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 разминочной задаче №1

Тема работы: Монахи и пироги

Выполнил

Проверил: Фадеева Е.П.

Минск 2021

содержание

[1 Постановка задачи 3](#_Toc88238251)

[2 Методика решения 4](#_Toc88238252)

[3 Структура данных 5](#_Toc88238253)

[4 Результаты расчетов 6](#_Toc88238254)

[Приложение А 7](#_Toc88238255)

[Приложение Б 10](#_Toc88238256)

# Постановка задачи

Есть три типа монахов: высшие, средние и низшие. Каждый монах ест такое количество пирогов, которое соответствует его чину. Пользователь вводит общее количество пирогов, количество всех монахов и общее количество всех съеденных пирогов всеми монахами каждого чина соответственно.

Вывести все возможные комбинации монахов каждой группы так, чтобы все условия выполнялись:

1. Количество всех монахов и количество монахов каждого типа есть числа целые и неотрицательные.
2. Количества съеденных монахами каждого типа пирогов могут быть как целыми, так и вещественными числами, но они должны быть больше нуля.
3. Монахи не могут голодать.
4. Монахи ранга ниже не могут есть больше пирогов, чем монахи ранга выше.
5. Программа учитывает возможный некорректный ввод пользователя.

# Методика решения

Вывод формул для нахождения количества обычных монахов и монахов-учеников. После вычислений количество обычных монахов необходимо округлить, так как оно не может быть дробным.

Cakes = Cakes1 × Rank1 + Cakes2 × Rank2 + Cakes3 × Rank3

Monks = Rank1 + Rank2 + Rank3

Cakes3 × Rank3 = Cakes - Cakes2 × Rank2 - Cakes1 × Rank1

Rank3 = Monks - Rank2 - Rank1

Rank3 = (Cakes - Cakes2 × Rank2 - Cakes1 × Rank1)/ Cakes3

Rank3 = Monks - Rank2 - Rank1

Monks - Rank2 - Rank1 = (Cakes - Cakes2 × Rank2 - Cakes1 × Rank1) / Cakes3

Rank3 = Monks - Rank2 - Rank1

Monks × Cakes3 - Rank2 × Cakes3 - Rank1 × Cakes3 = Cakes - Cakes2 × Rank2 - Cakes1 × Rank1

Rank3 = Monks - Rank2 - Rank1

Rank2 × Cakes3 - Rank2 × Cakes2 = Monks × Cakes3 - Rank1 × Cakes3 - Cakes + Cakes1 × Rank1

Rank3 = Monks - Rank2 - Rank1

Rank2 × (Cakes3 - Cakes2) = Monks × Cakes3 - Rank1 × Cakes3 - Cakes + Cakes1 × Rank1

Rank3 = Monks - Rank2 - Rank1

Rank2 = (Monks × Cakes3 - Rank1 × Cakes3 - Cakes + Cakes1 × Rank1) / (Cakes3 - Cakes2)

Rank3 = Monks - Rank2 - Rank1

# Структура данных

Таблица 2 – Данные

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| Cakes | Integer | Изначальное количество пирогов |
| Monks | Integer | Изначальное количество монахов |
| Rank1 | Integer | Количество высших монахов |
| Rank2 | Integer | Количество обычных монахов |
| Rank3 | Integer | Количество монахов-учеников |
| Cakes1 | Real | Сколько едят высшие монахи |
| Cakes2 | Real | Сколько едят обычные монахи |
| Cakes3 | Real | Сколько едят монахи-ученики |
| MaxRank1 | Integer | Максимальное количество высших монахов |

# Результаты расчетов

Вследствие результатов программы на экран выводятся следующие результаты расчетов:

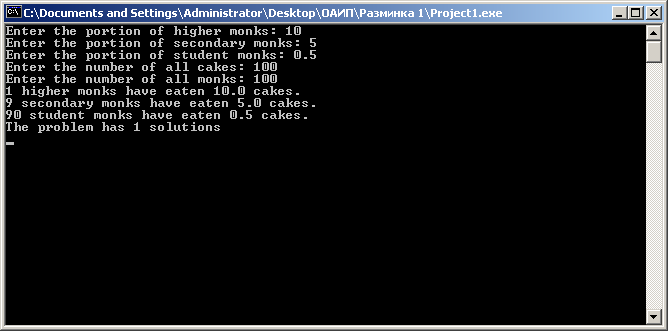


Рисунок 1 – Результаты расчетов

Приложение А

(обязательное)

Исходный код программы

Program Razminka1;

{This program calculates quantity of monks for every Rank. Initial data are portions, numbers of all monks and cakes.}

//Use app

{$APPTYPE CONSOLE}

//Declare modules

Uses

SysUtils;

