计算机学院 高级程序语言设计 课程实验报告

实验题目：模板特化、模板元编程		学号：202300130150
日期：2024.5.7	班级： 4	姓名： 王成意

实验目的：

1. 培养通用性泛型编程能力。
2. 练习模板特化、模板元编程

实验步骤与内容：

1. 模板特化：实践第 9 章 PPT，P93-94，对例 9-8，Stack 栈类模板进行偏特化，使得以下测试程序可以测试运行。注意，PPT 中给出的偏特化的模板目前声明了 7 个 public 函数，但是只有 push 和 pop 两个函数有定义，目前的代码是不能正常编译的。请问这是因为我们没有定义哪个函数造成的？按照上面的模板对这个函数进行定义，然后运行以下程序。在 sz 的值是 100 的时候打印的 sizeof(stk)是多少？为什么？

   代码：

```
#include <cassert>

template <class T, int SIZE = 50>
class Stack;
template <int SIZE>  //可变参数
class Stack<bool, SIZE> { //bool固定
private:
    enum {
        UNIT_BITS = sizeof(unsigned) * 8,
        ARRAY_SIZE = (SIZE - 1) / UNIT_BITS + 1
    };
    unsigned list[ARRAY_SIZE];
    int top;
public:
    Stack();
    void push(bool item);
    bool pop();
    void clear();
    bool peek() const;
    bool isEmpty() const;
    bool isFull() const;
};
template <int SIZE>
void Stack<bool, SIZE>::push(bool item) {
    assert(!isFull());
```

```
    int index = ++top / UNIT_BITS;
    list[index] = (list[index] << 1) | (item ? 1 : 0);
}

template <int SIZE>
bool Stack<bool, SIZE>::pop() {
    assert(!isEmpty());
    int index = top-- / UNIT_BITS;
    bool result = ((list[index] & 1) == 1);
    list[index] >>= 1;
    return result;
}



#include "stack.h"
#include <iostream>

using namespace std;
#define sz 100

int main() {
    Stack<bool, sz> stk;
    cout << "size of Stack<bool, 100>: " << sizeof(stk) << endl;
    for (int i = 0; i < sz; i++) {
        stk.push((bool)(i % 2));
    }
    for (int i = 0; i < sz; i++) {
        cout << i << "\t" << stk.pop() << endl;
    }
    return 0;
}
```

请问这是因为我们没有定义哪个函数造成的？

```
ublic: __cdecl Stack<bool,100>::Stack<bool,100>(void)" (??
ublic: bool __cdecl Stack<bool,100>::isEmpty(void)const "
ublic: bool __cdecl Stack<bool,100>::isFull(void)const " (
ist.exe : fatal error LNK1120: 3 个无法解析的外部命令
```

，可见是没有定义 isFull，isEmpty，及 Stack 的构造函数造成的。

在 sz 的值是 100 的时候打印的 sizeof(stk)是多少？为什么？

```
size of Stack<bool, 100>: 104
```

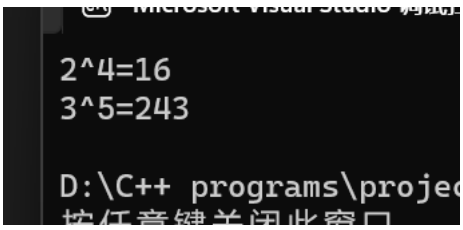
原因：100 个 bool 和一个 int，占用 104 字节。

2. 模板元编程：参照第 9 章 PPT，P100 例编写 Power.h,使得以下 main 函数可以测试运行。

代码：

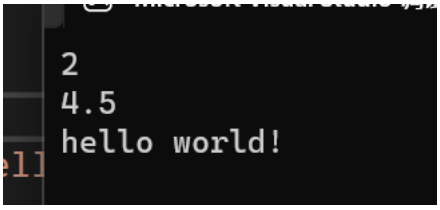
```
template <unsigned N>
struct Power { //类模板
    template <class T>
    static T value(T x) {//函数模板
        return x * Power<N - 1>::value(x);
    }
};

template <>
struct Power<1> { //特化 N=1
    template <class T>
    static T value(T x) {
        return x;
    }
};

//可以用Power<4>::value(x)来计算x的4次方
template <unsigned N, class T>
inline T power(T v) {//辅助函数模板
    return Power<N>::value(v);
}


#include "Power.h"
#include <iostream>
using namespace std;

int main() {
    int v = Power<4>::value(2);
    cout << "2^4=" << v << endl;
    v = power<5>(3);
    cout << "3^5=" << v << endl;
}
```

