

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Дата сдачи на проверку:

«__» _____ 2025 г.

Проверено:

«__» _____ 2025 г.

ИЗУЧЕНИЕ РАБОТЫ С МНОЖЕСТВАМИ

Отчёт по лабораторной работе №1

по дисциплине

«Дискретная математика»

Разработал студент гр. ИВТб-1301-05-00 _____ /Черкасов А. А./
(подпись)

Проверила преподаватель _____ /Пахарева И. В./
(подпись)

Работа защищена «__» _____ 2025 г.

Киров

2025

Цель

Цель работы: Изучение основ теории множеств, базовых операций над ними, разработка приложений на языке Паскаль согласно заданию.

Задание

- Реализовать программу для выполнения заданных операций над множествами. Программа должна работать при некорректном вводе, выдавая информационное сообщение об ошибке.

Вар / Множ	A	B	C	D	E	X	Y	K
27	\mathbb{R}	\mathbb{Q}	\mathbb{Z}	латиница	кириллица	$A \cap B \cap C$	$E \cup D$	$X \Delta Y$

Решение

Схема алгоритма решения представлена на рисунках 1.1, 1.2 и 1.3. Пример работы программы представлен на рисунках 2.1, 2.2. Исходный код решений представлен в Приложении А1.

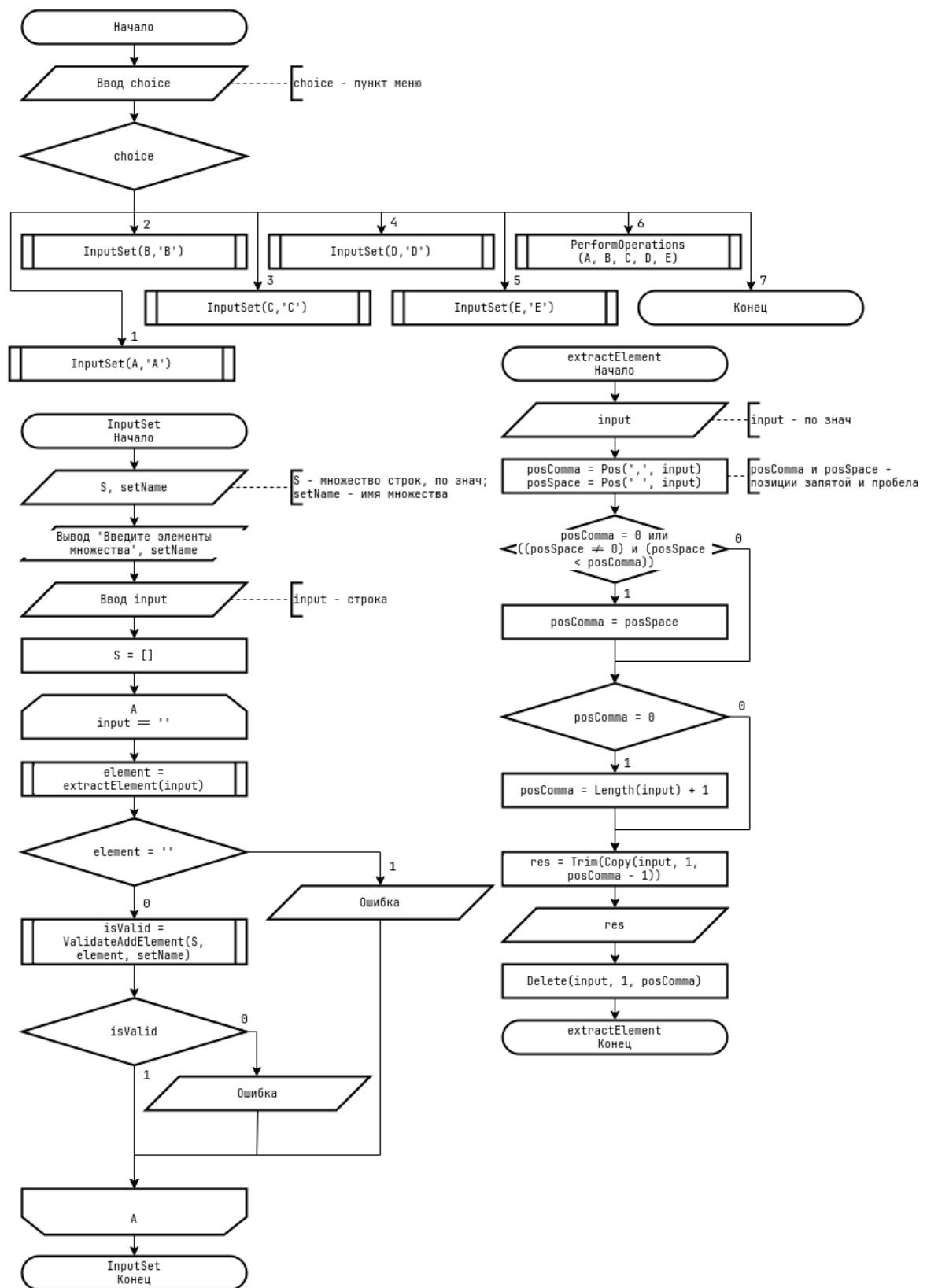


Рисунок 1.1 - Схемы алгоритмов основной программы, подпрограмм ввода множеств и извлечения элемента.

