## Контрольная работа, Ассоциативные контейнеры STL

### **Тема: Контейнеры STL и модульное тестирование**

Цель: Сформировать практические навыки разработки абстракций данных на основе контейнеров STL и модульного тестирования средствами Visual Studio.

### **Задание**

Реализовать абстрактный тип данных «Множество» в соответствии с вариантом задания и со спецификацией приведённой ниже. Протестировать его, используя средства модульного тестирования Visual Studio. Тестовые наборы необходимо построить на основе критериев тестирования C0,C1,C2 в зависимости от варианта задания.

### **Рекомендации к выполнению**

1. Абстракцию данных реализуйте, используя классы языка программирования и библиотеку шаблонов STL.
2. Для тестирования разработанного класса используйте средства модульного тестирования Visual Studio.

### **Выбор варианта контрольной работы.**

### Номер варианта контрольной работы соответствует двум последним цифрам вашего пароля, если эти две цифры образуют число меньшее или равное 24. Если это число больше 24, то номер вашего варианта вы получите вычитанием из него числа 24. Например, две последние цифры пароля 08, тогда номер вашего варианта будет 8.

### Например, две последние цифры пароля 25, тогда номер вашего варианта будет 25 – 24 = 1.

### **Варианты задания**

Варианты контрольной работы представлены в таблице 1. Вариант определяет тип значений помещаемых во множество и критерии тестирования разработанного множества и способ реализации разрабатываемого класса на основе шаблона.

Таблица 1 - Варианты контрольной работы

|  |  |  |  |
| --- | --- | --- | --- |
| №Варианта | Тип | Критерий тестирования | Реализация |
| 1 | int | С0,C1 | Агрегирование |
| 2 | long | С0,C1 | Агрегирование |
| 3 | string | С0,C10 | Агрегирование |
| 4 | char | С0,C1 | Агрегирование |
| 5 | int | C1,C2 | Агрегирование |
| 6 | long | C1,C2 | Агрегирование |
| 7 | string | C1,C2 | Агрегирование |
| 8 | char | C1,C2 | Агрегирование |
| 9 | int | C0,C3 | Агрегирование |
| 10 | long | C0,C3 | Агрегирование |
| 11 | string | C0,C3 | Агрегирование |
| 12 | char | C0,C3 | Агрегирование |
| 13 | int | С0,C1 | Наследование |
| 14 | long | С0,C1 | Наследование |
| 15 | string | С0,C10 | Наследование |
| 16 | char | С0,C1 | Наследование |
| 17 | int | C1,C2 | Наследование |
| 18 | long | C1,C2 | Наследование |
| 19 | string | C1,C2 | Наследование |
| 20 | char | C1,C2 | Наследование |
| 21 | int | C0,C3 | Наследование |
| 22 | long | C0,C3 | Наследование |
| 23 | string | C0,C3 | Наследование |
| 24 | char | C0,C3 | Наследование |

### Задание

1. В соответствии с приведенной ниже спецификацией реализуйте шаблон классов «множество», используя шаблон классов set библиотеки STL. Для тестирования в качестве параметра шаблона T выберите тип в соответствии с вариантом задания.
2. Протестируйте абстракцию данных, используя средства модульного тестирования Visual Studio.

#### Спецификация типа данных «множество»

**ADT** tset

**Данные**

Множества - это изменяемые неограниченные наборы элементов типа T. Содержимое множества изменяется следующими операциями:

* Опустошить (опустошение множества);
* Добавить (добавление элемента во множество);
* Удалить (извлечение элемента из множества).

Множество поддерживает следующую дисциплину записи и извлечения элементов: элемент может присутствовать во множестве только в одном экземпляре, при извлечении выбирается заданный элемент множества и удаляется из множества.

**Операции**

Операции могут вызываться только объектом «множество» (тип tset), указатель на который передаётся в них по умолчанию. При описании операций этот объект в разделе «Вход» не указывается.