结果：



3. PPT 104 页的代码可以帮助我们理解可变参数的传递，去掉 func 函数定义参数中的 const T &t 之后会出现什么样的编译错误？为什么会有这个错误？在函数调用的时候怎么做就不会出现这个错误了？：

运行结果：



去掉 func 函数定义参数中的 const T &t 之后会出现什么样的编译错误？

```
, 配置: Debug x64 ------

LinkList\test2.cpp(14,2): error C2672: “func”: 未找到匹配的重载函数
LinkList\test2.cpp(8,6): message : 可能是“void func(const TypeArgs &...)”
LinkList\test2.cpp(14,2): message : “void func(const TypeArgs &...)”: 无法推导“T”的 模板 参数
 的操作 - 失败。
```

为什么会有这个错误？

原因：T 模板是模糊的，定义了但是没用上。

在函数调用的时候怎么做就不会出现这个错误了？

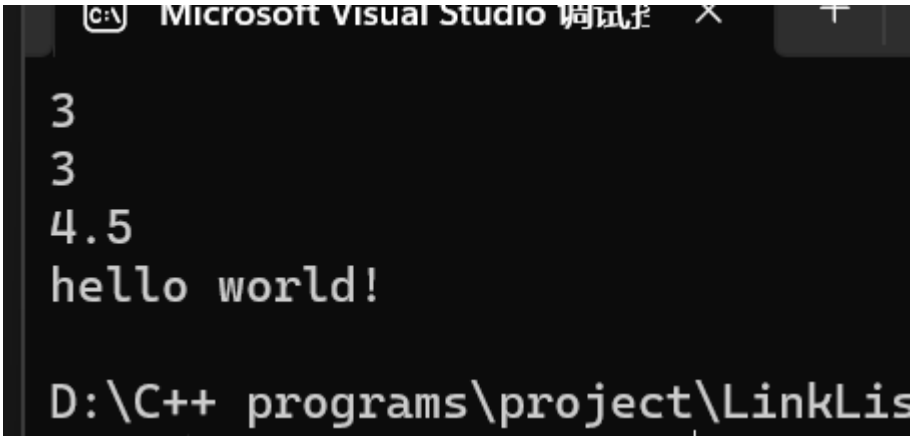
改成：

```
template <typename... TypeArgs>
```

代码：

```
#include<iostream>
#include "test2.h"
template <typename T>
void print(T& t) { //另一函数
    std::cout << t << std::endl;
}

template <typename... TypeArgs>
void func(const TypeArgs&... args) {
    std::cout << sizeof...(args) << std::endl; //参数个数
    (print(args), ...);  //折叠表达式，对args的每个参数调用print
}

int main() {
    func(3, 4.5, "hello world!");
    return 0;
}
```

结果：



4. 练习习题 9-21，模板特化，函数模板重载 。

原结果：

新增：

```
template <class T>
void bubbleSort(T *a[], int n) {
    int i = n - 1;
    while (i > 0) {
        int lastExchangeIndex = 0;
        for (int j = 0; j < i; j++)
            if (*a[j + 1] < *a[j]) {
                mySwap(*a[j], *a[j + 1]);
                lastExchangeIndex = j;
            }
        i = lastExchangeIndex;
    }
}
```

结果：

(2)在解决问题(1)时，如果采取了对 bubbleSort 函数模板重载的办法，将使得起泡排序的主要代码被书写两遍，造成了代码的冗余，请思考如何解决这一问题。

实现方法：套壳。代码：

```
template <class T>
void bubbleSort(T a[], int n) {
    int i = n - 1;
    while (i > 0) {
        int lastExchangeIndex = 0;
        for (int j = 0; j < i; j++)
            if (a[j + 1] < a[j]) {
                mySwap(a[j], a[j + 1]);
                lastExchangeIndex = j;
            }
        i = lastExchangeIndex;
    }
}


template <class T>
void bubbleSort(T *a[], int n) {
    bubbleSort(a, n);
  }
```

5. 请利用模板元编程设计一个类模板 template<unsigned M, unsigned N> Permutation，内含一个枚举值 VALUE，Permutation<M, N>::VALUE 的值为排列数 PMN(从 N 个值中选 M 个值有多少种可能性)。请参考排列数的定义以及在第九章课件第 98 页给出的通过模板元编程实现阶乘来实现上述类模板。main 函数如下

```
template <unsigned long long N>

struct Factorial {

    constexpr static long long value = N * Factorial<N

- 1>::value;
```

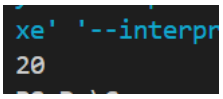
```
};
template<>
struct Factorial<0> {
    constexpr static long long value = 1;
};
```

```
#include"1.h"
template<unsigned M, unsigned long long N>
struct Permutation
{
    constexpr static int val = Factorial<N>::value /
(Factorial<N - M>::value * Factorial<M>::value);
};
```

```
#include"2.h"
#include<iostream>
using namespace std;
int main()
{
    cout << Permutation<3,6>::val;
    return 0;
```

```
}
```

实测 25 会炸精度，用 6 代替：



9：

结论分析与体会：非常好的实验，使我重新认识了模板元编程和提高了培养通用性泛型编程能力。