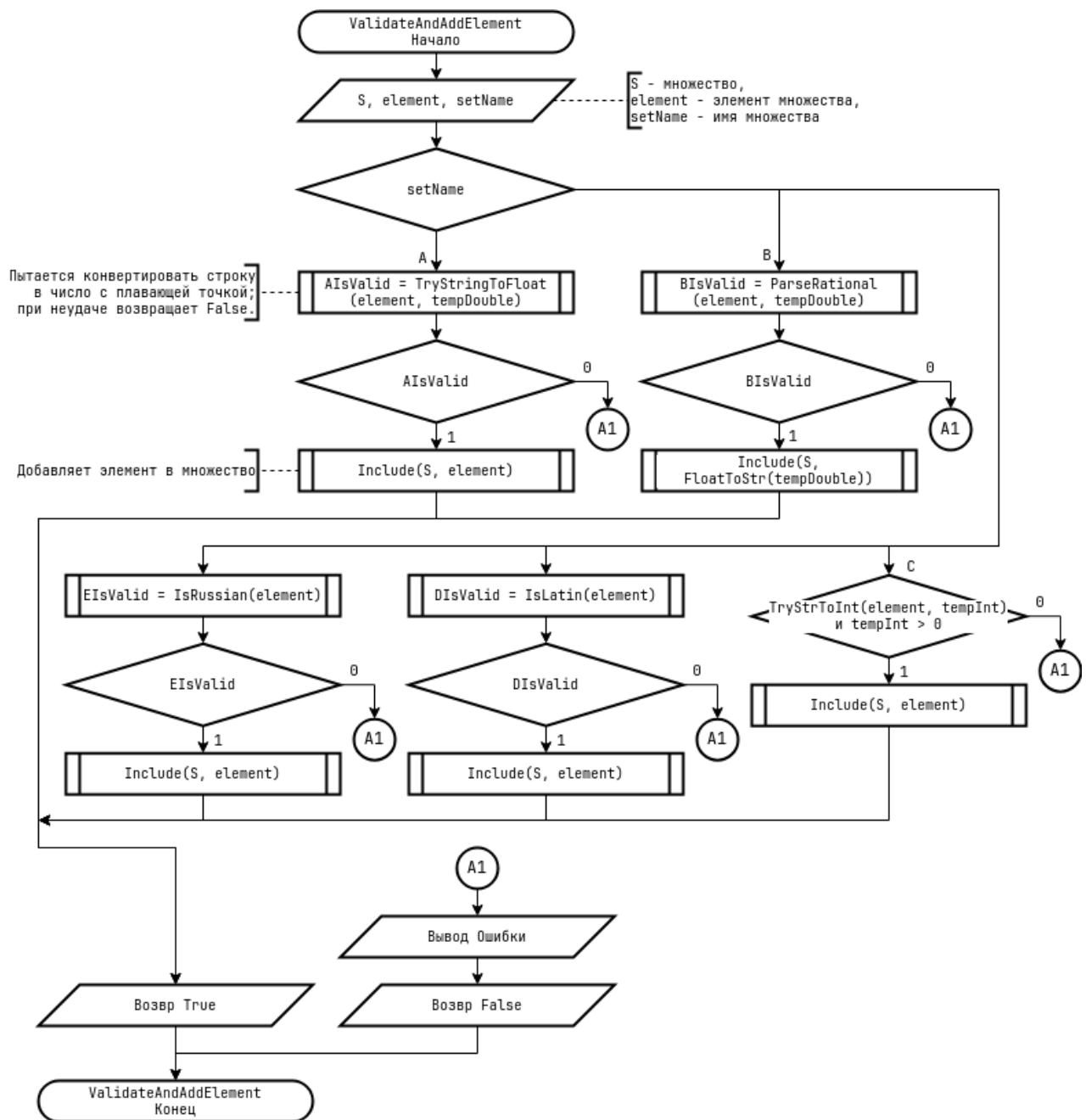


Рисунок 1.2 - Схема алгоритма подпрограммы проверки на валидность элемента.

