Разбор задания №3 (9 кл)

Сигналы из космоса

Постановка задачи

Астрономы засекли странный радиосигнал, идущий из космоса. Первичный анализ выявил, что частоты сигнала являются элементами числовой последовательности трибоначчи. Астрофизик Петр объяснил, что это числовая последовательность, в которой первые три числа это 0, 0 и 1, а каждое последующее число равно сумме трёх предыдущих чисел.

$F_0 = 0,\ F_1 = 0, F_2 = 1,$

$F_n = F_{n-1} + F_{n-2} + F_{n-3},\ n >= 3$

Первые 10 элементов последовательности выглядят так: 0, 0, 1, 1, 2, 4, 7, 13, 24, 44.

Технику Бобу была поставлена задача написать программу, отделяющую сигналы из космоса, частоты которых входят в числовую последовательность от сигналов других частот.

Постановка задачи

- **Входные данные для программы**
  - Программа на вход принимает одно число N, такое что 0 <= N <= 500 000 000.
  - Примечание: для хранения целых чисел в таком диапазоне используйте тип LONGINT.
- **Выходные данные для программы**
  - В стандартный поток вывода программа должна вывести знак “+”, если N является членом последовательности трибоначчи и знак “-” если не является.
  - Если на вход поступили некорректные данные (не целое число или число за границами диапазона), программа должна вывести: ERROR

Примеры

- 1 -> +
- 4 -> +
- 6 -> -
- 13 -> +
- 19 -> -
- 44 -> +
- -5 -> ERROR
- one -> ERROR

0, 0, 1, 1, 2, 4, 7, 13, 24, 44, 81, 149,
274, 504, 927, 1705, 3136, 5768, …

Подзадачи

- Считать число, которое нам нужно проверить
- Проверить входит ли оно в последовательность
- Вывести результат

Числа Трибоначчи

```
PROGRAM Tribonacci;
VAR
  A, B, C, D, I: LONGINT;
BEGIN
  A := 0;
  B := 0;
  C := 1;
  WRITE(A, ' ', B, ' ', C, ' ');
  FOR I := 1 TO 20
  DO
    BEGIN
      D := A + B + C;
      WRITE(D, ' ');
      A := B;
      B := C;
      C := D
    END
END.
```

I	A	B	C
	0	0	1
1	0	1	1
2	1	1	2
3	1	2	4

Вывод:
0 0 1 1 2 4 7 13 24 44 81 149 274 504 927 1705 3136 5768 10609 19513 35890 66012 121415

Простейшее решение

```
PROGRAM IsTribonacci;
VAR
  A, B, C, D, N: LONGINT;
BEGIN
  READLN(N);
  A := 0;
  B := 0;
  C := 1;
  …
```

```
WHILE C <= N
  DO
    BEGIN
      IF C = N
      THEN
        BEGIN
          WRITELN('+');
          EXIT
        END;
      D := A + B + C;
      A := B;
      B := C;
      C := D
    END;
  WRITELN('-')
END.
```

1 -> +

4 -> +

6 -> -

13 -> +

19 -> -

44 -> +

-5 -> -

one -> Runtime error

Анализ решения

- Решена часть задачи
  - Ввод данных
  - Вывод результата
  - Проходит часть «хороших» тестов
- Что не сделано
  - Обработка ошибок
  - Обработка граничных ситуаций
    - 0 -> -

Обработка граничных ситуаций

Было

```
WHILE C <= N
  DO
    BEGIN
      IF C = N
      THEN
        BEGIN
          WRITELN('+');
          EXIT
        END;
      D := A + B + C;
      A := B;
      B := C;
      C := D
    END;
  WRITELN('-')
END.
```

Стало

```
WHILE A <= N
  DO
    BEGIN
      IF A = N
      THEN
        BEGIN
          WRITELN('+');
          EXIT
        END;
      D := A + B + C;
      A := B;
      B := C;
      C := D
    END;
  WRITELN('-')
END.
```

