

Разбор задания №2

Анализ ДНК

Постановка задачи

Вы работаете в биотехнологической компании, которая разрабатывает алгоритмы для анализа цепочек ДНК. В процессе исследований биоинженеры представляют последовательности фрагментов ДНК в виде квадратных матриц, где каждая клетка содержит символ нуклеотида (А, Т, С, G).

Было замечено, что некоторые виды мутаций проявляются как палиндромные последовательности нуклеотидов вдоль строки, столбца или диагонали. Палиндром — это биологический маркер симметричной мутации, и его наличие может указывать на потенциально нестабильные участки ДНК. Необходимо найти все палиндромы.

Постановка задачи

Формат входных данных:

На вход программа принимает размер матрицы N ($2 \leq N \leq 10$), потом на N строках по N символов нуклеотидов (**A, T, C, G**). Необходимо найти все палиндромы (слова, читающиеся одинаково слева направо и справа налево), которые можно получить:

- В строках
- В столбцах
- На главных диагоналях (из левого верхнего в правый нижний угол и наоборот)

Гарантируется, что матрица вводится нужного размера.

Постановка задачи

Формат выходных данных:

- После нахождения всех уникальных палиндромов, программа выводит их в алфавитном порядке.
- Если палиндромов нет, вывести **'NONE'**.
- Если в матрице попадется символ, не являющийся нуклеотидом (**A, T, C, G**), вывести **'ERROR'** и завершить программу.
- Если N не входит в допустимый диапазон, вывести **'ERROR'** и завершить программу.

Примеры входных и выходных данных

Ввод	Вывод	Объяснение
3 ACA CGC TTT	ACA CGC TTT	В этой матрице присутствуют только палиндромы в строках
3 ACA CAC ACA	AAA ACA CAC	В этой матрице присутствуют палиндромы в строках (ACA, CAC), в столбцах (ACA, CAC) и на главных диагоналях (AAA), после проверки на уникальность и сортировке остаются AAA, ACA, CAC
4 ATCG AAAC GTAA TTTC	NONE	В этой матрице отсутствуют палиндромы.
4 ABCD EFGH IJKL MNOP	ERROR	В матрице присутствуют символы, не являющиеся нуклеотидами (A, T, C, G)
52 ...	ERROR	Размер матрицы не может превышать 10.

Заготовка

```
PROCEDURE SortPalindromesLexicographically(VAR PalindromeCount: INTEGER; VAR Palindromes: PalindromeArray);
VAR
  I, J: INTEGER;
  Temp: STRING;
BEGIN
  FOR I := 1 TO PalindromeCount - 1
  DO
    FOR J := I + 1 TO PalindromeCount
    DO
      IF Palindromes[I] > Palindromes[J]
      THEN
        BEGIN
          Temp := Palindromes[I];
          Palindromes[I] := Palindromes[J];
          Palindromes[J] := Temp;
        END
      END
    DO
  DO
END;
```

Заготовка

```
PROCEDURE ReadMatrix(VAR MatrixSize: INTEGER; VAR Matrix: MatrixType);
VAR
    I, J: INTEGER;
BEGIN
    READLN(MatrixSize);
    FOR I := 1 TO MatrixSize
    DO
        BEGIN
            FOR J := 1 TO MatrixSize
            DO
                READ(Matrix[I, J]);
            READLN
            END
        END
    END;
END;
```

Заготовка

```
PROCEDURE PrintPalindromes(VAR PalindromeCount: INTEGER; VAR Palindromes: PalindromeArray);
VAR
    I: INTEGER;
BEGIN
    IF PalindromeCount = 0
    THEN
        WRITELN('NONE')
    ELSE
        FOR I := 1 TO PalindromeCount
        DO
            WRITELN(Palindromes[I])
        END;
    END;
```


Анализ заготовки

Что мы можем:

- Считывать нашу матрицу
- Выводить нашу матрицу
- Сортировать палиндромы

Но:

- Нет обработки ошибок
- Нет палиндромов(

Можно сделать лучше

Добавляем константы, типы и глобальные переменные

CONST

```
Nucleotides = ['A', 'C', 'T', 'G'];  
MaxN = 10;  
MinN = 2;
```

TYPE

```
MatrixType = ARRAY[1 .. MaxN, 1 .. MaxN] OF CHAR;  
PalindromeArray = ARRAY[1 .. MaxN * 2 + 2] OF STRING;
```