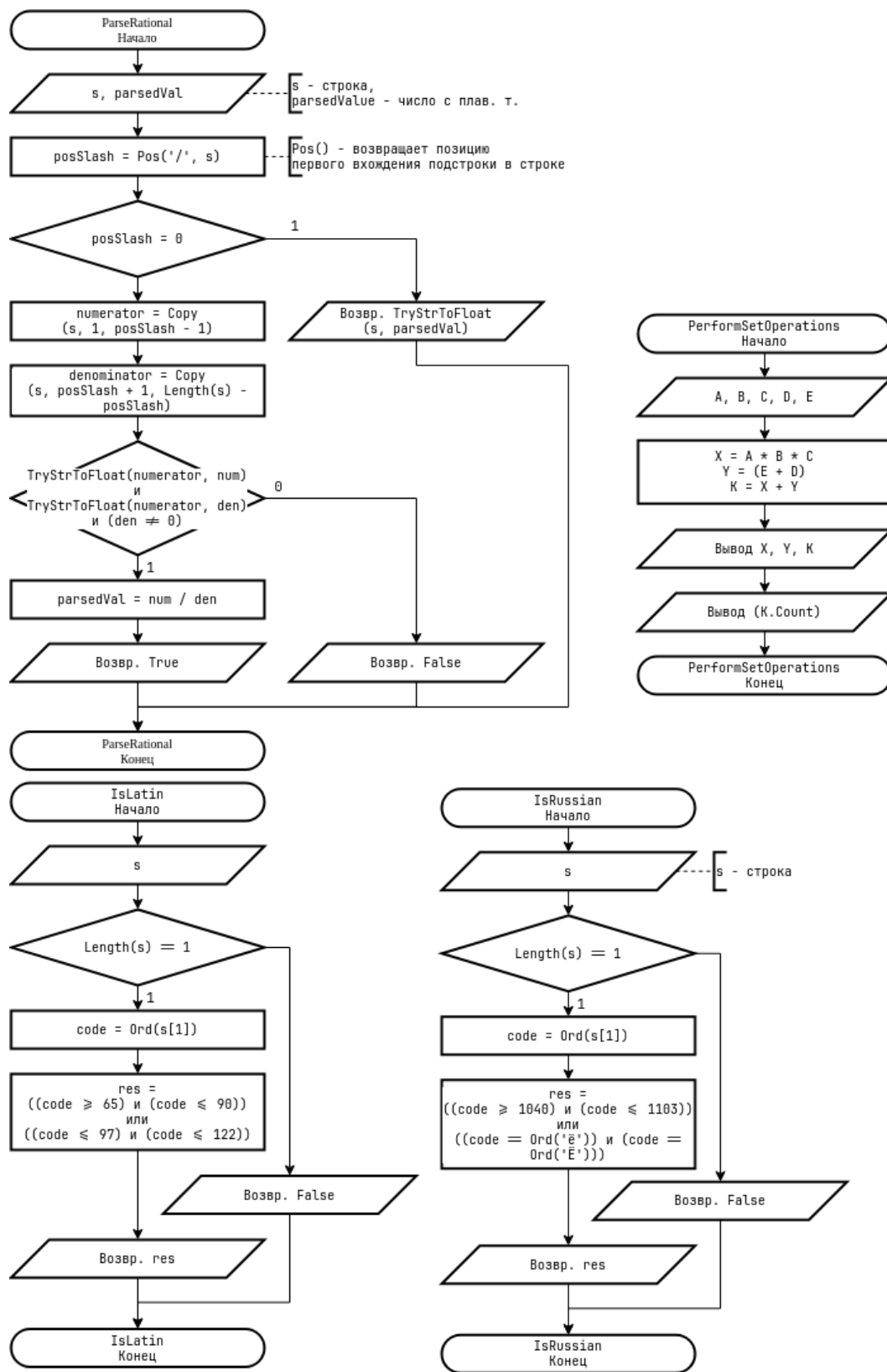


Рисунок 1.3 - Схема алгоритма подпрограмм проверки элементов на латинский и кириллический алфавит, подпрограмма выполнения операций с множествами.

```
Меню:
1. Ввести множество A (R - множество действительных чисел)
{}
2. Ввести множество B (Q - множество рациональных чисел)
{}
3. Ввести множество C (Z - множество целых чисел > 0)
{}
4. Ввести множество D (лат.)
{}
5. Ввести множество E (рус.)
{}
6. Выполнить операции над множествами
7. Выход
Выберите действие:
```

Рисунок 2.1 - Пример работы программы.

```
Меню:
1. Ввести множество A (R - множество действительных чисел)
{1,2,3,4,5}
2. Ввести множество B (Q - множество рациональных чисел)
{3,0.5,0.3333333333333333}
3. Ввести множество C (Z - множество целых чисел > 0)
{3,4,5}
4. Ввести множество D (лат.)
{q,w,a}
5. Ввести множество E (рус.)
{}
6. Выполнить операции над множествами
7. Выход
Выберите действие: 6
X - пересечение A, B и C = {3}
Y - симметрическая разность E и D = {q, w, a}
K - объединение X и Y = {3, q, w, a}
Мощность множества K: 4
Нажмите любую клавишу для продолжения...
```

Рисунок 2.2 - Пример работы программы.

Вывод

В результате работы была реализованна программа выполняющая операции над множествами на языке паскаль.

Приложение А1. Исходный код

```
uses crt;

type
  TStringSet = set of string;

var
  A, B, C, D, E, X, Y, K: TStringSet;
  choice: string;
  input, element: string;
  i: Integer;
  value: double;
  Power: Integer;

procedure ClearAndShowMenu;
begin
  ClrScr;
  WriteLn('Меню:');
  WriteLn('1. Ввести множество A (R - множество действительных чисел)');
  WriteLn(A);
  WriteLn('2. Ввести множество B (Q - множество рациональных чисел)');
  WriteLn(B);
  WriteLn('3. Ввести множество C (Z - множество целых чисел > 0)');
  WriteLn(C);
  WriteLn('4. Ввести множество D (лат.)');
  WriteLn(D);
  WriteLn('5. Ввести множество E (рус.)');
  WriteLn(E);
  WriteLn('6. Выполнить операции над множествами');
