в контейнере

Пример: стек. Операции со стеком:

- положить в стек
- взять из стека

11.2. Простой шаблон класса

// файл *file*.h

```
template <class MyType>
class Stack {
private:
    static const int SZ = 100;
    int top;
    MyType arr[SZ];
```

11.2. Простой шаблон класса
(продолжение)

```
public:
    Stack();
    int push(const MyType &);
    int pop(MyType &);
};

template <class Tp>
Stack<Tp>::Stack():top(0){}
```

11.2. Простой шаблон класса
(продолжение)

```
template <class T>
int Stack<T>::push(const T &el)
{
    if(top < SZ){
        arr[top++] = el;
        return 0;
    }
    return -1;
}
```

11.2. Простой шаблон класса
(продолжение)

```
template <class P>
int Stack<P>::pop(P &el)
{
    if(top > 0){
        el = arr[--top];
        return 1;
    }
    return 0;
}
```

11.3. Использование шаблона класса

```
// файл file.cpp
#include “file.h”

int main()
{
    Stack<int> st1, st2;
    st1.push(12); // в стеке число 12
    st1.push(25); // в стеке число 25
    st2 = st1;
```

11.3. Использование шаблона класса (продолжение)

```
    int x, y;
    st1.pop(x); // извлечение числа 25
    st1.pop(y); // извлечение числа 12
    cout << "x = " << x << ", y = " << y <<
   endl;
    return 0;
}
```

```
x = 25, y = 12
Для продолжения нажмите любую клавишу . . .
```

11.4. Вариант 1 шаблона класса

```
// файл  file.h

template <class MyType, int SZ>
class Stack {
private:
    int top;
    MyType arr[SZ];
```

11.4. Вариант 1 шаблона класса (продолжение)

```
public:
    Stack();
    int push(const MyType &);
    int pop(MyType &);
};

template <class Tp, int N>
Stack<Tp, N>::Stack():top(0){}
```

11.4. Вариант 1 шаблона класса (продолжение)

```
template <class T, int N>
int Stack<T, N>::push(const T &el)
{
    if(top < N){
        arr[top++] = el;
        return 0;
    }
    return -1;
}
```

11.4. Вариант 1 шаблона класса (продолжение)

```
template <class P, int N>
int Stack<P, N>::pop(P &el)
{
    if(top > 0){
        el = arr[--top];
        return 1;
    }
    return 0;
}
```

11.5. Использование шаблона класса

```
// файл file.cpp
#include “file.h”

int main()
{
    Stack<int, 100> st1;
    Stack<int, 25> st2;
    st1.push(12); // в стеке число 12
    st1.push(25); // в стеке число 25
```

11.5. Использование шаблона класса (продолжение)

```
    int x, y;
    st1.pop(x); // извлечение числа 25
    st1.pop(y); // извлечение числа 12
    cout << "x = " << x << ", y = " << y <<
   endl;
    return 0;
}
```

```
x = 25, y = 12
Для продолжения нажмите любую клавишу . . .
```

11.6. Вариант 2 шаблона класса

```
template <class T>
class Stack {
private:
    int top;
    T *arr;
    int sz;
public:
    Stack(int = 0);
```

11.6. Вариант 2 шаблона класса (продолжение)

```
    int push(const T &);
    int pop(T &);
    Stack(const Stack<T> &);
    ~Stack(){delete [ ] arr; }
    Stack<T> &operator =(const Stack<T> &);
};
```

11.6. Вариант 2 шаблона класса (продолжение)

```
template <class T>
int Stack<T>::push(const T &el)
{
    if(top < sz){
        arr[top++] = el;
        return 0;
    }
    return -1;
}
```

11.6. Вариант 2 шаблона класса (продолжение)

```
template <class P>
int Stack<P>::pop(P &el)
{
    if(top > 0){
        el = arr[--top];
        return 1;
    }
    return 0;
}
```

11.6. Вариант 2 шаблона класса (продолжение)

```
template <class T>
Stack<T>::Stack(int n):top(0)
{
    sz = n > 0 ? n : 10;
    arr = new T[sz];
}
```