VAR

```
Matrix: MatrixType;  
Palindromes: PalindromeArray;  
PalindromeCount, MatrixSize: INTEGER;  
Error: BOOLEAN;
```

Добавляем обработку неправильной записи

```
PROCEDURE ReadMatrix(VAR MatrixSize: INTEGER; VAR Matrix: MatrixType; VAR Error: BOOLEAN);
VAR
  I, J: INTEGER;
BEGIN
  READLN(MatrixSize);
  IF NOT (MatrixSize IN [MinN .. MaxN])
  THEN
    BEGIN
      Error := TRUE;
      EXIT
    END;
  FOR I := 1 TO MatrixSize
  DO
    BEGIN
      FOR J := 1 TO MatrixSize
      DO
        BEGIN
          READ(Matrix[I, J]);
          IF NOT (Matrix[I, J] IN Nucleotides)
          THEN
            BEGIN
              Error := TRUE;
              EXIT
            END
          END;
        END;
      READLN
    END
  END;
END;
```

Добавляем основной блок программы

```
BEGIN
  ReadMatrix(MatrixSize, Matrix, Error);
  IF Error
  THEN
    BEGIN
      WRITELN('ERROR');
      EXIT
    END;
  SearchPalindromesInMatrix(Palindromes, PalindromeCount, MatrixSize, Matrix);
  SortPalindromesLexicographically(PalindromeCount, Palindromes);
  PrintPalindromes(PalindromeCount, Palindromes)
END.
```

Добавляем функцию поиска палиндромов

```
PROCEDURE SearchPalindromesInMatrix(VAR Palindromes: PalindromeArray; VAR PalindromeCount, MatrixSize: INTEGER; Matrix: MatrixType);
VAR
    RowWord, ColWord, MainDiagonal, SecondaryDiagonal: STRING;
    J, I: INTEGER;
BEGIN
    PalindromeCount := 0;
    FOR I := 1 TO MatrixSize
    DO
        BEGIN
            RowWord := '';
            ColWord := '';
            FOR J := 1 TO MatrixSize
            DO
                BEGIN
                    RowWord := RowWord + Matrix[I, J];
                    ColWord := ColWord + Matrix[J, I]
                END;
                AddIfUniquePalindrome(PalindromeCount, Palindromes, RowWord);
                AddIfUniquePalindrome(PalindromeCount, Palindromes, ColWord)
            END;
            MainDiagonal := '';
            SecondaryDiagonal := '';
            FOR I := 1 TO MatrixSize
            DO
                BEGIN
                    MainDiagonal := MainDiagonal + Matrix[I, I];
                    SecondaryDiagonal := SecondaryDiagonal + Matrix[I, MatrixSize - I + 1];
                END;
                AddIfUniquePalindrome(PalindromeCount, Palindromes, MainDiagonal);
                AddIfUniquePalindrome(PalindromeCount, Palindromes, SecondaryDiagonal)
            END;
        END;
    END;
```

Добавляем функцию проверки на уникальность

```
PROCEDURE AddIfUniquePalindrome(VAR PalindromeCount: INTEGER; VAR Palindromes: PalindromeArray; Str: STRING);
VAR
    I: INTEGER;
BEGIN
    IF NOT IsPalindrome(Str)
    THEN
        EXIT;
    FOR I := 1 TO PalindromeCount
    DO
        IF Palindromes[I] = Str
        THEN
            EXIT;
        PalindromeCount := PalindromeCount + 1;
        Palindromes[PalindromeCount] := Str
    END;
```

Добавляем функцию проверки на палиндром

```
FUNCTION IsPalindrome(Str: STRING): BOOLEAN;  
VAR  
    Left, Right: INTEGER;  
BEGIN  
    Left := 1;  
    Right := LENGTH(Str);  
    WHILE (Left < Right) AND (Str[Left] = Str[Right])  
    DO  
        BEGIN  
            Left := Left + 1;  
            Right := Right - 1  
        END;  
    IsPalindrome := Left >= Right  
END;
```


Что изменилось

Теперь мы можем:

- Проверить является ли строка палиндромом
- Проверить уникальность палиндрома
- Обработываем ошибки

Вопросы