Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ОТЧЕТ

по разминке №3

Тема работы: Крючки

Выполнил

студент: гр. 251003 Панкратьев Е.С.

Проверил: Фадеева Е.П.

Минск 2023

Содержание

[1 Постановка задачи 3](#_Toc136777778)

[2 Дополненная постановка задачи 4](#_Toc136777779)

[3 Метод решения 5](#_Toc136777780)

[3.1 Описание оператора try..except..end 5](#_Toc136777781)

[3.2 Условия ввода 5](#_Toc136777782)

[3.3 Проверка введенных данных 5](#_Toc136777783)

[3.4 Краткое описание алгоритма решения задачи 5](#_Toc136777784)

[4 Описание алгоритмов решения задачи 7](#_Toc136777785)

[5 Структура данных 8](#_Toc136777786)

[6 Результаты расчетов 9](#_Toc136777787)

[Приложение А 10](#_Toc136777788)

[Приложение Б 12](#_Toc136777789)

# Постановка задачи

Вводится количество крючков, из которых можно сделать «и» и «ш». Найти количество последовательностей данных символов, которые можно составить из данного количества крючков.

# Дополненная постановка задачи

Вводится количество крючков, из которых можно сделать «и» и «ш». Найти количество последовательностей данных символов и вывести сами последовательности, которые можно составить из данного количества крючков.

# Метод решения

## Описание оператора try..except..end

Оператор try..except..end имеет вид:

try  
 операторы;  
except  
  блок обработки исключений;  
end;

Выполнение блока начинается с секции try, при отсутствии исключительных ситуаций только она и выполняется. Секция except получает управление в случае возникновения исключительной ситуации. После обработки выполняются операторы, стоящие после end.

## Условия ввода

Условия ввода:

* количество крючков должно быть целочисленным значением;
* количество крючков должно быть в диапозоне [2..1000];

## Проверка введенных данных

Проверка введенных данных происходит с помощью цикла с предусловием repeat..until, чтобы при вводе некорректных данных пользователь заново заполнял их. В теле цикла с помощью оператора try..except..end (описание оператора см. [главу 3.1](#_Описание_оператора_try..except..end)) проверяем соответствие с типом данным; оператором if проверяет принадлежность заданному диапазону, где это необходимо.

## Краткое описание алгоритма решения задачи

Алгоритм решения задачи по поиску всех возможных последовательностей крючков:

Таблица 1 – Краткое описание алгоритма решения задачи

|  |  |
| --- | --- |
| Шаг | Действия |
| Базовый случай. Если количество оставшихся крючков равно 0 | Вывести текущую последовательность.  величить счетчик допустимых последовательностей |

Продолжение Таблицы 1

|  |  |
| --- | --- |
| Рекурсивный шаг. Если количество оставшихся крючков больше или равно определенного значения | Рекурсивно вызвать функцию, уменьшив количество оставшихся крючков на соответствующее значение.  Добавить соответствующий символ к текущей последовательности |
| Поиск последовательностей | Для каждого возможного значения количества крючков вызывать рекурсивно для поиска всех последовательностей |
| Вывод результатов | Вывести все найденные допустимые последовательности.  Вывести общее количество допустимых последовательностей. |

Алгоритм основан на применении рекурсии для перебора всех возможных комбинаций последовательностей крючков. На каждом шаге рекурсии проверяется, есть ли еще крючки для добавления к текущей последовательности. Если есть, то рекурсивно вызывается функция для поиска следующих комбинаций. Если остаются только символы "ш" или "и", то соответствующий символ добавляется к последовательности. Когда количество оставшихся крючков становится равным 0, достигается базовый случай, и текущая последовательность выводится на экран. После завершения поиска всех комбинаций выводятся результаты – найденные последовательности и общее количество допустимых последовательностей.

