Лабораторная работа № 1_1

Постановка задачи.

Написать программу для обработки множеств, которая позволяет ввести три множества символов a, b и c и вычислить множество, являющееся:

- объединением множеств a и b;
- пересечением множеств a и b;
- разностью множеств a и b;
- множеством, полученным из множеств a, b и c по формуле (a ∪ b) \ (b ∪ c).

Уточнение: Реализовать множество с помощью символьного массива и массива логических элементов. У обеих реализаций дожен быть одинаковый интерфейс.

Таблица данных

Класс	Имя	Смысл	Тип	Структура
Входные данные	a,b,c	исходные множества	TCharSet	массив или запись(в зависимости от реализации)
Выходные данные	res	результирующее множество	TCharSet	массив или запись(в зависимости от реализации)
Промежуточные данные	fin,fout	входной и выходной файл	текстовый файл	файл

Таблица данных модуля с реализацией через массив символов

Тип TCharSet - запись с двумя полями:

1. n - количество элементов в множестве
2. s - массив символов(различных)

Класс	Имя	Смысл	Тип	Структура
Входные данные	x,x1,x2	исходные множества	TCharSet	запись
Промежуточные данные	ch	символ	символьный	прост. перем.
Промежуточные данные	f	входной и выходной файл	текстовый файл	файл

Таблица данных модуля с реализацией через массив логических элементов

Тип TCharSet - массив логических элементов

Класс	Имя	Смысл	Тип	Структура
Входные данные	x,x1,x2	исходные множества	TCharSet	массив
Промежуточные данные	ch	символ	символьный	прост. перем.
Промежуточные данные	f	входной и выходной файл	текстовый файл	файл

Входная форма

< множество a \>
< множество b \>
< множество c \>

Выходная форма

Union: Объединение множеств a и b
Intersection: Персечение множеств a и b
Difference: Разность множеств a и b
Expr: Множество, (a ∪ b) \ (b ∪ c)

Аномалии

- Недостаточно параметров.
- Невозможно открыть файл для чтения.

Тестовые примеры

Входные данные:
abcdih
cdef
ghij
Ожидаемый результат
Union: abcdefhi
Intersection: cd
Difference: abhi
Expr: ab

Метод

Считываем посимвольно данные из файла и добавляем их в множество
Находим новые множества используя реализацию множества с помощью массива символов или массива логических элементов
Выводим в файл используя цикл и проверку на вхождение символа в множество

Алгоритм

Программа

```
program Lab1;

uses mycharset_bools;

var a,b,c,res: TCharSet;
    fin,fout: textfile;

begin
    if ParamCount < 2 then writeln('Недостаточно параметров!')
    else begin
        if not FileExists(ParamStr(1)) then          { Проверяем существование файла
            writeln('Невозможно открыть файл ''', ParamStr(1), ''' для чтения')
            else begin
                AssignFile(fin, ParamStr(1));
                Reset(fin);
                vvod(a,fin);
                vvod(b,fin);
                vvod(c,fin);
```

```
                closefile(fin);
                AssignFile(fout, ParamStr(2));
                Rewrite(fout);
                res := Union(a,b); // объединение
                write(fout,'Union: ');
                vivod(res,fout);
                writeln(fout);
                res := Intersection(a,b);// пересечение
                write(fout,'Intersection: ');
                vivod(res,fout);
                writeln(fout);
                res := Difference(a,b);// разность множеств
                write(fout, 'Difference: ');
                vivod(res,fout);
                writeln(fout);
                res := Difference(Union(a,b), Union(b,c));   //(а или б) искл (б ил
                write(fout,'Expr: ');
                vivod(res,fout);
                closefile(fout);
                end;
    end;
end.
```

