МНОЖЕСТВА (SETS)

1. Определения

http://www.cplusplus.com/reference/unordered_set/unordered_set/

Множеството е неподредена съвкупност от различни елементи

A = {3, 6, 7}

Мощност на множество е броят на елементите на дадено множество

|A| = 3

Празно множество е множество, в което няма елементи

E = { }

2. Основни операции над множества

2.1. **Обединението** на две множества е множество, съдържащо елементите и на двете множества.

A = {3, 6, 7}

B = {7, 9, 89, 888}

A U B = {3, 6, 7, 9, 89, 888}

2.2. **Сечението** на две множества е множество, съдържащо елементите, които принадлежат и на двете множества.

A = {3, 6, 7}

B = {7, 9, 89, 888}

A ∩ B = {7}

2.3. Проверка дали дадено множество е **празно множество**.

2.4. Проверка дали даден елемент **е елемент на** (принадлежи на) множество.

3. Реализация на множества в C/C++

- **Побитов вектор (Bit(wise) arrays) (C/C++)**

Използва се масив от 1/0 или true/false.

Нека множеството A да е множество, в което може да има някое от числата от 0 до 9. Например A = {3, 6, 7}

индекс	0	1	2	**3**	4	5	**6**	**7**	8	9
	0	0	0	**1**	0	0	**1**	**1**	0	0

Празното множество може да бъде представено чрез

Index	0	1	2	3	4	5	6	7	8	9
	0	0	0	0	0	0	0	0	0	0

Обединение на две множества

A = {3, 6, 7}

Index	0	1	2	**3**	4	5	**6**	**7**	8	9
	0	0	0	**1**	0	0	**1**	**1**	0	0

B = {3, 5, 7, 9}

Index	0	1	2	**3**	4	**5**	6	**7**	8	**9**
	0	0	0	**1**	0	**1**	0	**1**	0	**1**

A U B = {3, 5, 6, 7, 9}

Index	0	1	2	**3**	4	**5**	**6**	**7**	8	**9**
	0	0	0	**1**	0	**1**	**1**	**1**	0	**1**

Сечение на две множества

A = {3, 6, 7}

Index	0	1	2	**3**	4	5	**6**	**7**	8	9
	0	0	0	**1**	0	0	**1**	**1**	0	0

B = {3, 5, 7, 9}

Index	0	1	2	**3**	4	**5**	6	**7**	8	**9**
	0	0	0	**1**	0	**1**	0	**1**	0	**1**

A ∩ B = {3, 7}

Index	0	1	2	**3**	4	5	6	**7**	8	9
	0	0	0	**1**	0	0	0	**1**	0	0

- **STL C++ (множества - sets, мултимножества - multisets(bag))**

<set> от **Standard Template Library**

Деклариране на променлива S, която е множество

```
set <int> S;
```

Деклариране на променлива it – итератор за множества от цели числа.

```
set <int> :: iterator it;
```

Функции (Модификатори)

Функции	**Описание**
insert(element)	Вмъква element в множеството
empty()	Проверява дали множеството е празно
find(element)	Проверява дали елементът е от множеството

Функции	Описание
	и връща итератор към него
erase(element)	Изтрива element от множеството
clear()	Изтрива всички елементи от множество

При реализацията на <set> се използва структура, в резултат от употребата, на която елементите се съхраняват подредени. Това е полезно в повечето случаи. Ако искаме да използвам елементите по реда на въвеждането им може да използваме **<unordered_set>.**

http://www.cplusplus.com/reference/unordered_set/unordered_set/

Използването на **<multiset>** вместо **<set>** позволяа в множеството да има елементи с повторение.

- **Bitset**

Bitset е структура от STL, която е проектирана да съхранява битове (0 или 1). Тя е подобна на масив, но е оптимизирана по отношение на използваната памет: всеки елемент заема само **един бит**.

Възможна е реализация на множество с „побитов вектор“, вместо с масив от 0/1 с <bitset>.

Създаване на структура Biset

```
bitset <10> b;                    // празен bitset b
bitset <10> bs (string("01011")); // инициализация от стринг
```

Достъп до битовете

```
b[0], b[1], …
```

Функции **операции над битовете**	**Описание**
set() set(pos, val)	Задава 1ци на всички битове Задача стойност val на позиция bit
reset()	Задава 0ли на всички битове
flip()	Обръща стойността на битовете: 0ли в 1ци и обратно.

Functions **bitset operations**	**Description**
count()	Връща броя на единиците в bitset
test(pos)	Връща true/false, ако бит на позиция pos в bitset е 1.
any()	Проверява дали всеки бит е 1
none()	Проверява дали всеки бит е 0

+ Bitwise operations

знак	име	**пример**	
		операция	**резултат**
~	отрицание negation	**~** 01010010	10101101
&	и and	01010010 **&**11001001	01000000
|	или or	01010010 **|** 11001001	11011011
^	изключващо или xor	01010010 **^** 11001001	10011011
>>	Shift right	01010010 **>>** 3 01010010 **>>** -3	00001010 10010000
<<	Shift left	01010010 **<<** 3 01010010 **<<** -3	10010000 00001010

Hash_set

Hash_set е разширение на STL и помага да се съхраняват и откриват бързо данни от колекция, в която стойностите на елементите са уникални и служат за ключове.

4. Размити множества (Fuzzy sets)

Размитите множества са представени едновременно от Lotfi A. Zadeh и Dieter Klaua през 1965.

Размитите множества са множества, чиито елементи имат степен на принадлежност (между 0 и 1).

http://forum.codecall.net/topic/44067-implementing-a-fuzzy-set/

http://jfuzzylogic.sourceforge.net/html/java.html

5. Непресичащи се множества (Disjoint sets)

Две множества са непресичащи се, ако нямат общи елементи.

Например: {1, 2, 3} и {4, 5, 6} са непресичащи се множества.

В компютърните науки се използва структура „непресичащи се множества“, която се използва в алгоритмите за графи.

Типични операции върху непресичащи се множества са:

MAKE-SET(x), UNION(x, y), FIND-SET(x)

Литература

1. http://www.cplusplus.com
2. Anany Levitin, Introduction to the Design and Analysis of Algorithms
3. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, Introduction to Algorithms
4. Steve S. Skiena, The Algorithm Design Manual
5. G. Momcheva, lecture notes, etc.

Recommended environments for coding

Codeblocks - www.codeblocks.org - (! A portable verson also exists)

http://compileonline.com/

19.01.2013