Ввод/вывод

1 -> +

4 -> +

6 -> -

13 -> +

19 -> -

44 -> +

-5 -> -

one -> Runtime error

0 -> +

500000000 -> -

500000001 -> -

Обработка ошибок

```
PROGRAM IsTribonacci;
VAR
  A, B, C, D, N: LONGINT;

FUNCTION ReadLongIntegerRange(VAR N: LONGINT; Min, Max: LONGINT): BOOLEAN;
VAR
  ErrorCode: INTEGER;
  Buffer: STRING;
BEGIN
  READLN(Buffer);
  VAL(Buffer, N, ErrorCode);
  ReadLongIntegerRange := (ErrorCode = 0) AND (N >= Min) AND (N <= Max)
END;
```

Обработка ошибок (продолжение)

```
BEGIN
  IF NOT ReadLongIntegerRange(N, 0, 500000000)
  THEN
    BEGIN
      WRITELN('ERROR');
      EXIT
    END;
  A := 0;
  B := 0;
  C := 1;
  …
END.
```

1 -> +
4 -> +
6 -> -
13 -> +
19 -> -
44 -> +
-5 -> ERROR
one -> ERROR
0 -> +
500000000 -> -
500000001 -> ERROR

Анализ решения

- + Проходит часть все «хорошие» тесты
- + Проходит все «плохие»
- - Несколько точек выхода из программы
- - Низкоуровневый код в основном теле программы

Выделяем функцию IsTribonacci

```
FUNCTION IsTribonacci(N: LONGINT): BOOLEAN;
VAR
  A, B, C, D: LONGINT;
BEGIN
  IsTribonacci := FALSE;
  A := 0;
  B := 0;
  C := 1;
  WHILE N >= A
  DO
    BEGIN
      IF N = A
      THEN
        IsTribonacci := TRUE;
      D := A + B + C;
      A := B;
      B := C;
      C := D
    END
END;
```

Выделяем функцию IsTribonacci

```
PROGRAM SignalsFromSpace;
VAR
  N: LONGINT;
…
BEGIN
  IF NOT ReadLongIntegerRange(N, 0, 500000000)
  THEN
    WRITELN('ERROR')
  ELSE
    IF IsTribonacci(N)
    THEN
      WRITELN('+')
    ELSE
      WRITELN('-')
END.
```

Анализ решения

- Проходит часть все «хорошие» тесты
- Проходит все «плохие»
- Одна точка выхода из программы
- Только высокоуровневый код в основном теле программы

Выводы

- Не пытайтесь написать всю программу сразу
  - Начните с простейшей работающей версии
- Прячьте сложный код внутри функций с понятными именами
- На каждом шаге проводите анализ своего решения
- Сперва обдумайте решение, а потом пишите код

Разбор задания №3 (11 кл)

Гай Юлий

Постановка задачи

Вы - разработчик в компании "Одногруппники". В полночь вам звонит менеджер проекта рыдает в трубку: "Все пропало!". Спустя 20 минут объяснений и 2 чашки ромашкового чая вы понимаете, что произошло.

Кто-то из пользователей, воспользовавшись недоработкой в системе безопасности загрузил на сервер вредоносный код. За пару часов он зашифровал все данные клиентов и теперь ни не прочесть. Хорошо, что у нас есть резервные копии некоторых файлов, по которым можно понять, как файл выглядел в начале и что с нити стало после шифрования. Вы изучаете зашифрованные файлы, успокаиваете менеджера и обещаете все починить в течение пары часов.

Внимательно посмотрев на зашифрованные файлы, вы определяете, что они закодированы с помощью модифицированного шифра Цезаря. Это один из самых простых алгоритмов шифрования, суть которого заключается в том, что каждый символ английского алфавита исходной последовательности циклически смещается на некоторое фиксированное расстояние. Остаётся только узнать величину смещения.

Постановка задачи

Строка "Zebra 12" со смещением на 1 символ будет выглядеть как "Afcsb 12" (каждая буква смещается на 1, а прочие символы остаются без изменения).

