

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»
(БГТУ им.В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем

Лабораторная работа №1.1
дисциплина: Дискретная математика
тема: «Операции над множествами»

Выполнил: ст. группы ПВ-201
Машуров Дмитрий Русланович
Проверил: Бондаренко Т.В.

Белгород 2020

Лабораторная работа №1.1

«Операции над множествами»

Цель работы: изучить и научиться использовать алгебру подмножеств, изучить различные способы представления множеств в памяти ЭВМ, научиться программно реализовывать операции над множествами и выражения в алгебре подмножеств.

Задания

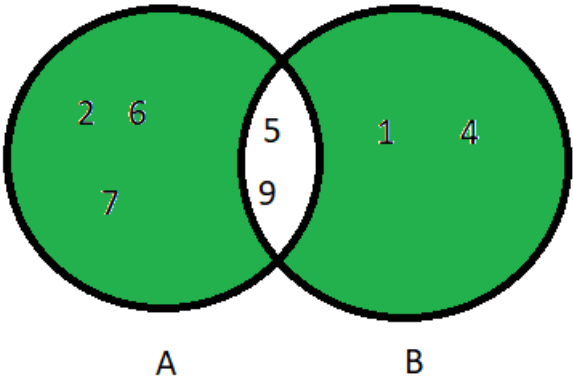
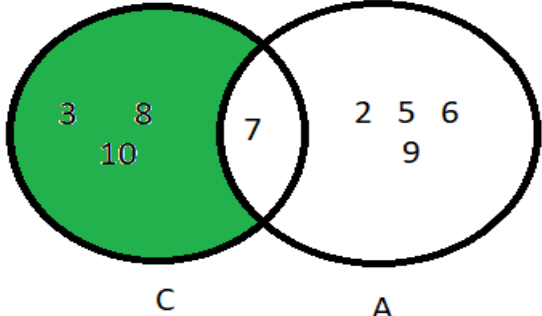
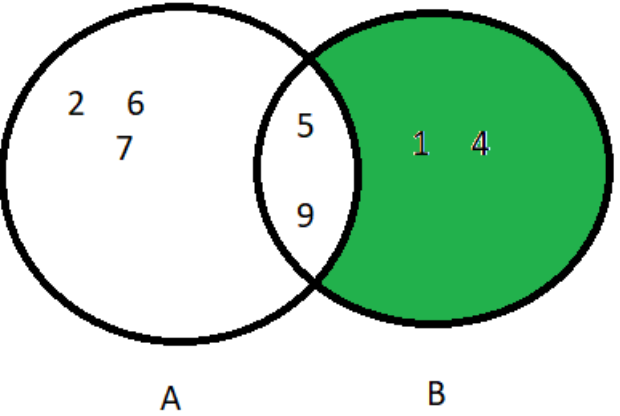
1. Вычислить значение выражения (см. Варианты заданий, п. а). Во всех вариантах считать $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Решение изобразить с помощью кругов Эйлера.
2. Записать выражение в алгебре подмножеств, значение которого при заданных множествах A , B и C равно множеству D (см. Варианты заданий, п. б).
3. Программно реализовать операции над множествами, используя следующие способы представления множества в памяти ЭВМ:
 - а) элементы множества A хранятся в массиве A . Элементы массива A неупорядочены;
 - б) элементы множества A хранятся в массиве A . Элементы массива A упорядочены по возрастанию;
 - в) элементы множества A хранятся в массиве A , элементы которого типа `boolean`. Если $i \in A$, то $A_i = \text{true}$, иначе $A_i = \text{false}$.
4. Написать программы для вычисления значений выражений (см. Задания, п.1 и п.2).
5. Используя программы (см. Задания, п.4), вычислить значения выражений (см. Задания, п.1 и п.2).

