

# Шаблони (Templates) (Rev: 1.4)

Любомир Чорбаджиев<sup>1</sup>  
lchorbadjiev@elsys-bg.org

<sup>1</sup>Технологическо училище “Електронни системи”  
Технически университет, София

2 май 2007 г.

# Съдържание

- 1 Нужда от шаблони
- 2 Дефиниране на шаблонен клас
- 3 Екземпляри на шаблона
- 4 Проверка на шаблона
- 5 Пример: шаблонен стек (статичен)
- 6 Шаблонни функции
- 7 Пример: шаблонен масив с проверка на границите
- 8 Пример: шаблонен стек (динамичен)

# Шаблони

- Шаблоните обезпечават непосредствената поддръжка на така нареченото *обобщено програмиране*, т.е. програмиране, при което като параметри се използват типове.
- Механизмът на шаблоните в C++ позволява използването на типове в качеството на параметри при дефинирането на функции и класове.
- Шаблонът зависи само от тези свойства на параметъра-тип, които той явно използва; поради това не е необходимо различните типове, които се използват като параметри на шаблона да бъдат свързани по какъвто и да било начин.

# Дефиниране на шаблон

```
template<class T> class stack {  
    T data_[128];  
public:  
    const T& pop(void) const;  
    //...  
};
```

- Префиксът `template<class T>` се използва за дефиниране на шаблон (`template`).
- При използване на даден шаблон, на мястото на “формалния параметър” `class T` се предава фактическият тип.
- В дефиницията на шаблона формалното име на тип `T` се използва точно по същия начин, по който се използват и имената на другите типове.

# Дефиниране на шаблон

```
template<class T> class stack {  
    T data_[128];  
public:  
    const T& pop(void) const;  
    //...  
};
```

- Областта на видимост за `T` завършва в края на обявата, започнала с `template<class T>`.
- В дефиницията `template<class T>` `T` е име на произволен тип; не е задължително `T` да бъде име на клас.

# Екземпляри на шаблона

```
stack<double> doubleStack;  
stack<int> intStack;
```

- Процесът на генериране на клас от (1) шаблон на клас и (2) аргумент на шаблона се нарича **създаване на екземпляр на шаблона (template instantiation)**.
- Генерирането на клас от шаблон на клас се изпълнява от компилатора.
- Класът, генериран от шаблон на клас, е обикновен C++ клас. Използването на шаблони не предполага допълнителни механизми по време на изпълнение на кода.
- Шаблоните обезпечават ефективен начин за генериране на код.

# Параметри на шаблона

- Като параметри на даден шаблон могат да се използват не само типове:

## Пример:

```
template<class T, int size> class Buffer {
    T data_[size];
    int size_;
public:
    Buffer(void) : size_(size)
    {}
    //...
};
```

# Проверка на типовете

- Проверка в точката на дефиниция: проверка за синтактични грешки и грешки, които не зависят от фактическите параметри-типове на шаблона.
- Проверка при създаване на екземпляр на шаблона: проверка за съответствие на фактическите типове, предадени на шаблона.
- Проверка в момента на свързване.



# Проверка на типовете: пример

```
1 template<class T> class stack {
2     T data_[128];
3     int top_;
4 public :
5     stack(void) : top_(-1) {}
6     //...
7     void print_all(void) {
8         for(int i=0;i<=top_;++i)
9             cout << data_[i] << '␣';
10        cout << endl;
11    }
12};
```

# Проверка на типовете: пример

- Да приемем, че за класа `Rec` не е дефиниран оператор за изход `operator<<(ostream& out, const Rec& r)`.
- Тогава екземплярът `recStack` на шаблона `stack<T>` дефиниран в ред 2 съдържа грешка в метода `print_all()`, тъй като този метод разчита елементите на стека да имат предефиниран оператор за изход.

```
1 class Rec { /* ... */ };  
2 stack<Rec> recStack; // ?? error;  
3 recStack.print_all(); // error;
```

# Пример: заглавен файл `stack.hpp`

```
1 #ifndef STACK_HPP__
2 #define STACK_HPP__
3
4 #include <exception>
5
6 template<class T>
7 class stack {
8     static const unsigned size_=128;
9     T data_[size_];
10    int top_;
11 public:
12    stack(void);
```

Пример: заглавен файл `stack.hpp`

```
13  const T& top(void) const ;
14  void pop(void) ;
15  void push(const T& val) ;
16  bool empty(void) const ;
17  };
18
19  template<class T>
20  stack<T>::stack(void)
21    : top_(-1)
22  {}
```