Таблица 2 - Описание операций на ADT tset.

|  |  |
| --- | --- |
| Наименование Операции | Описание |
| ***Конструктор*** |  |
| Начальные значения: | Нет. |
| Процесс: | Создаёт пустое множество элементов типа T. |
|  | |
| ***Опустошить*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Удаляет из множества все элементы. |
| Выход: | Нет. |
| Постусловия: | Множество - пусто. |
|  | |
| ***Добавить*** |  |
| Вход: | d – элемент типа Т. |
| Предусловия: | Нет. |
| Процесс: | Добавляет d во множество, если в нем нет такого элемента. |
| Выход: | Нет. |
| Постусловия: | Множество содержит элемент d. |
|  | |
| ***Удалить*** |  |
| Вход: | d – элемент типа Т. |
| Предусловия: | Нет. |
| Процесс: | Удаляет элемент d из множества, если d принадлежит множеству. |
| Выход: | Нет. |
| Постусловия: | Множество не содержит элемент d. |
|  | |
| ***Пусто*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Определяет, содержит ли множество элементы. Возвращает значение True, если множество не пусто, False – в противном случае. |
| Выход: | Булевское значение. |
| Постусловия: | Нет. |
|  | |
| ***Принадлежит*** |  |
| Вход: | d – элемент типа Т. |
| Предусловия: | Нет. |
| Процесс: | Определяет, принадлежит ли элемент d множеству. Возвращает True, если d принадлежит множеству, False - в противном случае. |
| Выход: | Булевское значение. |
| Постусловия: | Нет. |
|  | |
| ***Объединить*** |  |
| Вход: | Множество q. |
| Предусловия: | Нет |
| Процесс: | Создаёт множество, полученное в результате объединения множества с множеством q. |
| Выход: | Множество. |
| Постусловия: | Нет. |
|  | |
| ***Вычесть*** |  |
| Вход: | Множество q. |
| Предусловия: | Нет. |
| Процесс: | Создаёт множество, полученное в результате вычитания из множества множество q. |
| Выход: | Множество. |
| Постусловия: | Нет. |
|  | |
| ***Умножить*** |  |
| Вход: | Множество q. |
| Предусловия: | Нет. |
| Процесс: | Создаёт множество, являющееся пересечением множества с множеством q. |
| Выход: | Множество. |
| Постусловия: | Нет. |
|  | |
| ***Элементов*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Подсчитывает и возвращает количество элементов во множестве, если множество пустое - ноль |
| Выход: | Целое - количество элементов во множестве. |
| Постусловия: | Нет. |
|  | |
| ***Элемент*** |  |
| Вход: | j - номер элемента множества. |
| Предусловия: | Нет. |
| Процесс: | Обеспечивает доступ к элементу множества для чтения по индексу j так, что если изменять j от 1 до количества элементов во множестве, то можно просмотреть все элементы множества. |
| Выход: | Элемент множества типа Т. |
| Постусловия: | Множество не модифицируется |

***end tset***

### Рекомендации к выполнению

1. Тип данных реализуйте, используя ассоциативный контейнер множество (set).
2. Шаблон классов «множество» реализуйте одним из двух способов в соответствии с вариантом:

наследуя от ассоциативного контейнера set, как представлено на рисунке 1:



**Рисунок 1 -** Шаблон классов «множество»

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include <set>

#include <string>

//--------------------------------------------------------------

using namespace std;

class MySet:set<int, less<int> >

{

public:

MySet();

void Insert(int a);

void OutPut();

//MySet operator\* (MySet b);

private:

};

MySet::MySet()

{

}

void MySet::Insert(int a)

{

insert(a);

}

void MySet::OutPut()

{

// Вывод элементов множества.

set<int, less<int> >::iterator iter;

iter = begin();

while (iter != end())

cout << \*iter++ << endl;

cout << endl;

}

//MySet MySet::operator\* (MySet b)

//{

//}

int \_tmain(int argc, \_TCHAR\* argv[])

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

MySet a = MySet();

MySet b = MySet();

MySet c = MySet();

a.Insert(3);

a.Insert(1);

a.Insert(7);

a.Insert(7);

b.Insert(8);

b.Insert(9);

b.Insert(7);