11.6. Вариант 2 шаблона класса (продолжение)

```
template <class T>
Stack<T>::Stack(const Stack<T> &s):
     top(s.top), sz(s.sz), arr(new T[s.sz])
{
    for(int i = 0; i < top; ++i)
        arr[i] = s.arr[i];
}
```

11.6. Вариант 2 шаблона класса (продолжение)

```
template <class T>
Stack<T> &Stack<T>::operator =(
    const Stack<T> &s)
{
    if(this != &s){
        delete [] arr;
        top = s.top;
        sz = s.sz;
```

11.6. Вариант 2 шаблона класса (продолжение)

```
        arr = new T[sz];
        for(int i = 0; i < top; ++i)
            arr[i] = s.arr[i];
    }
    return *this;
}
```

11.7. Использование шаблона класса

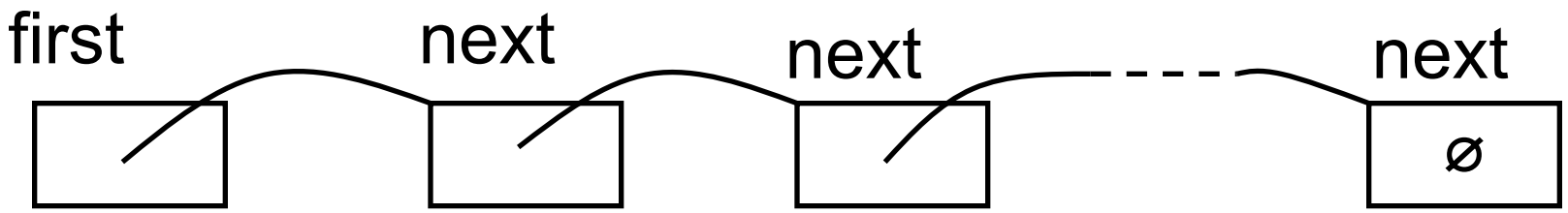
```
int main()
{
    Stack<int> st1, st2;
    st1.push(12);
    st1.push(25);
    st2 =st1;
    int x, y;
    st1.pop(x);
```

11.7. Использование шаблона класса (продолжение)

```
    st2.pop(y);
    cout << "x = " << x << ", y = " << y <<
   endl;
    return 0;
}
```

```
x = 25, y = 25
Для продолжения нажмите любую клавишу . . .
```

11.8. Контейнер: список




```
struct SLink {
    SLink *next;
    SLink(): next(NULL){ }
    SLink(SLink *p): next(p){ }
};
```

11.8. Контейнер: список
(продолжение)

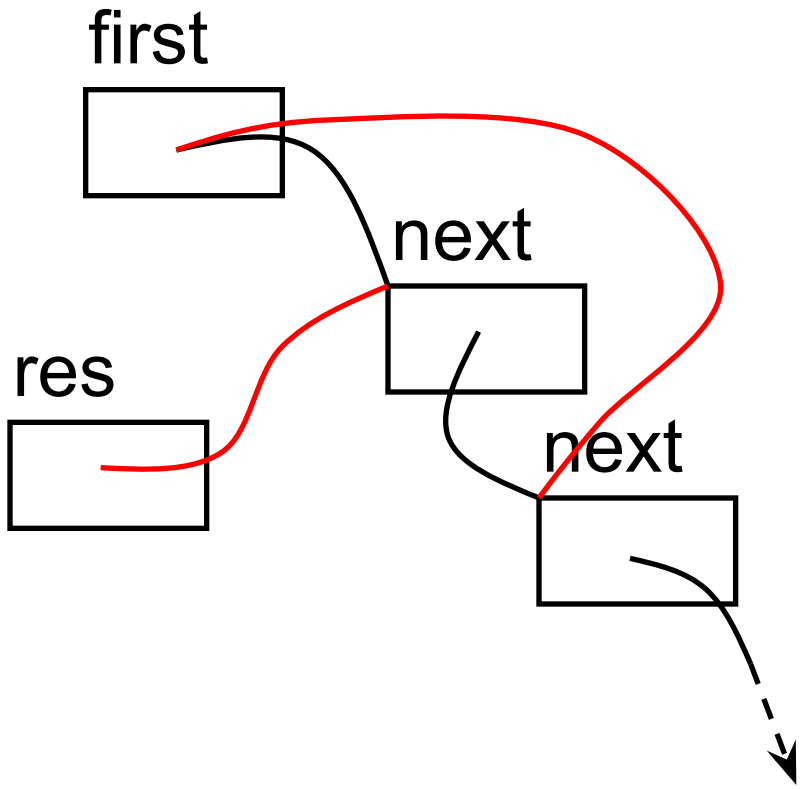
```
class BList {
protected:
    SLink *first;
public:
    BList(): first(NULL){ }
    BList(SLink *a): first(a){ }
    void insert(SLink *);
    SLink *get();
    friend class BListIter;
};
```