# Описание алгоритмов решения задачи

Таблица 2 – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п.п. | Наименование алгоритма | Название алгоритма | Формальные  параметры | Предпо-лагаемый тип реали-зации |
| 1 | Основной алгоритм | Вызов следующих подпрограмм:  OutputAllHooks и GetAmountOfHooks |  |  |
| 2 | OutputAllHooks  (  AHooksAmount,  ASequence,  ACount  ) | Находит количество возможных последовательностей букв и саму последовательность из AHooksAmount крючков. Последовательность записывает в ASequence, а количество в ACount | AHooksAmount – получает от фактического параметра адрес с защитой.  ASequence – получает от фактического параметра адрес с защитой.  ACount – получает от фактического параметра адрес | Проце-дура |
| 3 | GetAmountOf-Hooks  (  Res  ) | Получение положительного целочисленного числа от пользователя. Результат записывает в Res | Res – получает от фактического параметра адрес,  возвращаемый параметр | Функция. Res – возвра-щаемый функцией параметр |

# Структура данных

Таблица 3 – Структура данных основной программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| Count | Integer | Количество последовательностей |
| HooksAmount | Integer | Количество крючков |

Таблица 4 – Структура данных алгоритма OutputAllHooks(AHooksAmount, ASequence, ACount)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| AHooksAmount | Integer | Количество крючков | Формальный |
| ASequence | String | Последовательность | Формальный |
| ACount | Integer | Количество последовательностей | Формальный |

Таблица 5 – Структура данных алгоритма GetAmountOfHooks(Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Res | Integer | Введенное количество крючков | Формальный |
| IsCorrect | Boolean | Индикатор правильного ответа | Локальный |
| Min | Integer | Минимальное количество крючков | Локальный |
| Max | Integer | Максимальное количество крючков | Локальный |

# Результаты расчетов

В ходе работы были получены следующие результаты:

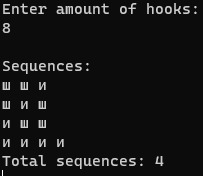


Рисунок 1 – Результаты расчетов

Приложение А

(обязательное)

Исходный код программы

program Recursion;

{

Finds the whole sequence of hooks

Finds the entire hook sequence. "и" gives 2 hooks, "ш" - 3

hooks

}

uses

SysUtils;

// Finds the whole sequence of hooks

procedure OutputAllHooks(const AHooksAmount: Integer; const

ASequence: string; var ACount: Integer);

begin

if AHooksAmount >= 3 then

// Recursively find sequences with three hooks

OutputAllHooks(AHooksAmount - 3, ASequence + 'ш ',

ACount);

if AHooksAmount >= 2 then

// Recursively find sequences with two hooks

OutputAllHooks(AHooksAmount - 2, ASequence + 'и ',

ACount)

else if AHooksAmount = 0 then

begin

// Output the sequence if there are no more hooks

Writeln(ASequence);

// Increment the count of valid sequences

Inc(ACount);

end;

end;

// Returns the correct number of hooks

function GetAmountOfHooks: Integer;

const

Min = 2;

Max = MaxInt;

// Maximum and minimum allowed value for the number of hooks

var

IsCorrect: Boolean;

// Flag to indicate if the input is correct

begin

Writeln('Enter amount of hooks: ');

// Repeat the loop until the input is correct

repeat

IsCorrect := True;

try

// Read the user input and assign it to the function

// result

Readln(Result);

except

// Output an error message for invalid input

IsCorrect := False;

Writeln('Incorrect input');

end;

if ((Result < Min) or (Result > Max)) and IsCorrect then

begin

IsCorrect := False;

// Output an error message for invalid input

Writeln('Incorrect input');

end;

until IsCorrect;

Writeln;

end;

var

Count: Integer;

HooksAmount : Integer;

begin

// Initialize the count of valid sequences

Count:= 0;

// Finds the whole sequence of hooks

HooksAmount := GetAmountOfHooks;

Writeln('Sequences:');

OutputAllHooks(HooksAmount, '', Count);

// Output sequence count

Writeln('Total sequences: ', count);

Readln;

end.

Приложение Б

(обязательное)

Тестовые наборы

**Тестовая ситуация: некорректный ввод данных**

**Тест 1**

Исходные данные: Некорректный ввод дат

Ожидаемый результат: Повторная попытка

Полученный результат:

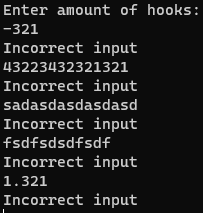


Рисунок 2 – Результаты расчетов

**Тестовая ситуация: корректный ввод данных**

Таблица 6 – Тестовые наборы

|  |  |  |
| --- | --- | --- |
| Тест | Рекурсивный вариант | Итерационный вариант |
| 1. |  |  |
| 2. |  |  |

Продолжение Таблицы 6

|  |  |  |
| --- | --- | --- |
| 3. |  |  |
| 4. |  |  |
| 5. |  |  |

Продолжение Таблицы 6

|  |  |  |
| --- | --- | --- |
| 6. |  |  |