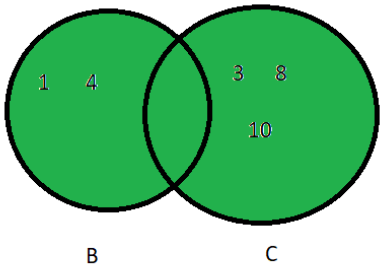
Задание варианта №2

- а) $D = B \cap (A \Delta B) \cup (C - A)$, $A = \{2, 5, 6, 7, 9\}$, $B = \{1, 4, 5, 9\}$, $C = \{3, 7, 8, 10\}$
- б) $A = \{1, 2, 3, 8\}$ $B = \{3, 6, 7\}$ $C = \{2, 3, 4, 5, 7\}$ $D = \{1, 3, 4, 5, 6, 8\}$

Выполнение:

1. Выполню вычисление выражения $D = B \cap (A \triangle B) \cup (C - A)$, где $A = \{2, 5, 6, 7, 9\}$, $B = \{1, 4, 5, 9\}$, $C = \{3, 7, 8, 10\}$ по действиям:

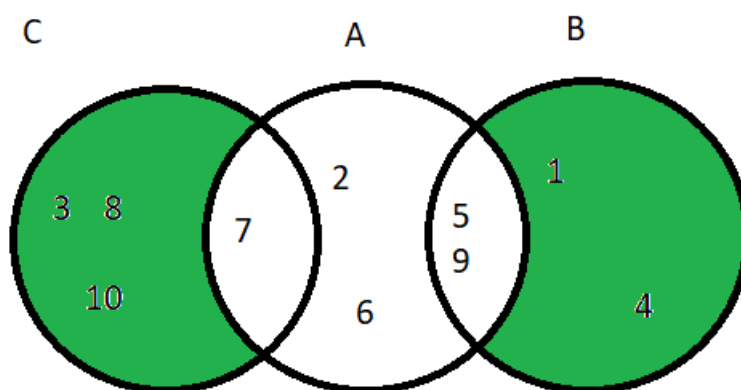
№	Действие	Графическое отображение в виде кругов Эйлера
1	$A = \{2, 5, 6, 7, 9\}$ $B = \{1, 4, 5, 9\}$ $A \triangle B = \{1, 2, 4, 6, 7\}$	 <p style="text-align: center;">A B</p>
2	$C = \{3, 7, 8, 10\}$ $A = \{2, 5, 6, 7, 9\}$ $C - A = \{3, 8, 10\}$	 <p style="text-align: center;">C A</p>
3	$B \cap (1)$	 <p style="text-align: center;">A B</p>

4	$(3) \cup (2)$	
---	----------------	--

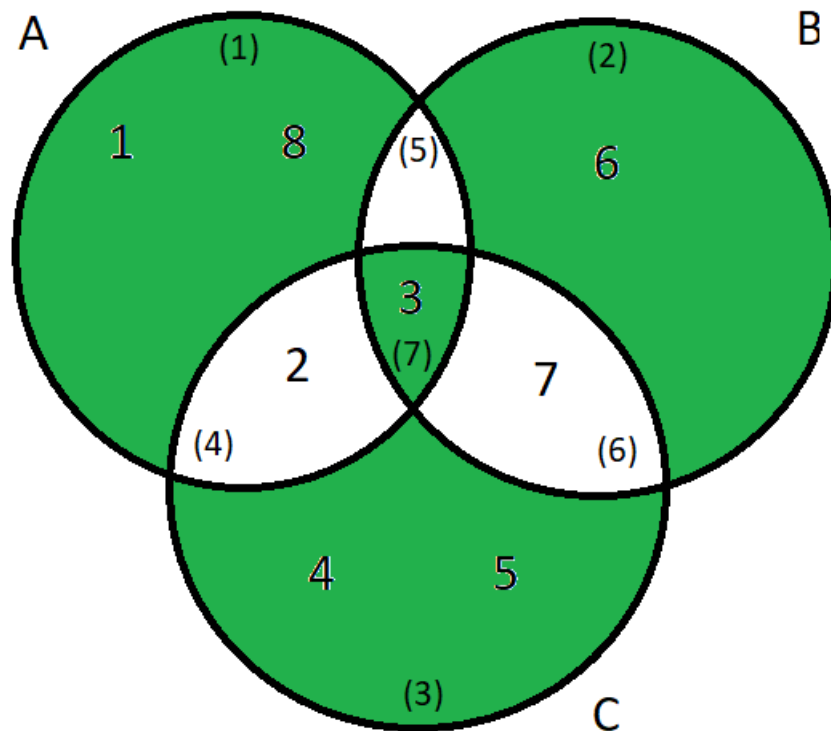
В итоге получаем:

$$\begin{aligned}
 D &= B \cap (A \triangle B) \cup (C - A) = \{1, 4, 5, 9\} \cap \{1, 2, 4, 6, 7\} \cup \{3, 8, 10\} \\
 &= \{1, 4\} \cup \{3, 8, 10\} = \{1, 3, 4, 8, 10\}
 \end{aligned}$$