Разработайте программу, которая анализирует две строки из стандартного потока ввода: оригинальную и зашифрованную, подбирает смещение модифицированного шифра Цезаря и выводит его в виде числа в стандартный поток вывода. Смещение - число от 0 до 25.

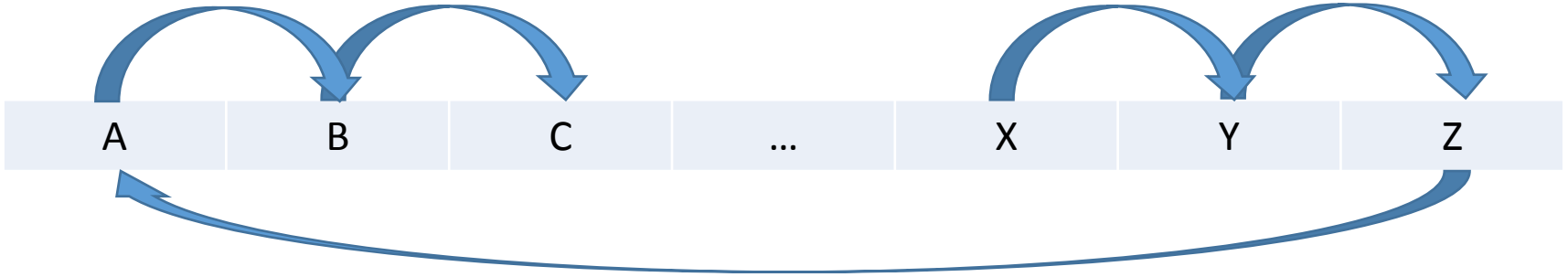
В зашифрованной строке все буквы английского алфавита циклически смещены на одно и то же значение.

Постановка задачи

- **Входные данные для программы**
  - Со стандартного потока ввода поступают две строки: исходная и зашифрованная.
- **Выходные данные для программы:**
  - Найденное смещение (число от 0 до 25). Если по входным данным невозможно определить смещение, вывести строку: NONE

Примеры

Ввод программы	Вывод программы
Apple Bqqmf	1
Cat Zebra 7 Ecv Bgdtc 7	2
12345 12345	NONE



Код символа

- Каждый символ в компьютере представляется его порядковым номером, называемым **кодом символа**
- Заглавные и строчные символы имеют разные коды
- Коды букв английского алфавита расположены последовательно
- В Pascal для получения кода символа используется функция ORD

Символ	A	B	Y	Z	a	b	y	z	0	1	8	9
Код	65	66	89	90	97	98	121	122	48	49	56	57

Простейшее решение

```
PROGRAM Ceasar;
VAR
  Original, Crypted: STRING;
BEGIN
  READLN(Original);
  READLN(Crypted);
  IF (Original[1] in ['A'..'Z']) OR (Original[1] in ['a'..'z'])
  THEN
    WRITELN(ORD(Crypted[1]) - ORD(Original[1]))
  ELSE
    WRITELN('NONE')
END.
```

Анализ решения

- Решена часть задачи
  - Ввод данных
  - Вывод результата
  - Проходит часть тестов
- Что не сделано
  - Проверяет только первый символ
  - Выводит отрицательное смещение при переходе через конец алфавита (к примеру при смещении Z в A выводит -25 вместо 1)

Ищем корректное смещение

```
PROGRAM Ceasar;
CONST
  AlphabetSize = ORD('Z') - ORD('A') + 1;
VAR
  Original, Crypted: STRING;

BEGIN
  READLN(Original);
  READLN(Crypted);
  IF (Original[1] in ['A'..'Z']) OR (Original[1] in ['a'..'z'])
  THEN
    WRITELN((ORD(Crypted[1]) - ORD(Original[1]) + AlphabetSize) MOD AlphabetSize)
  ELSE
    WRITELN('NONE')
END.
```

Анализ решения

- Решена часть задачи
  - Проходит больше тестов
- Что не сделано
  - Проверяет только первый символ
  - Сложный низкоуровневый код в основном теле программы

