STL

Тема 6. Адаптеры для указателей функций

Указатель на унарную функцию

```
template <class Arg, class Result>
class pointer_to_unary_function : public unary_function <Arg,Result>
{
 protected:
        Result(*pfunc)(Arg);
 public:
        explicit pointer_to_unary_function ( Result (*f)(Arg) ) : pfunc (f)
        { }
        Result operator() (Arg x) const
        {
                return pfunc(x);
        }
};
```

Указатель на бинарную функцию

```
template <class Arg1, class Arg2, class Result>
class pointer_to_binary_function : public binary_function <Arg1,Arg2,Result>
{
protected:
        Result(*pfunc)(Arg1,Arg2);

public:
        explicit pointer_to_binary_function ( Result (*f)(Arg1,Arg2) ) : pfunc (f)
         {}
        Result operator() (Arg1 x, Arg2 y) const
        {
                return pfunc(x,y);
        }
};
```

Указатель на бинарную функцию

```
int main ()
{
pointer_to_binary_function <double,double,double> PowPointer (pow);
double numbers[] = {1.0, 2.0, 3.0, 4.0, 5.0};
double squares[5];
transform (numbers, numbers+5, squares, bind2nd(PowPointer,3) );
for (int i=0; i<5; i++)
        cout << squares[i] << " ";   cout << endl;
return 0;
}
```

Указатель на функцию

```
template <class Arg, class Result> pointer_to_unary_function<Arg,Result>
ptr_fun (Result (*f)(Arg))
{
        return pointer_to_unary_function<Arg,Result>(f);
}

template <class Arg1, class Arg2, class Result>
pointer_to_binary_function<Arg1,Arg2,Result> ptr_fun (Result (*f)(Arg1,Arg2))
{
        return pointer_to_binary_function<Arg1,Arg2,Result>(f);
}
```

Указатель на функцию

```
int main( )
{
  vector <char*> v1;
  vector <char*>::iterator RIter;
  v1.push_back ( "I" );   v1.push_back ( "love" );
  v1.push_back ( "C++" ); v1.push_back ( "lecture" );
  v1.push_back ( "!" );
  RIter = find_if( v1.begin( ), v1.end( ),not1 ( bind2nd (ptr_fun ( strcmp ), "C++" ) ) );
 if ( RIter != v1.end( ) )
 {
    cout << "Found a match: " << *RIter << endl;
 }
}
```

Указатель на функцию класса

```
template <class S, class T>
class mem_fun_t : public unary_function <T*,S>
{
        S (T::*pmem)();
  public:
        explicit mem_fun_t ( S (T::*p)() ) : pmem (p)
        {}

        S operator() (T* p) const
        {
                return (p->*pmem)();
        }
};
```

Указатель на функцию класса

```
template <class S, class T, class A>
class mem_fun1_t : public binary_function <T*,A,S>
{
        S (T::*pmem)(A);
  public:
        explicit mem_fun1_t ( S (T::*p)(A) ) : pmem (p)
        {}

        S operator() (T* p, A x) const
        {
                return (p->*pmem)(x);
        }
};
```

Указатель на функцию класса

```
template <class S, class T> class const_mem_fun_t :
public unary_function <T*,S>

template <class S, class T, class A> class mem_fun1_t :
public binary_function <T*,A,S>
```

Указатель на функцию класса

```
template <class S, class T> mem_fun_t<S,T> mem_fun (S (T::*f)())
{ return mem_fun_t<S,T>(f);  }

template <class S, class T, class A>
mem_fun1_t<S,T,A> mem_fun (S (T::*f)(A))
{ return mem_fun1_t<S,T,A>(f); }

template <class S, class T> const_mem_fun_t<S,T>
mem_fun (S (T::*f)() const)
{ return const_mem_fun_t<S,T>(f); }

template <class S, class T, class A> const_mem_fun1_t<S,T,A> mem_fun
(S (T::*f)(A) const)
{ return const_mem_fun1_t<S,T,A>(f); }
```

Указатель на функцию класса

```
int main ()
{
 vector <string*> numbers;
 numbers.push_back ( new string ("one") );
 numbers.push_back ( new string ("two") );
 numbers.push_back ( new string ("three") );
 numbers.push_back ( new string ("four") );
 numbers.push_back ( new string ("five") );

 vector <int> lengths ( numbers.size() );
 transform (numbers.begin(), numbers.end(), lengths.begin(), mem_fun(&string::length));
 for (int i=0; i<5; i++)
                { cout << *numbers[i] << " has " << lengths[i] << " letters.\n"; }
 return 0;
}
```

Указатель на функцию класса

```
template <class S, class T>
class mem_fun_ref_t : public unary_function <T,S>
{
        S (T::*pmem)();
  public:
        explicit mem_fun_ref_t ( S (T::*p)() ) : pmem (p)
        {}

        S operator() (T& p) const
        {
                return (p.*pmem)();
        }
};
```

Указатель на функцию класса

```
template <class S, class T, class A>
class mem_fun1_ref_t : public binary_function <T,A,S>

template <class S, class T>
class const_mem_fun_ref_t : public unary_function <T,S>

template <class S, class T, class A>
class mem_fun1_ref_t : public binary_function <T,A,S>
```

mem_fun_ref

Указатель на функцию класса

```
int main ()
{
 vector <string> numbers;
 numbers.push_back ("one" );
 numbers.push_back ( "two");
 numbers.push_back ("three");
 numbers.push_back ( "four") ;
 numbers.push_back ( "five" );

 vector <int> lengths ( numbers.size() );
 transform (numbers.begin(), numbers.end(), lengths.begin(),
                                                    mem_fun_ref(&string::length));
 for (int i=0; i<5; i++)
          { cout << numbers[i] << " has " << lengths[i] << " letters.\n"; }
 return 0;
}
```

Практическое задание

1. Дан массив произвольных целых чисел. Получить массив пар – частное, остаток, от деления этих чисел на 5
2. Дан текст. Удалит из него все слова длинна которых меньше 3
3. Дан список городов. Требуется получить их аббревиатуры состоящие из первых трех букв.
4. Для к классу студента функцию вывода на экран и осуществить вывод всех студентов
5. Добавить к классу студента функцию возвращающую отличник ли он и вывести количество не отличников