Модуль с реализацией с помошью символьного массива

```
unit mycharset_chars;

interface

type TCharSet = record
                    s: array[1..255] of char;
                    n: integer;
                    end;

procedure Init(var x:TCharSet);
function InSet(const ch:char; const x:TCharSet):boolean;
procedure Add(var x:TCharSet; ch: char);
function Union(const x1,x2:TCharSet):TCharSet;
function Intersection(const x1,x2: TCharSet):TCharSet;
function Difference(const x1,x2: TCharSet):TCharSet;
function Equal(const x1,x2: TCharSet):boolean;
function NotEqual(const x1,x2: TCharSet):boolean;
procedure vvod(var x:TCharSet; f:textfile);
procedure vivod(var x:TCharSet; f:textfile);


implementation

procedure vvod(var x:TCharSet; f:textfile);
var ch:char;
begin
```

```
    while not eoln(f) do begin
        read(f,ch);
        Add(x,ch);
    end;
    readln(f);
end;


procedure vivod(var x:TCharSet; f:textfile);
var ch:char;
    i: integer;
begin
    for i := 1 to x.n do begin
        write(f,x.s[i]);
    end;
    writeln(f);
end;

procedure Init(var x:TCharSet);
begin
    x.n := 0;
end;


function InSet(const ch:char; const x:TCharSet):boolean;
var i:integer;
begin
    result := false;
    i := 1;
    while(i <= x.n) and not result do begin
        if x.s[i]=ch then result := true;
        i := i + 1;
    end;
end;


procedure Add(var x:TCharSet; ch: char);
begin
    if not InSet(ch, x) then begin
        x.n := x.n + 1;
        x.s[x.n] := ch;
    end;
end;


function Union(const x1,x2:TCharSet):TCharSet;
var i: integer;
begin
    result := x1;
    for i := 1 to x2.n do begin
        if not InSet(x2.s[i], x1) then begin
            result.n := result.n + 1;
            result.s[result.n] := x2.s[i];
        end;
    end;
end;
```

```
function Intersection(const x1,x2: TCharSet):TCharSet;
var i:integer;
begin
    result.n := 0;
    for i := 1 to x1.n do
        if InSet(x1.s[i], x2) then begin
            result.n := result.n + 1;
            result.s[result.n] := x1.s[i];
        end;
end;


function Difference(const x1,x2: TCharSet):TCharSet;
var i:integer;
begin
    result.n := 0;
    for i := 1 to x1.n do
        if not InSet(x1.s[i], x2) then begin
            result.n := result.n + 1;
            result.s[result.n] := x1.s[i];
        end;
end;


function Equal(const x1,x2: TCharSet):boolean;
var i: integer;
begin
    result := true;
    i := 1;
    while (i <= x1.n) and result do begin
        if not InSet(x1.s[i], x2) then result := false;
        i := i + 1;
    end;
end;


function NotEqual(const x1,x2: TCharSet):boolean;
begin
    result := not Equal(x1,x2);
end;


end.
```

Модуль с реализацие с помощью массива логических элементов

```
unit mycharset_bools;

interface
```

```
type TCharSet = array [0..255] of boolean;

procedure Init(var x:TCharSet);
function InSet(const ch:char; const x:TCharSet):boolean;
procedure Add(var x:TCharSet; ch: char);
function Union(const x1,x2:TCharSet):TCharSet;
function Intersection(const x1,x2: TCharSet):TCharSet;
function Difference(const x1,x2: TCharSet):TCharSet;
function Equal(const x1,x2: TCharSet):boolean;
function NotEqual(const x1,x2: TCharSet):boolean;
procedure vvod(var x:TCharSet; f:textfile);
procedure vivod(var x:TCharSet; f:textfile);


implementation

procedure vvod(var x:TCharSet; f:textfile);
var ch:char;
begin
    while not eoln(f) do begin
        read(f,ch);
        Add(x,ch);
    end;
    readln(f);
end;


procedure vivod(var x:TCharSet; f:textfile);
var ch:char;
    i:integer;
begin
    for i := 0 to 255 do begin
        if x[i] then write(f,chr(i));
    end;
    writeln(f);
end;

procedure Init(var x:TCharSet);
var i:integer;
begin
    for i := 0 to 255 do
        x[i] := false;
end;


function InSet(const ch:char; const x:TCharSet):boolean;
var i:integer;
begin
    result := x[ord(ch)];
end;


procedure Add(var x:TCharSet; ch: char);
begin
    x[ord(ch)] := true;
end;
```

```
function Union(const x1,x2:TCharSet):TCharSet;
var i: integer;
begin
    for i := 0 to 255 do
        result[i] := x1[i] or x2[i];
end;


function Intersection():TCharSet;
var i: integer;
begin
    for i := 0 to 255 do
        result[i] := x1[i] and  x2[i];
end;


function Difference(const x1,x2: TCharSet):TCharSet;
var i:integer;
begin
    for i := 0 to 255 do begin
        result[i] := x1[i] and not x2[i];
    end;
end;


function Equal(const x1,x2: TCharSet):boolean;
var i: integer;
begin
    result := true;
    i := 0;
    while (i <= 255) and result do begin
        if x1[i] xor x2[i] then result := false;
        i := i + 1;
    end;
end;


function NotEqual(const x1,x2: TCharSet):boolean;
begin
    result := not Equal(x1,x2);
end;




initialization

finalization

end.
```