Пример: заглавен файл `stack.hpp`

```
23
24 template<class T> const T&
25 stack<T>::top(void) const {
26     if (top_ < 0) {
27         throw std::exception();
28     }
29     return data_[top_];
30 }
31 template<class T> void
32 stack<T>::pop(void) {
33     if (top_ < 0){
34         throw std::exception();
35     }
36     top_--;
37 }
```

Пример: заглавен файл `stack.hpp`

```
38 template<class T> void  
39 stack<T>::push(const T& val){  
40     if( size_ <= top_+1 ) {  
41         throw std::exception();  
42     }  
43     data_[++top_]=val;  
44 }  
45 template<class T> bool  
46 stack<T>::empty(void) const {  
47     return top_<0;  
48 }  
49 #endif
```

# Пример: използване на шаблонен стек

```
1 #include <iostream>
2 #include "stack.hpp"
3
4 int main(void) {
5     stack<int> si;
6
7     for(int i=0; i<10; ++i){
8         si.push(i);
9     }
10
11    while(! si.empty() ){
12        std::cout << si.top() << "␣";
13        si.pop();
14    }
15    std::cout << std::endl;
```

# Пример: използване на шаблонен стек

```
17 stack<float> sf;  
18 for(int i=0; i<10; ++i){  
19     sf.push(10.0*i);  
20 }  
21 while(! sf.empty() ){  
22     std::cout << sf.top() << "□";  
23     sf.pop();  
24 }  
25 std::cout << std::endl << std::endl;
```



# Пример: използване на шаблонен стек

```
27 stack<stack<int>> ssi;  
28 for(int i=0;i<5;++i){  
29     stack<int> temp;  
30     for(int j=0;j<10;++j){  
31         temp.push(i);  
32     }  
33     ssi.push(temp);  
34 }
```

# Пример: използване на шаблонен стек

```
36 while(!ssi.empty()){
37     stack<int> ts=ssi.top();
38     while(!ts.empty()){
39         std::cout << ts.top() << "␣";
40         ts.pop();
41     }
42     std::cout << std::endl;
43     ssi.pop();
44 }
45
46 return 0;
47 }
```

# Пример: стек

## Резултати:

```
9 8 7 6 5 4 3 2 1 0
90 80 70 60 50 40 30 20 10 0
```

```
4 4 4 4 4 4 4 4 4 4
3 3 3 3 3 3 3 3 3 3
2 2 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
```

# Шаблони на функции

- Механизмът на шаблоните може да се ползва за обобщено дефиниране на функции:

## Пример:

```
1 template<class R, class T>  
2 const R& fun(T& a){  
3     //...  
4 }
```

```
1 template<class T>  
2 void sort(vector<T>& v){  
3     //...  
4 }
```

# Шаблоны на функции

```
1 template <class T>
2 void swap(T& a, T& b) {
3     T tmp=a;
4     a=b;
5     b=tmp;
6 }
```

# Шаблони на функции

- При шаблоните на функции съществен момент се явява възможността за **извеждане (deduction)** на типа на аргументите на шаблона.

## Пример:

```
1 template<class T> T& fun(const T& val) {/*...*/}
2 int i=0, p=10;
3 i=fun(p);
4
5 template<class T> const T& fun1(void) {/*...*/}
6 int x=fun1(); //error
7 int y=fun1<int>();
```

# Шаблоны на функции

```
1 template<class R, class T> R fun2(T& v) {/*...*/}
2 int z=0;
3 double w=0;
4 w=fun2<double, int>(z);
5 w=fun2<double>(z);
6 w=fun2(z); // error!!
```

# Използване на шаблони: масив с проверка на границите

```
1 #include <iostream>
2 #include <exception>
3 using namespace std;
4
5 template<class T>
6 class Array {
7     unsigned int size_;
8     T* data_;
9 public:
10    Array(unsigned int size=10)
11        : size_(size), data_(new T[size_])
12    {}
```



# Използване на шаблони: масив с проверка на границите

```
13 Array(const Array& other)
14     : size_(other.size_), data_(new T[size_])
15     {
16         for(unsigned int i=0; i< size_; i++)
17             data_[i]=other.data_[i];
18     }
19 ~Array(void) {
20     delete [] data_;
21 }
22 unsigned size() const {
23     return size_;
24 }
```

# Използване на шаблони: масив с проверка на границите

```
25 Array& operator=(const Array& other) {  
26     if(this!=&other) {  
27         delete [] data_;  
28         size_=other.size_;  
29         data_=new T[size_];  
30         for(unsigned i=0;i<size_;i++)  
31             data_[i]=other.data_[i];  
32     }  
33     return *this;  
34 }
```