В виде кругов Эйлера:



2. Поскольку в итоговом множестве должны быть элементы $\{1,3,4,5,6,8\}$, то изобразим их на кругах Эйлера и отметим нужные области:



Следовательно нам нужно из объединения всех областей вычесть область (4) и (6)

В итоге получаем выражение:

$$D = (A \cup B \cup C) - (((A \cap C) \cup (B \cap C)) - (A \cap B)) = \{1, 3, 4, 5, 6, 8\}$$

3. Текст реализаций программных функций операций над множествами с комментариями:

- 1) Элементы множества A хранятся в массиве A. Элементы массива A неупорядочены;

```
//
//начало блока вспомогательных функций
//

{сдвигает элементы массива arr размера n влево}
procedure move_elems_to_left(var arr: t_arr; n: integer);
var i,j: byte;
    key: integer;
begin
    for i := 2 to n do
        begin
            key := arr[i];
            j := i;
            while (key <> 0) and (j > 1) and ((arr[j-1] > key) or (arr[j-1] = 0))
do
                begin
                    arr[j] := arr[j-1];
                    j := j - 1;
                end;
            arr[j] := key;
        end;
    end;

{возвращает "истину", если элем содержится в arr размера n, иначе - "ложь"}
function is_elem_in(arr: t_arr; n: integer; elem: integer): boolean;
var
    i: byte;
    f: boolean;
begin
    f := false;
    is_elem_in := false;
    i := 1;

    while (f = false) and (i <= n) do
        begin
            if (elem = arr[i]) then
                begin
                    is_elem_in := true;
                    f := true;
                end;
            i := i + 1;
        end;
    end;

//
//конец блока вспомогательных функций
//

//
//начало блока основных функций операций над мн-вами
//

{возвращает "истину", если мас-в arr2 размера n2 включает в
себя мас-в arr1 размера n1, иначе - "ложь"}
```

```

function isInclude(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer):
boolean;
var
  i: byte;
  f: boolean;
begin
  isInclude := true;

  while (f = true) and (i <= n1) do
  begin
    if not(is_elem_in(arr2, n2, arr1[i])) then
    begin
      f := false;
      isInclude := false;
    end;

    i := i + 1;
  end;
end;

{возвращает "истину", если мас-в arr2 размера n2 строго включает в себя
мас-в arr1 размера n1, иначе - "ложь"}
function isStrictInclude(arr1: t_arr; n1: integer; arr2: t_arr; n2:
integer): boolean;
var
  i: byte;
  f: boolean;
begin
  if (isInclude(arr1, n1, arr2, n2)) then
    if (n1 <> n2) then
      isStrictInclude := true
    else
      isStrictInclude := false;
end;

{возвращает объединение мас-вов arr1 размера n1 и arr2 размера n2}
function union(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer): t_arr;
var
  res: t_arr;
  i,k: byte;
begin
  for i := 1 to n1 do
    res[i] := arr1[i];

  k := n1;

  for i := 1 to n2 do
    if not (is_elem_in(res, n1, arr2[i])) then
    begin
      k := k + 1;
      res[k] := arr2[i];
    end;

  union := res;
end;

{возвращает пересечение мас-вов arr1 размера n1 и arr2 размера n2}
function inters(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer): t_arr;
var
  res: t_arr;
  i,k: byte;
begin