  WriteLn('7. Выход');
  Write('Выберите действие: ');
end;
```



```

function ParseRational(const s: string; var parsedValue: double): boolean;
var
    posSlash: Integer;
    numerator, denominator: string;
    num, den: Double;
begin
    posSlash := Pos('/', s);
    if posSlash = 0 then
        ParseRational := TryStrToFloat(s, parsedValue)
    else
        begin
            numerator := Copy(s, 1, posSlash - 1);
            denominator := Copy(s, posSlash + 1, Length(s) - posSlash);
            if TryStrToFloat(numerator, num) and TryStrToFloat(denominator, den)
                and (den <> 0) then
                begin
                    parsedValue := num / den;
                    ParseRational := True;
                end
            else
                ParseRational := False;
            end;
        end;
end;

```

```

function IsRussian(s: string): boolean;
var
    ch: Char;
    code: Integer;
begin
    if Length(s) = 1 then
        begin
            ch := s[1];
            code := Ord(ch);
            IsRussian := ((code >= 1040) and (code <= 1103))
        end;
    end;
end;

```

```

        or (code = Ord('ë')) or (code = Ord('Ë'));
    end
else
    IsRussian := False;
end;

function IsLatin(s: string): boolean;
var
    ch: Char;
    code: Integer;
begin
    if Length(s) = 1 then
        begin
            ch := s[1];
            code := Ord(ch);
            IsLatin := ((code >= 65) and (code <= 90)) or ((code >= 97) and (code <= 122));
        end
    else
        IsLatin := False;
    end;
end;

function ExtractElement(var input: string): string;
var
    posComma, posSpace: Integer;
begin
    posComma := Pos(',', input);
    posSpace := Pos(' ', input);
    if (posComma = 0) or ((posSpace <> 0) and (posSpace < posComma)) then
        posComma := posSpace;
    if posComma = 0 then
        posComma := Length(input) + 1;
    Result := Trim(Copy(input, 1, posComma - 1));
    Delete(input, 1, posComma);
end;