11.8. Контейнер: список
(продолжение)

```
class BListIter {
private:
    BList *cs;
    SLink *cl;
public:
    BListIter(BList &a):cs(&a), cl(a.first){ }
    SLink *operator()();
};
```

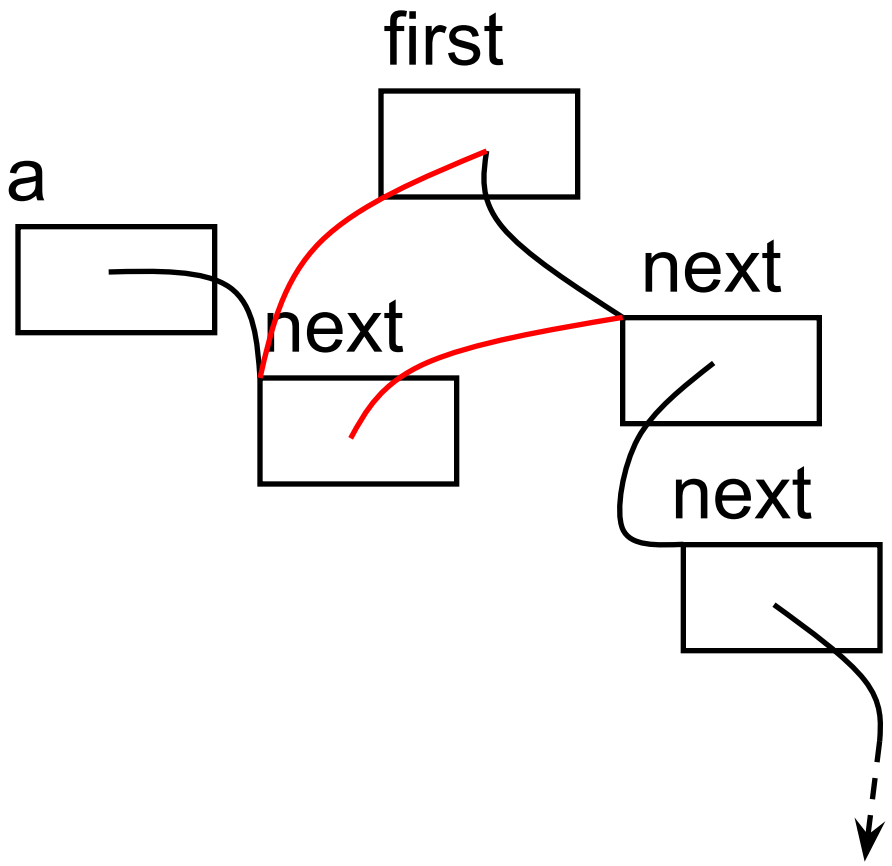
11.8. Контейнер: список
(продолжение)

```
SLink *BList::get()
{
    SLink *res = first;
    if(first){
        first = first->next;
        res->next = NULL;
    }
    return res;
}
```



11.8. Контейнер: список
(продолжение)

```
void BList::insert(SLink *a)
{
    a->next = first;
    first = a;
    a = NULL;
}
```

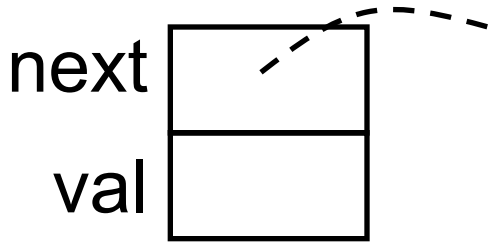


11.8. Контейнер: список
(продолжение)

```
SLink *BListIter::operator()()
{
    SLink *ret = cl;
    cl = cl ? cl->next : cs->first;
    return ret;
}
```

