1 非类型模板参数

1.1 非类型的模板参数

```
template  <typename T = int, int MAXSIZE = 100>
class Stack{
public:
    T elems[MAXSIZE];
};
    Stack<int, 20> int20stack;
    Stack<int, 40> int40stack;
    Stack<std::string, 40> str40stack;
    int20stack = int40stack;
    /*
     * 每个模板实列都有自己的类型，int20stack和int40stack也是不同类型，
     * 而且这种类型之间不存在显式或隐式类型转换，
     * 所以他们之间不能相互替换，更不能相互赋值。
     * */
```

1.2 非类型的函数模板参数

```
    void templates::nontype_template_parameters(){
    std::vector<int> src = {1,2,3,4,5};
    std::vector<int> dst(5);
    std::vector<int> dst_auto(5);
    std::transform(src.begin(), src.end(), dst.begin(),
            (int(*)(int const&))add_value<int, 5> );
    std::transform(src.begin(), src.end(), dst_auto.begin(),
            add_value<int, 5> );
    /*
     * 模板实列通常看成用来命名一组重载函数的集合（即使只有一个函数）。
     * 重载函数的集合不能被用于模板参数演绎。需要把这个函数模板的实参强制类型转为具体类型。
     * 目前C++11已经解了决这个问题，只有在考虑可移植的情况才需要使用这种强制转换。
     * */
}
```

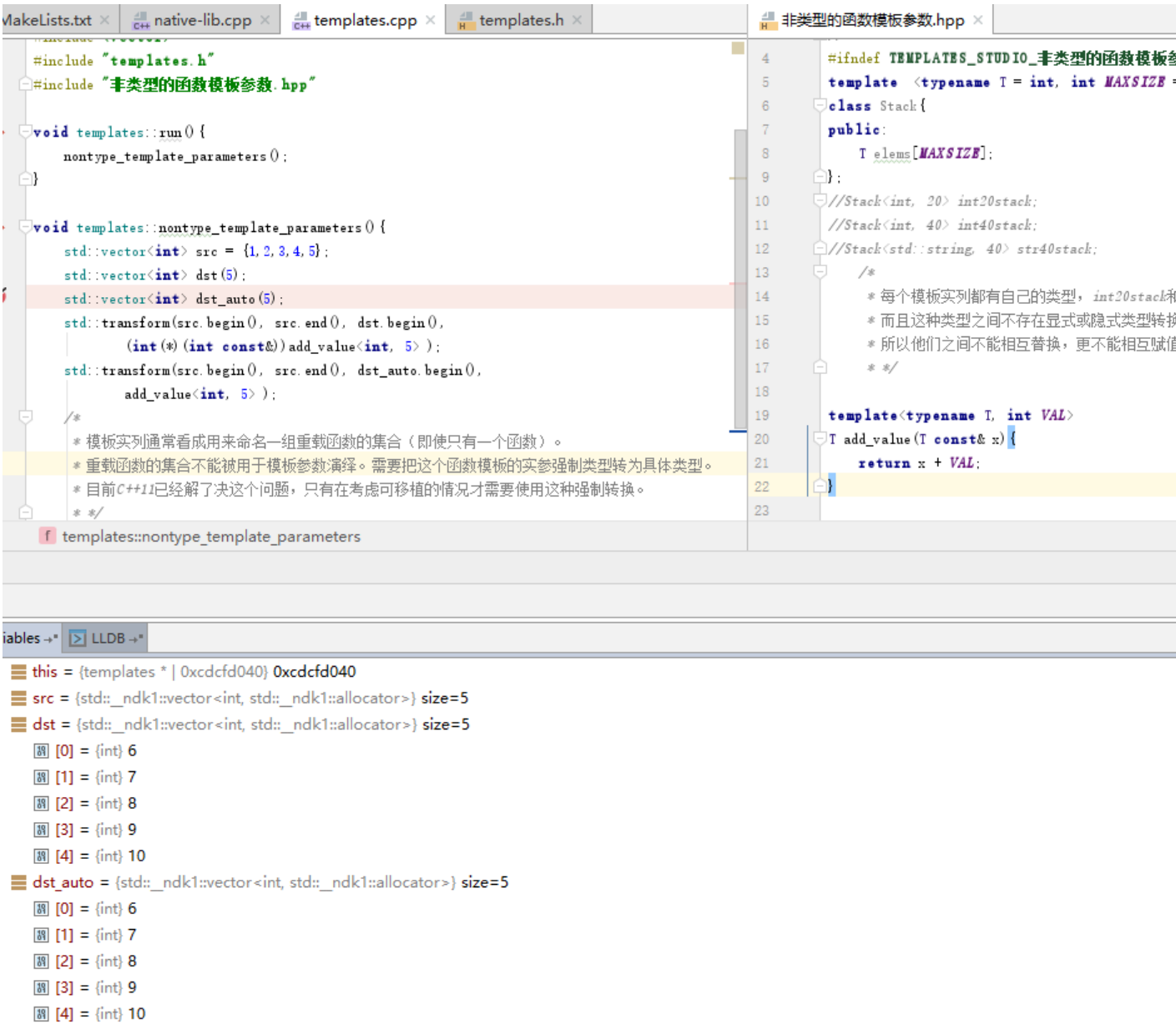

图 1: 非类型的函数模板参数

1.3 非类型模板参数的限制

非类型模板参数类型是有限制的，通常是常整数，枚举值，或者指向外部链接对象的指针。
浮点数应为历史原因不能作为非类型模板参数，以后可能会支持。
字符串文字是内部链接对象，类对象更不能作为非类型模板参数。

```
template<double VAT>
double process(double v){
    return v * VAT;
}
// error: a non-type template parameter cannot have type '
   double'

template<std::string name>
class TemStr{
public:
    std::string _str = name;
};
// error: a non-type template parameter cannot have type 'std::
   string'

template<char const* name>
class TemPtr{
public:
    std::string _str = name;
};
// extern char const* sa = "test";//ERROR
// error: non-type template argument of type 'const char *' is
   not a constant expression
// extern char const sa[] = "test";//OK

//浮点和string直接不使用也会编译error.
//全局字符数组是一个外部链接对象，可以作为非类型模板参数。
```