//c = a\*b;

a.OutPut();

b.OutPut();

c.OutPut();

return 0;

}

и агрегированием, описав поле типа ассоциативный контейнер set<T> класса «множество», в разрабатываемом классе, как показано на рисунке 2:



Рисунок 2 - Ассоциативный контейнер set<T> класса «множество»

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include <set>

#include <string>

//--------------------------------------------------------------

using namespace std;

class MySet

{

public:

MySet();

void Insert(int a);

void OutPut();

//MySet operator\* (MySet b);

private:

set<int, less<int> > S;

};

MySet::MySet()

{

S = set<int,less<int> >();

}

void MySet::Insert(int a)

{

S.insert(a);

}

void MySet::OutPut()

{

// Вывод элементов множества.

set<int, less<int> >::iterator iter;

iter = S.begin();

while (iter != S.end())

cout << \*iter++ << endl;

cout << endl;

}

//MySet MySet::operator\* (MySet b)

//{

//}

int \_tmain(int argc, \_TCHAR\* argv[])

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int z;

MySet a = MySet();

MySet b = MySet();

MySet c = MySet();

a.Insert(3);

a.Insert(1);

a.Insert(7);

a.Insert(7);

b.Insert(8);

b.Insert(9);

b.Insert(7);

//c = a\*b;

a.OutPut();

b.OutPut();

c.OutPut();

return 0;

}

Для обработки объектов с помощью множества tset, пользователю необходимо будет вместо идентификатора T подставить тип его объектов.

1. Тип данных реализуйте в отдельном файле utset.

### Порядок выполнения

Реализуйте заданную абстракцию данных в режиме консольного приложения:

1. Создайте консольное приложение и сохраните его под именем pset.
2. Добавьте к консольному приложению файл и сохраните его под именем uset.
3. В модуле uset опишите шаблон классов tset.
4. Подключите файл к консольному приложению с помощью директивы #include.
5. Разработайте тестовый набор данных для тестирования операций, заданных на множестве. Тестовый набор поместите в таблицу следующего вида (таблица 3):

Таблица 3 - Тестовый набор для тестирования шаблона классов «множество».

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тестовый набор для тестирования операции Сложить множества целых чисел | | | | |
| Номер теста | Исходные данные | | Ожидаемый результат | |
| Вход | Множество | Возвращаемое значение | Множество |
| 1 | () | () | () | () |
| 2 | (0) | () | (0) | () |
| 3 | (1) | (0) | (1 0) | (0) |
| 4 | (1 0) | (1 0) | (1 0) | (1 0) |
| 5 | (1 2 3) | (3 4 5) | (1 2 3 4 5) | (3 4 5) |

1. Протестируйте разработанную абстракцию данных с помощью средств модульного тестирования Visual Studio.

### Контрольные вопросы

1. В чём состоит сущность критерия C0?
2. В чём состоит сущность критерия C1?
3. В чём состоит сущность критерия C2?
4. Что такое УПГ?
5. Что такое путь в УПГ?
6. Что такое ветвь УПГ?
7. В каком файле описан ассоциативный контейнер set?
8. Что означает имя iterator в области видимости ассоциативного контейнера set?
9. Назначение метода insert() ассоциативного контейнера set?
10. Назначение метода erase() ассоциативного контейнера set?
11. Назначение метода empty() ассоциативного контейнера set?
12. Назначение метода clear() ассоциативного контейнера set?
13. Назначение метода size() ассоциативного контейнера set?
14. Назначение метода find() ассоциативного контейнера set?
15. В чём особенности абстрактных методов?
16. В чём особенности ассоциативный контейнер в отличие от последовательных контейнеров?
17. Как пользоваться реализованными вами шаблонами классов «множество»?

### Содержание отчета

1. Задание.
2. Текст программы.
3. Тестовый набор данных.

## Литература

1. Руководство по программированию на C# [Электронный ресурс]  URL: https://metanit.com/cpp/tutorial/1.1.php (дата обращения 20.03.18).