```

```

k := 0;

for i := 1 to n1 do
  if (is_elem_in(arr2, n2, arr1[i])) then
    begin
      k := k + 1;
      res[k] := arr1[i];
    end;

  inters := res;
end;

{возвращает разность мас-вов arr1 размера n1 и arr2 размера n2}
function subtract(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer):
t_arr;
var
  res: t_arr;
  i,k: byte;
begin
  k := 0;

  for i := 1 to n1 do
    if not (is_elem_in(arr2, n2, arr1[i])) then
      begin
        k := k + 1;
        res[k] := arr1[i];
      end;

  subtract := res;
end;

{возвращает симметричную разность мас-вов arr1 размера n1 и arr2 размера
n2}
function sim_subtract(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer):
t_arr;
var
  res: t_arr;
  i: byte;
  AsubB,BsubA: t_arr;
begin
  AsubB := subtract(arr1,n1,arr2,n2);
  BsubA := subtract(arr2,n2,arr1,n1);
  res := union(AsubB,n1+n2,BsubA,n2+n1);

  move_elems_to_left(res,50);
  sim_subtract := res;
end;

//
//конец блока основных функций операций над мн-вами
//

```

- 2) Элементы множества A хранятся в массиве A. Элементы массива A упорядочены по возрастанию:

```

//
//начало блока основных функций операций над мн-вами
//

{возвращает "истину", если мас-в arr2 размера n2 включает в себя мас-в arr1
размера n1, иначе - "ложь"}

```



```

function isInclude(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer):
boolean;
var
  i: byte = 1;
  j: byte = 1;
  f: boolean = true;
begin
  isInclude := true;

  while (f = true) and (i <= n1) do
    begin
      if (arr1[i] = arr2[j]) then
        begin
          i := i + 1;
          j := j + 1;
        end
      else
        if (arr1[i] > arr2[j]) then
          j := j + 1
        else
          f := false;
        end;
      end;
    end;

    {возвращает "истину", если мас-в arr2 размера n2 строго включает в себя
    мас-в arr1 размера n1, иначе - "ложь"}
function isStrictInclude(arr1: t_arr; n1: integer; arr2: t_arr; n2:
integer): boolean;
var
  i: byte;
  f: boolean;
begin
  if (isInclude(arr1, n1, arr2, n2)) then
    if (n1 <> n2) then
      isStrictInclude := true
    else
      isStrictInclude := false;
    end;
  end;

  {возвращает объединение мас-вов arr1 размера n1 и arr2 размера n2}
function union(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer): t_arr;
var
  res: t_arr;
  i: byte = 1;
  j: byte = 1;
  k: byte = 0;
begin
  while (i <= n1) and (j <= n2) do
    begin
      k := k + 1;

      if (arr1[i] = arr2[j]) then
        begin
          res[k] := arr1[i];
          i := i + 1;
          j := j + 1;
        end
      else
        if (arr1[i] > arr2[j]) then
          begin
            res[k] := arr2[j];
            j := j + 1;
          end
        else
          res[k] := arr1[i];
          i := i + 1;
        end;
      end;
    end;
  end;

```

```

        end
    else
        begin
            res[k] := arr1[i];
            i := i + 1;
        end;
    end;

while (i <= n1) do
begin
    k := k + 1;
    res[k] := arr1[i];
    i := i + 1;
end;

while (j <= n2) do
begin
    k := k + 1;
    res[k] := arr2[j];
    j := j + 1;
end;

union := res;
end;

{возвращает пересечение мас-вов arr1 размера n1 и arr2 размера n2}
function inters(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer): t_arr;
var
    res: t_arr;
    i: byte = 1;
    j: byte = 1;
    k: byte = 0;
begin
    while (i <= n1) and (j <= n2) do
        begin
            if (arr1[i] = arr2[j]) then
                begin
                    k := k + 1;
                    res[k] := arr1[i];
                    i := i + 1;
                    j := j + 1;
                end
            else
                if (arr1[i] > arr2[j]) then
                    j := j + 1
                else
                    i := i + 1;
                end;
            end;

        inters := res;
    end;

    {возвращает разность мас-вов arr1 размера n1 и arr2 размера n2}
function subtract(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer):
t_arr;
var
    res: t_arr;
    i: byte = 1;
    j: byte = 1;
    k: byte = 0;
begin
    while (i <= n1) and (j <= n2) do