Выделяем функцию получения смещения

```
FUNCTION GetCeasarShift(Original, Crypted: STRING; VAR Shift: INTEGER): BOOLEAN;
CONST
  AlphabetSize = ORD('Z') - ORD('A') + 1;
VAR
  I, OriginalCode, CryptedCode: INTEGER;
BEGIN
  GetCeasarShift := FALSE;
  FOR I := 1 TO LENGTH(Original)
  DO
    IF (Original[I] in ['A'..'Z']) OR (Original[I] in ['a'..'z'])
    THEN
      BEGIN
        OriginalCode := ORD(Original[I]);
        CryptedCode := ORD(Crypted[I]);
        Shift := (CryptedCode - OriginalCode + AlphabetSize) MOD AlphabetSize;
        GetCeasarShift := TRUE;
        BREAK
      END
END;
```

Используем функцию получения смещения

```
PROGRAM Ceasar;
VAR
  Original, Crypted: STRING;
  Shift: INTEGER;

BEGIN
  READLN(Original);
  READLN(Crypted);

  IF GetCeasarShift(Original, Crypted, Shift)
  THEN
    WRITELN(Shift)
  ELSE
    WRITELN('NONE')
END.
```

Анализ решения

- Проходит все тесты
- Основное тело программы краткое и содержит только высокоуровневые операции
- Функция `GetCeasarShift` перегружена

Проверку, что символ является буквой, выносим в функцию IsAlpha

```
// Checks if Ch is an alphabetic symbol
FUNCTION IsAlpha(Ch: CHAR): BOOLEAN;
CONST
  Upper = ['A'..'Z'];
  Lower = ['a'..'z'];
BEGIN
  IsAlpha := (Ch IN Upper) OR (Ch IN Lower)
END;
```

Выделяем функцию вычисления позиции первой буквы в строке

```
// Returns first alphabet symbol index of the string
FUNCTION GetFirstAlphaIndex(Str: STRING): INTEGER;
VAR
  I: INTEGER;
BEGIN
  GetFirstAlphaIndex := 0;
  FOR I := 1 TO LENGTH(Str)
  DO
    IF IsAlpha(Str[I])
    THEN
      BEGIN
        GetFirstAlphaIndex := I;
        BREAK
      END
END;
```

Выделяем функцию расчёта смещения для двух символов

```
// Calculates the shift between two characters
FUNCTION CalculateCeasarShift(OriginalChar, CryptedChar: CHAR): INTEGER;
CONST
  AlphabetSize = ORD('Z') - ORD('A') + 1;
VAR
  OriginalCode, CryptedCode: INTEGER;
BEGIN
  OriginalCode := ORD(OriginalChar);
  CryptedCode := ORD(CryptedChar);
  CalculateCeasarShift := (CryptedCode - OriginalCode + AlphabetSize)
      MOD AlphabetSize
END;
```

Упрощаем код

```
// Returns Ceasar shift or FALSE if there are no alphabet characters
FUNCTION GetCeasarShift(Original, Crypted: STRING; VAR Shift: INTEGER): BOOLEAN;
VAR
  FirstAlphaIndex: INTEGER;
  OriginalChar, CryptedChar: CHAR;
BEGIN
  FirstAlphaIndex := GetFirstAlphaIndex(Original);
  IF FirstAlphaIndex > 0
  THEN
    BEGIN
      OriginalChar := Original[FirstAlphaIndex];
      CryptedChar := Crypted[FirstAlphaIndex];
      Shift := CalculateCeasarShift(OriginalChar, CryptedChar);
      GetCeasarShift := TRUE
    END
  ELSE
    GetCeasarShift := FALSE
END;
```

Анализ решения

- Проходит все тесты
- Код разбит на отдельные функции, которые выполняют небольшие задачи и хорошо читаются

Что ещё можно было бы улучшить?

- Сделать программу более устойчивой к некорректным входным данным
  - Пользователь ввёл 1 или 0 строк вместо двух
  - Зашифрованная и оригинальная строки содержат разное количество символов
  - Разные символы зашифрованной строки смещены на разное расстояние
  - Смещены не только символы английского алфавита

Спасибо за внимание!