11.9. Контейнер: список – использование

```
class StackInt: public SLink{
protected:
    int val;
public:
    StackInt(int a = 0):val(a){}
    int getVal() const {return val;}
};
```



11.9. Контейнер: список – использование (продолжение)

```
int main()
{
    BList st;
    StackInt *p = NULL;
    st.insert(new StackInt(12));
    st.insert(new StackInt(25));
    st.insert(new StackInt(38));
```

11.9. Контейнер: список – использование (продолжение)

```
p = static_cast<StackInt *>( st.get() );
 std::cout << "item #3: " << (p->getVal())
<<
 std::endl;
```

11.9. Контейнер: список – использование (продолжение)

```
std::cout << "Iterator: ";
BListIter it(st);
SLink *pl;
while(pl = it()){
    p = static_cast<StackInt *>(pl);
    std::cout << (p->getVal()) << ' ';
}
std::cout << std::endl;
```

11.9. Контейнер: список – использование (продолжение)

```
    p = static_cast<StackInt *>(st.get());
    std::cout << "item #2: " << (p->getVal())
   <<
     std::endl;

    return 0;
}
```

11.9. Контейнер: список – использование (продолжение)

```
item #3: 38
Iterator: 25 12
item #2: 25
Для продолжения нажмите любую клавишу . . .
```

11.10. Шаблон списка

```
template<class T>
struct TLink: public SLink{
    T info;
    TLink(const T &a):info(a){ }
};
```

11.10. Шаблон списка
(продолжение)

```
template<class T>
class SListIter;

template<class T>
class SList:private BList {
    friend class SListIter<T>;
public:
    void insert(const T &a);
    T get();
};
```

11.10. Шаблон списка
(продолжение)

```
template <class T>
void SList<T>::insert(const T &a)
{
    BList::insert(new TLink<T>(a));
}
```

11.10. Шаблон списка
(продолжение)

```
template <class T>
T SList<T>::get()
{
    T res;
    TLink<T> *lnk =
    static_cast<TLink<T>*>(BList::get());
```

11.10. Шаблон списка
(продолжение)

```
    if(lnk){
        res = lnk->info;
        delete lnk;
    }
    return res;
}
```

11.10. Шаблон списка
(продолжение)

```
template <class T>
class SListIter:public BListIter{
public:
    SListIter(SList<T> &a): BListIter(a){ }
    T *operator()();
};
```

11.10. Шаблон списка
(продолжение)

```
template<class T>
T *SListIter<T>::operator()()
{
    SLink *p = BListIter::operator()();
    return p ?
    &(static_cast<TLink<T> *>(p))->info :
    NULL;
}
```

11.11. Использование шаблона списка

```
int main()
{
    SList<int> st;
    st.insert(12);
    st.insert(25);
    st.insert(38);
    int res = st.get();
    std::cout << "item #3: " << res <<
   std::endl;
```

11.11. Использование шаблона списка (продолжение)

```
SListIter<int> it(st);
int *p = NULL;
std::cout << "Iterator: ";
while(p = it())
    std::cout << (*p) << ' ';
std::cout << std::endl;
```

11.11. Использование шаблона списка (продолжение)

```
    std::cout << "item #2: " << st.get() <<
    std::endl;
    return 0;
}
```

```
item #3: 38
Iterator: 25 12
item #2: 25
Для продолжения нажмите любую клавишу . . .
```

11.12. Структура программы

Архивный файл mytempl

Файл	Описание
Shape.h	Определение классов: Point (точка) абстрактный базовый класс Shape (фигура) производный класс Circle (окружность) производный класс Rect (прямоугольник)
Shape.cpp	Реализация классов Point, Shape, Circle и Rect
Assoc_T.h	Определение и реализация шаблонов классов: Pair<IND, INF> – структура; элемент контейнера Assoc<IND, INF> – контейнер; ассоциативный массив AssocIt<IND, INF> – класс-итератор для контейнера
Appl_T.cpp	Тестирующая программа для отладки разработанных классов