```

```

begin
  if (arr1[i] = arr2[j]) then
    begin
      i := i + 1;
      j := j + 1;
    end
  else
    if (arr1[i] > arr2[j]) then
      j := j + 1
    else
      begin
        k := k + 1;
        res[k] := arr1[i];
        i := i + 1;
      end;
    end;

  while (i <= n1) do
    begin
      k := k + 1;
      res[k] := arr1[i];
      i := i + 1;
    end;

  subtract := res;
end;

{возвращает симметричную разность мас-вов arr1 размера n1 и arr2 размера n2}
function sim_subtract(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer):
t_arr;
var
  res: t_arr;
  a,b: t_arr;
  i: byte = 1;
  j: byte = 1;
  k: byte = 0;
begin
  while (i <= n1) and (j <= n2) do
    begin
      if (arr1[i] = arr2[j]) then
        begin
          i := i + 1;
          j := j + 1;
        end
      else
        if (arr1[i] > arr2[j]) then
          begin
            k := k + 1;
            res[k] := arr2[j];
            j := j + 1;
          end
        else
          begin
            k := k + 1;
            res[k] := arr1[i];
            i := i + 1;
          end;
        end;
      end;

  while (i <= n1) do
    begin

```

```

        k := k + 1;
        res[k] := arr1[i];
        i := i + 1;
    end;

    while (j <= n2) do
    begin
        k := k + 1;
        res[k] := arr2[j];
        j := j + 1;
    end;

    sim_subtract := res;
end;

//
//конец блока основных функций операций над мн-вами
//

```

- 3) Элементы множества A хранятся в массиве A , элементы которого типа `boolean`. Если $i \in A$, то $A_i = \text{true}$, иначе $A_i = \text{false}$

```

type
    t_arr = array[1..50] of integer;
    t_arr_bln = array[1..50] of boolean;

//
//блок процедур ввода и вывода мас-ов
//

procedure read_arr(var arr: t_arr; n: integer);
var
    i: byte;
begin
    for i := 1 to n do
        read(arr[i]);
    end;

procedure print_arr(arr: t_arr; n: integer);
var
    i: byte;
begin
    for i := 1 to n do
        if (arr[i] <> 0) then
            write(arr[i], ' ');
    end;

//
//конец блока процедур ввода и вывода мас-ов
//

//
//начало блока вспомогательных функций
//

{возвращает массив с элементами логического типа, где значение "истина" -
элементы массива arr размера n присутствующие в универсуме u размера un}
function to_bln_arr(arr: t_arr; n: integer; u: t_arr; un: integer):
    t_arr_bln;
var i: byte;
    res: to_arr_bln;

```

```

begin
  for i := 1 to n do
    if (is_elem_in(u,un,arr[i])) then
      res[i] := true;
    end;

    //
    //конец блока вспомогательных функций
    //

    //
    //начало блока основных функций операций над мн-вами
    //

    {возвращает "истину", если массив arr2 размера n2 включает в
    себя массив arr1 размера n1, иначе - "ложь"}
    function isInclude(arr1: t_arr_bln; arr2: t_arr_bln; un: integer): boolean;
    var
      i: byte = 1;
      f: boolean = true;
    begin
      while (f = true) and (i <= un) do
        begin
          f := arr1[i] <= arr2[i];
          i := i + 1;
        end;
        isInclude := f;
      end;

      {возвращает объединение мас-вов arr1 размера n1 и arr2 размера n2}
      function union(arr1: t_arr; arr2: t_arr; un: integer): t_arr_bln;
      var
        res: t_arr_bln;
        i: byte;
      begin
        for i := 1 to un do
          res[i] := (arr1[i] or arr2[i]);

          union := res;
        end;

        {возвращает пересечение массивов arr1 размера n1 и arr2 размера n2}
        function inters(arr1: t_arr; arr2: t_arr; un: integer): t_arr;
        var
          res: t_arr;
          i: byte;
        begin
          for i := 1 to un do
            res[i] := arr1[i] and arr2[i];

            inters := res;
          end;

          {возвращает разность массивов arr1 размера n1 и arr2 размера n2}
          function subtract(arr1: t_arr_bln; arr2: t_arr_bln; un: integer): t_arr;
          var
            res: t_arr_bln;
            i: byte;
          begin
            for i := 1 to un do
              res[i] := (arr1[i] > arr2[i]);
            end;
          end;
        end;
      end;
    end;
  end;
end;

