**ЛЗ 6. Решение задач по теме Ссылки**

Выполнил : Величко Валентин Александрович. П1-17 30.04.20

[1] Упражнения раздела 2.5

**1)** **Упражнение 2.24.** Какие из следующих определений (если они есть) недопустимы? Почему? Как их исправить?

(a)int ival = 1.01;

(b) int &rvall = 1.01;

(с)int &rval2 = ival;

(d) const int &rval3 = 1;

(a) int ival = 1.01; - Допустимо, но скомпилируется без сотой части. Можно для этого использовать тип double.

(b) int &rvall = 1.01; - Недопустимо: ссылка может ссылаться только на переменную, а не на значение.

(с) int &rval2 = ival; - Допустимо.

(d) const int &rval3 = 1; - Недопустимо: ссылка может ссылаться только на переменную, а не на значение.

**2) Упражнение 2.25.** Какие из следующих присвоений (если они есть) являются недопустимыми (с учетом предыдущих определений)? Если таковые есть, объясните, почему.

(а)rva12 = 3.14159;

(b) rva12 = rva13;

(с)ival = rval3;

(d) rval3 = ival;

Недопустимо: d (rval3 - константа)

**3) Упражнение 2.26.** В чем разница между определениями (а) и присвоениями (Ь)?

Какие из них (если есть) недопустимы?

(a)int ival = 0; (b) ival = ri;

(c)const int &ri = 0;(d) ri = ival;

Недопустимы: a - ссылку надо инициализировать объектом

**4)** Что отобразит на экране следующий код?

int i, &ri = i; i = 5; ri = 10;

std::cout << i << " " << ri << std::endl;

10 10

[1] Упражнения раздела 4.2.3

**5) Упражнение 4.15.** Укажите принципиальные различия между указателями и ссылками.

1.Указатель может быть переназначен любое количество раз, в то время как ссылка после привязки не может быть перемещена на другую ячейку памяти.

2. Указатели могут указывать "в никуда" (быть равными NULL), в то время как ссылка всегда указывает на определенный объект.

3. Мы не можем получить адрес ссылки, как можете это делать с указателями.

4. Не существует арифметики ссылок, в то время как существует арифметика указателей. Однако есть возможность получить адрес объекта, указанного по ссылке, и применить к этому адресу арифметику указателей

[1] Упражнения раздела 7.2.2

**6) Упражнение 7.7.** Объясните различие между параметрами в следующих двух объявлениях.

void f(Т);

передается копия значения переменной

void f(Т&);

передается адрес переменной

**7)Упражнение 7.8**. Приведите пример случая, когда параметр должен быть ссылочным.

Приведите пример случая, когда параметр не должен быть ссылочным.

void swap(int &a, int &b)

{

int k = a;

a = b;

b = k;

}

int max(int x, int y)

{

if(x > y)

return x;

else

return y;

}

**8) Упражнение 7.10.** Следующая функция хоть и вполне допустима, но менее полезна,

чем могла бы быть. Выявите и устраните ограничения.

bool test(string& s) { return s.empty(); } Написать main для аргументации своего решения

#include <iostream>

using namespace std;

bool test (string& s)

{

return s.empty();

}

int main()

{

cout << test("hello"); //выдаст ошибку, т к параметр string& s-объект класса string

}

Константная ссылка может ссылаться сразу на значение; передам параметр в виде константной ссылки

#include <iostream>

using namespace std;

bool test ( const string& s)

{

return s.empty();

}

int main()

{

cout << test("hello"); //не выдаст ошибку

}

**9) Упражнение 7.11.** Когда ссылочные параметры должны быть константными?

Какие проблемы могут возникнуть, если параметр является обычной ссылкой,

а какие — если он является константной ссылкой?

Не константная ссылка может быть связана только с не константным объектом такого же типа (например, нельзя передать строковые литералы в функцию). Если в функцию необходимо передать большие объекты, которые не должны изменяться, то параметры определяются как константные ссылки, т.е. передача немодифицированных параметров константными ссылками.