***Лабораторная работа 1.***

***Множества. Работа с битовой маской.***

Рассмотрим некоторые алгоритмы на множествах.

*Алгоритм построения булеана (генерации всех подмножеств).*

Пусть задано некоторое конечное множество , мощности |*A| = n*. Мощность булеана множества *A* равна: . Поставим каждому подмножеству  множества *A* двоичный код  длины *n*, определяемый следующим образом:

, 

Число таких двоичных *n*-разрядных кодов также равно . Таким образом, *мы однозначно закодировали каждое подмножество множества* *A* двоичным кодом длины *n* (установили *взаимно однозначное соответствие*).При этом пустому множеству ∅ соответствует код *00…0*, подмножеству, содержащему только первый элемент множества *A* – *10…0*, подмножеству, содержащему элементы  – *0101…0*, и, соответственно, самому множеству *A* код- *11…1*. Будем трактовать двоичный код как число в двоичной системе счисления, которому соответствует целое десятичное число в диапазоне , например, коду *00…11* соответствует число *3*. Генерируя целые числа указанного диапазона, и представляя их в двоичном виде, мы тем самым генерируем кодируемые  подмножеств булеана множества *A*.

При программной реализации множество удобно представлять в виде массива из *n* элементов. Генерируя, например, число *3*, мы ставим ему в соответствие подмножество, состоящее из элементов массива *a[0]*, *a[1]*.

Для построения булеана может использоваться *алгоритм построения бинарного кода Грэя*, при котором каждое последующее подмножество получается из предыдущего путем прибавления или удаления одного элемента.

***Алгоритм «слияние» построения объединения двух множеств.***

В приведенном алгоритме для простоты пояснения исходные и результирующее множество задаются массивами. Вам необходимо задать их упорядоченными списками.

#include <conio.h>

#include <stdio.h>

//пример алгоритма объединения множеств методом слияния на массивах

/\*

суть алгоритма заключается в поочередном сравнении элементов из двух

множеств (начиная с младших) и записи наименьшего из сравниваемых

элементов в новое множество. В множестве из которого мы записали элемент

мы передвигаемся на следующий элемент. В случае когда сравнение дает

равенство элементов, мы записываем в новое множество один элемент

( на выбор из любого из двух множеств) и передвигаемся на следующий элемент

сразу в двух множествах. Эти действия мы производим пока не закончится

одно из множеств, после чего в новое множество мы дописываем элементы

из оставшегося множества.

В итоге получаем множество как объединение двух первоначальных множеств.

\*/

int main()

{

int a=7; //количество элементов множества А

int b=11; //количество элементов множества В

int am[100]={2,3,5,6,8,9,11};// массив с элементами упорядоченного множества А

int bm[100]={1,2,3,4,5,6,8,9,22,23,24};// массив с элементами упорядоченного множества В

int cm[200];// массив с элементами множества С (объединение А и В)

int i=0,j=0,k=0;

while( i!= a && j!= b) // пока не закончатся элементы одного из множеств

{

int data;

if (am[i] < bm[j])

{

data=am[i];

i++;

}

else

if(am[i] > bm[j])

{

data=bm[j];

j++;

}

else

{

data=am[i]; //можно date=bm[j]

i++;

j++;

}

cm[k]=data;

k++;

}

if(i==a && j==b) ;

else // в случае необходимости дописываем

{ // оставшиеся элементы

if(i!=a)

{

while (i!=a)

{

cm[k]=am[i];

k++;

i++;

}

}

if(j!=b)

{

while (j!=b)

{

cm[k]=bm[j];

k++;

j++;

}

}

}

for(i=0;i<k;i++) printf(" %d ",cm[i]);

getch();

return 0;

}

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

***Задание.***

На универсуме *U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10,11}* заданы множества *A, B, C*

Варианты заданий указаны в *таблице 1*. Для указанных множеств:

1. Для заданного множества *A* построить булеан.
2. Реализовать *на выбор* *сортировку слиянием* или объединение *слиянием*. При сортировке слиянием на вход подается *неупорядоченное множество (неотсортированный массив)*. Выход – *упорядоченное множество* (отсортированный по возрастанию массив). При объдинении сливаем два *упорядоченных массива*.
3. Реализовать *все* операции над множествами:объединение, пересечение, разность, симметрическую разность, дополнение, используя *битовую* *маску*.
4. Вычислить программно , используя битовую маску, выражение согласно варианту, проиллюстрировать результат диаграммой *Эйлера-Венна*.
5. Указать для номера вашего варианта (порядок генерации) *бинарный код Грея*.

*Таблица 1- Варианты заданий 1.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Множества* | | | *Выражение* |
| *A* | *B* | *C* |
| 1. | {1,2,3,4,5,6} | {5,6,7,8} | {2,3,5,7,8} |  |
| 2, | {2,3,4,5} | {3,4,6,7} | {4,8,9} |  |
| 3. | {2,3,4,5,6} | {3,4,6,7} | {3,5,8,9} |  |
| 4. | {1,2,3,4,5} | {4,5,6,7} | {3,5,7,8} |  |
| 5. | {1,3,4,7} | {3,5,6,7,8} | {2,4,5,7} |  |
| 6. | {2,4,5,8} | {2,3,6,8,9} | {1,4,7,8,9} |  |
| 7. | {1,2,6,9} | {2,3,5,6,7} | {2,3,5,6,8} |  |
| 8. | {1,2,4,7,8} | {2,4,5,6,8} | {3,4,5,8,9} |  |
| 9. | {1,2,3,4,6,9} | {3,4,5,8,9} | {5,6,7,9} |  |
| 10. | {1,3,5,6,7,8} | {2,3,4,5,8} | {1,2,3,5,6} |  |
| 11. | {1,3,5,6,7,8} | {2,3,5,7,9 | {3,4,6,7} |  |
| 12. | {1,3,4,7,8,9} | {2,4,6,8} | {1,3,4,7.8,9} |  |
| 13. | {2,3,4,5,6} | {3,4,6,7} | {3,5,8,9} |  |
| 14. | {1,3,5,6,7,8} | {2,3,4,5,8} | {1,2,3,5,6} |  |
| 15. | {1,2,3,4,5,6} | {5,6,7,8} | {2,3,5,7,8} |  |
| 16. | {1,2,3,4,5} | {4,5,6,7} | {3,5,7,8} |  |
| 17. | {1,2,3,4,5,6} | {5,6,7,8} | {2,3,5,7,8} |  |
| 18. | {2,4,5,8} | {2,3,6,8,9} | {1,4,7,8,9} |  |