```

```

        subtract := res;
    end;

    {возвращает симметричную разность массивов arr1 размера n1 и arr2 размера n2}
    function sim_subtract(arr1: t_arr_bln; arr2: t_arr_bln; un: integer):
    t_arr;
    var
        res: t_arr_bln;
        i: byte;
    begin
        for i := 1 to un do
            res[i] := arr1[i] <> arr2[i];

            sim_subtract := res;
        end;

        //
        //конец блока основных функций операций над мн-вами
        //

```

4. Текст программы:

```

type
    t_arr = array[1..50] of integer;

    //
    //блок процедур ввода и вывода мас-ов
    //

    procedure read_arr(var arr: t_arr; n: integer);
    var
        i: byte;
    begin
        for i := 1 to n do
            read(arr[i]);
        end;

    procedure print_arr(arr: t_arr; n: integer);
    var
        i: byte;
    begin
        for i := 1 to n do
            if (arr[i] <> 0) then
                write(arr[i], ' ');
        end;

        //
        //конец блока процедур ввода и вывода мас-ов
        //

        //
        //начало блока основных функций операций над мн-вами
        //

    {возвращает "истину", если мас-в arr2 размера n2 включает в себя мас-в arr1
    размера n1, иначе - "ложь"}
    function isInclude(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer):
    boolean;
    var
        i: byte = 1;
        j: byte = 1;

```

```

    f: boolean = true;
begin
    isInclude := true;

    while (f = true) and (i <= n1) do
        begin
            if (arr1[i] = arr2[j]) then
                begin
                    i := i + 1;
                    j := j + 1;
                end
            else
                if (arr1[i] > arr2[j]) then
                    j := j + 1;
                else
                    f := false;
                end;
            end;
        end;

        {возвращает "истину", если мас-в arr2 размера n2 строго включает в себя мас-в
        arr1 размера n1, иначе - "ложь"}
        function isStrictInclude(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer):
        boolean;
        var
            i: byte;
            f: boolean;
        begin
            if (isInclude(arr1, n1, arr2, n2)) then
                if (n1 <> n2) then
                    isStrictInclude := true
                else
                    isStrictInclude := false;
                end;
            end;

            {возвращает объединение мас-вов arr1 размера n1 и arr2 размера n2}
            function union(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer): t_arr;
            var
                res: t_arr;
                i: byte = 1;
                j: byte = 1;
                k: byte = 0;
            begin
                while (i <= n1) and (j <= n2) do
                    begin
                        k := k + 1;

                        if (arr1[i] = arr2[j]) then
                            begin
                                res[k] := arr1[i];
                                i := i + 1;
                                j := j + 1;
                            end
                        else
                            if (arr1[i] > arr2[j]) then
                                begin
                                    res[k] := arr2[j];
                                    j := j + 1;
                                end
                            else
                                begin
                                    res[k] := arr1[i];
                                    i := i + 1;
                                end
                            end;
                        end;
                    end;
                end;
            end;
        end;
    end;
end;

```

```

        end;
    end;

    while (i <= n1) do
    begin
        k := k + 1;
        res[k] := arr1[i];
        i := i + 1;
    end;

    while (j <= n2) do
    begin
        k := k + 1;
        res[k] := arr2[j];
        j := j + 1;
    end;

    union := res;
end;

{возвращает пересечение мас-вов arr1 размера n1 и arr2 размера n2}
function inters(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer): t_arr;
var
    res: t_arr;
    i: byte = 1;
    j: byte = 1;
    k: byte = 0;
begin
    while (i <= n1) and (j <= n2) do
    begin
        if (arr1[i] = arr2[j]) then
        begin
            k := k + 1;
            res[k] := arr1[i];
            i := i + 1;
            j := j + 1;
        end
        else
            if (arr1[i] > arr2[j]) then
                j := j + 1
            else
                i := i + 1;
            end;
        end;

    inters := res;
end;

{возвращает разность мас-вов arr1 размера n1 и arr2 размера n2}
function subtract(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer): t_arr;
var
    res: t_arr;
    i: byte = 1;
    j: byte = 1;
    k: byte = 0;
begin
    while (i <= n1) and (j <= n2) do
    begin
        if (arr1[i] = arr2[j]) then
        begin
            i := i + 1;
            j := j + 1;
        end
    end;
end;