```

```

procedure PrintSet(const setName: string; const setArray: array of string);
var
    i: Integer;
begin
    Write('    Множество ', setName, ': {'');
    for i := Low(setArray) to High(setArray) do
    begin
        Write(setArray[i]);
        if i < High(setArray) then
            Write(', ');
    end;
    WriteLn('}');
end;

function ValidateAndAddElement(var S: TStringSet; const element: string;
    const setName: string): boolean;
var
    tempInt: Integer;
    tempDouble: Double;
begin
    case setName of
        'A':
            begin
                if TryStrToFloat(element, tempDouble) then
                begin
                    Include(S, element);
                    ValidateAndAddElement := True;
                end
                else
                begin
                    WriteLn('Ошибка: Множество A может содержать только действительные числа.');
```

```

end;
'B':
begin
    if ParseRational(element, tempDouble) then
    begin
        Include(S, FloatToStr(tempDouble));
        ValidateAndAddElement := True;
    end
    else
    begin
        WriteLn('Ошибка: Множество В может содержать только рациональные числа.');

```

```

        end
    else
    begin
        WriteLn('Ошибка: Множество D может содержать только латинские буквы.');
```

ValidateAndAddElement := False;

```

    end;
end;
'E':
begin
    if IsRussian(element) then
    begin
        Include(S, element);
        ValidateAndAddElement := True;
    end
    else
    begin
        WriteLn('Ошибка: Множество E может содержать только русские буквы.');
```

ValidateAndAddElement := False;

```

    end;
end;
end;
end;

procedure InputSet(var S: TStringSet; const setName: string; const criteria: string);
begin
    Write('Введите элементы множества ', setName, ' (', criteria, '): ');
    ReadLn(input);
    S := [];
    while input <> '' do
    begin
        element := ExtractElement(input);
        if element = '' then
        begin
            WriteLn('Ошибка: Введена пустая строка.');
```

```

    WriteLn('Нажмите любую клавишу для продолжения...');
    ReadKey;
    Continue;
end;

if not ValidateAndAddElement(S, element, setName) then
begin
    WriteLn('Нажмите любую клавишу для продолжения...');
    ReadKey;
end;
end;
end;

procedure PerformSetOperations;
var
    XArray, YArray, KArray: array of string;
    i: Integer;
begin
    X := A * B * C;
    Y := (E + D) - (E * D);
    K := X + Y;

    Write('X - пересечение A, B и C = {');
    XArray := X.ToArray;
    for i := Low(XArray) to High(XArray) do
    begin
        Write(XArray[i]);
        if i < High(XArray) then
            Write(', ');
    end;
    WriteLn('}');

    Write('Y - симметрическая разность E и D = {');
    YArray := Y.ToArray;

```

```

for i := Low(YArray) to High(YArray) do
begin
    Write(YArray[i]);
    if i < High(YArray) then
        Write(', ');
end;
WriteLn('}');

Write('K - объединение X и Y = {');
KArray := K.ToArray;
for i := Low(KArray) to High(KArray) do
begin
    Write(KArray[i]);
    if i < High(KArray) then
        Write(', ');
end;
WriteLn('}');

Power := K.Count;
WriteLn('Мощность множества K: ', Power);
WriteLn('Нажмите любую клавишу для продолжения...');
ReadKey;
end;

begin
repeat
    ClearAndShowMenu;
    ReadLn(choice);
    case choice of
        '1': InputSet(A, 'A', 'R - множество действительных чисел');
        '2': InputSet(B, 'B', 'Q - множество рациональных чисел');
        '3': InputSet(C, 'C', 'Z - множество целых чисел > 0');
        '4': InputSet(D, 'D', 'лат. ');
        '5': InputSet(E, 'E', 'рус. ');
    end;
until choice = '5';
end;

```

```
        '6': PerformSetOperations;
        '7': WriteLn('Программа завершена.');
```

else

```
        WriteLn('Неверный выбор. Попробуйте снова.');
```

end;

until choice = '7';

end.