# Използване на шаблони: масив с проверка на границите

```
35 T& operator [] (unsigned int index)
36     throw (exception)
37 {
38     if (index >= size_) {
39         throw exception();
40     }
41     return data_[index];
42 }
43 };
```

# Използване на шаблони: масив с проверка на границите

```
44 int main(void) {  
45     Array<int> a1(3), a2;  
46     for(int i=0;i<3;++i) {  
47         a1[i]=i;  
48     }  
49     a2=a1;  
50     for(int i=0;i<3;i++) {  
51         cout << "a2[" << i << "]=" << a2[i] << endl;  
52     }
```

# Използване на шаблони: масив с проверка на границите

```
53 try {  
54     cout << "a2[" << 3 << "]=" << a2[3] << endl;  
55 } catch(exception &e) {  
56     cout << "exception caught..."  
57         << endl;  
58 }
```

# Използване на шаблони: масив с проверка на границите

```
60 Array<double> a3(3), a4;  
61 for(int i=0;i<3;++i) {  
62     a3[i]=(i+1)*3.14;  
63 }  
64 a4=a3;  
65 for(int i=0;i<3;i++) {  
66     cout << "a4[" << i << "]= " << a4[i] << endl;  
67 }
```

# Използване на шаблони: масив с проверка на границите

```
68 try {  
69     cout << "a3[" << 3 << "]=" << a3[3] << endl;  
70 } catch(exception &e) {  
71     cout << "exception caught..."  
72         << endl;  
73 }  
74  
75 return 0;  
76 }
```

# Използване на шаблони: масив с проверка на границите

```
lubo@dobby:~/school/cpp/notes> ./a.out  
a2[0]=0  
a2[1]=1  
a2[2]=2  
exception caught...  
a4[0]=3.14  
a4[1]=6.28  
a4[2]=9.42  
exception caught...
```



# Използване на шаблони: динамичен стек

```
1 #include <iostream>
2 #include <exception>
3 using namespace std;
4
5 template<class T>
6 class Stack {
7     const static unsigned int chunk_=2;
8     int size_;
9     T *data_;
10    int top_;
```

# Използване на шаблони: динамичен стек

```
11 public :  
12     Stack(void)  
13         : size_(chunk_),  
14           data_(new T[size_]),  
15           top_(-1)  
16     {}  
17     ~Stack(void) {  
18         delete [] data_;  
19     }
```

# Използване на шаблони: динамичен стек

```
20 Stack(const Stack& other)
21     : size_(other.size_),
22       data_(new T[size_]),
23       top_(other.top_)
24 {
25     for(int i=0; i<=top_; i++)
26         data_[i]=other.data_[i];
27 }
```

# Използване на шаблони: динамичен стек

```
28 Stack& operator=(const Stack& other) {
29     if (this != &other) {
30         delete [] data_;
31         size_ = other.size_;
32         top_ = other.top_;
33         data_ = new T[size_];
34         for (int i = 0; i <= top_; i++)
35             data_[i] = other.data_[i];
36     }
37     return *this;
38 }
```

# Използване на шаблони: динамичен стек

```
39 void push(const T& v) {  
40     if (top_ >= (size_ - 1)) {  
41         resize();  
42     }  
43     data_[++top_] = v;  
44 }  
45 T pop(void) {  
46     if (top_ < 0) {  
47         throw exception();  
48     }  
49     return data_[top_--];  
50 }
```

# Използване на шаблони: динамичен стек

```
51 private :  
52 void resize(void) {  
53     T *oldData=data_;  
54     data_=new T[size_+chunk_];  
55     for(int i=0;i<size_;i++)  
56         data_[i]=oldData[i];  
57     delete [] oldData;  
58     size_+=chunk_;  
59 }  
60 };
```

# Използване на шаблони: динамичен стек

```
61 int main(void) {  
62     Stack<int> st;  
63     st.push(1);  
64     st.push(2);  
65     st.push(3);  
66  
67     Stack<int> st1=st;  
68     cout << st.pop() << endl;  
69     cout << st.pop() << endl;  
70     cout << st.pop() << endl;  
71  
72     cout << st1.pop() << endl;  
73     cout << st1.pop() << endl;  
74     cout << st1.pop() << endl;
```

# Използване на шаблони: динамичен стек

```
76 try {  
77     cout << st1.pop() << endl;  
78 } catch(const exception& e) {  
79     cout << "exception caught..." << endl;  
80 }  
81 return 0;  
82 }
```



# Исползване на шаблони: динамичен стек

```
lubo@kid:~/school/cpp/notes$ ./a.out
3
2
1
3
2
1
exception caught...
```