```



```

        else
            if (arr1[i] > arr2[j]) then
                j := j + 1
            else
                begin
                    k := k + 1;
                    res[k] := arr1[i];
                    i := i + 1;
                end;
            end;
        end;

    while (i <= n1) do
        begin
            k := k + 1;
            res[k] := arr1[i];
            i := i + 1;
        end;

    subtract := res;
end;

```

{возвращает симметричную разность мас-вов arr1 размера n1 и arr2 размера n2}

```

function sim_subtract(arr1: t_arr; n1: integer; arr2: t_arr; n2: integer):
t_arr;
var
    res: t_arr;
    a,b: t_arr;
    i: byte = 1;
    j: byte = 1;
    k: byte = 0;
begin
    while (i <= n1) and (j <= n2) do
        begin
            if (arr1[i] = arr2[j]) then
                begin
                    i := i + 1;
                    j := j + 1;
                end
            else
                if (arr1[i] > arr2[j]) then
                    begin
                        k := k + 1;
                        res[k] := arr2[j];
                        j := j + 1;
                    end
                else
                    begin
                        k := k + 1;
                        res[k] := arr1[i];
                        i := i + 1;
                    end;
                end;
            end;

        while (i <= n1) do
            begin
                k := k + 1;
                res[k] := arr1[i];
                i := i + 1;
            end;

        while (j <= n2) do
            begin

```

```

        k := k + 1;
        res[k] := arr2[j];
        j := j + 1;
    end;

    sim_subtract := res;
end;

//
//конец блока основных функций операций над мн-вами
//

var
    t1, t2, t3: t_arr;
    a, b, c, d: t_arr;

    t4, t5, t6, t7, t8: t_arr;
    a1, b1, c1, d1: t_arr;
    aNb: t_arr;

begin
    //программа для задания 1
    writeln('Задание 1');
    writeln('Введите мн-во a');
    read_arr(a, 5);
    writeln('Введите мн-во b');
    read_arr(b, 4);
    writeln('Введите мн-во c');
    read_arr(c, 4);

    t1 := sim_subtract(a, 5, b, 4);
    t2 := subtract(c, 4, a, 5);
    t3 := inters(b, 4, t1, 5);

    d := union(t3, 2, t2, 3);

    print_arr(d, 20);

    writeln('');
    //программа для задания 2

    writeln('Задание 2');
    writeln('Введите мн-во a');
    read_arr(a1, 4);
    writeln('Введите мн-во b');
    read_arr(b1, 3);
    writeln('Введите мн-во c');
    read_arr(c1, 5);

    //находим объединение всех областей
    t4 := union(a1, 4, b1, 3);
    d := union(c1, 5, t4, 6);

    //находим пересечение A с B
    aNb := inters(a1, 4, b1, 3);

    //находим объединение пересечений A с C и B с C
    t5 := inters(a1, 4, c1, 5);
    t6 := inters(b1, 3, c1, 5);
    t7 := union(t5, 2, t6, 2);

```

```

//вычитаем из объединения пересечений A с C и B с C область пересечения A с
B
t8 := subtract(t7, 4, aNb, 1);

//находим итоговое значение
d := subtract(d, 12, t8, 2);
print_arr(d, 50);
end.

```

5. Результаты вычислений программы:

```

Задание 1
Введите мн-во a
2 5 6 7 9
Введите мн-во b
1 4 5 9
Введите мн-во c
3 7 8 10
1 4 3 8 10
Задание 2
Введите мн-во a
1 2 3 8
Введите мн-во b
3 6 7
Введите мн-во c
2 3 4 5 7
3 4 5 1 8 6

```

Результаты вычислений, рассчитанные программой, совпадают с результатами, вычисленными ручным способом