//Declare modules

Var

Cakes1, Cakes2, Cakes3: Real;

Input: String;

Error, MaxRank1, Solutions, Rank1, Rank2, Rank3, Monks, Cakes: Integer;

//Cakes1, Cakes2, Cakes3 - quantity of cakes per portion

//Rank1, Rank2, Rank3 - quantity of monks per rank

//Solutions - quantity of solutions

//Input - input string for checking input

//Error - operator for checking input

//MaxRank1 - maximum quantity of higher monks

Begin

//Checking for the correct input

Repeat

Write('Enter the portion of higher monks: ');

ReadLn(Input);

Val(Input, Cakes1, Error);

If (Cakes1 <= 0) or (Error <> 0) then

WriteLn('Invalid Input. Enter another number.');

Until (Cakes1 > 0) and (Error = 0);

//Checking for the correct input

Repeat

Write('Enter the portion of secondary monks: ');

ReadLn(Input);

Val(Input, Cakes2, Error);

If (Cakes2 >= Cakes1) or (Cakes2 <= 0) or (Error <> 0) then

WriteLn('Invalid Input. Enter another number.');

Until (Cakes2 < Cakes1) and (Error = 0) and (Cakes2 > 0);

//Checking for the correct input

Repeat

Write('Enter the portion of student monks: ');

ReadLn(Input);

Val(Input, Cakes3, Error);

If (Cakes3 >= Cakes2) or (Cakes3 < 0) or (Error <> 0) then

WriteLn('Invalid Input. Enter another number.');

Until (Cakes3 < Cakes2) and (Error = 0) and (Cakes3 > 0);

//Checking for the correct input

Repeat

Write('Enter the number of all cakes: ');

ReadLn(Input);

Val(Input, Cakes, Error);

If (Cakes <= 0) or (Error <> 0) then

WriteLn('Invalid Input. Enter another number.');

Until (Cakes > 0) and (Error = 0);

//Checking for the correct input

Repeat

Write('Enter the number of all monks: ');

ReadLn(Input);

Val(Input, Monks, Error);

If (Monks <= 0) or (Error <> 0) then

WriteLn('Invalid Input. Enter another number.');

Until (Monks > 0) and (Error = 0);

MaxRank1:= Trunc(Cakes / Cakes1);

Solutions:= 0;

{This cycle starts with higher monks' number selection in range of [0, MaxRank1]. Number of monks calculates with formulas, which are described in the documentation.}

For Rank1:= 0 to MaxRank1 do

Begin

//Rank2 calculations

Rank2:= Round((Monks \* Cakes3 - Rank1 \* Cakes3 - Cakes + Cakes1 \* Rank1) / (Cakes3 - Cakes2));

If Rank2 >= 0 then

Begin

//Rank3 calculations

Rank3:= Monks - Rank1 - Rank2;

If Rank3 >= 0 then

Begin

If Cakes = Rank1 \* Cakes1 + Rank2 \* Cakes2 + Rank3 \* Cakes3 then

Begin

Solutions:= Solutions + 1;

If Solutions > 1 then

WriteLn('The second solution:');

WriteLn(Rank1, ' higher monks have eaten ', Cakes1:2:1, ' cakes.');

WriteLn(Rank2, ' secondary monks have eaten ', Cakes2:2:1, ' cakes.');

WriteLn(Rank3, ' student monks have eaten ', Cakes3:2:1, ' cakes.')

End

End

End

End;

//Next solution

If Solutions >= 1 then

WriteLn('The problem has ', Solutions, ' solutions')

//No solutions

Else

WriteLn('There is no solutions.');

ReadLn;

End.

Приложение Б

(обязательное)

Тестовые наборы

Тестовая ситуация 1: проверка ввода данных

Тестовая ситуация: проверка, сообщит ли программа о некорректном вводе, если введено отрицательное число или буквенное значение.

Исходные данные: Cakes1 = -4.

Ожидаемый результат: Invalid input. Enter another number.

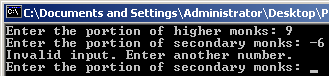
Полученный результат:



Исходные данные: Cakes2 = -6.

Ожидаемый результат: Invalid input. Enter another number.

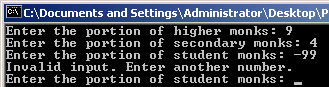
Полученный результат:



Исходные данные: Cakes3 = -99.

Ожидаемый результат: Invalid input. Enter another number.

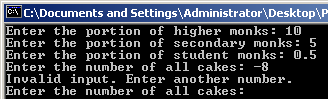
Полученный результат:



Исходные данные: Cakes = -8.

Ожидаемый результат: Invalid input. Enter another number.

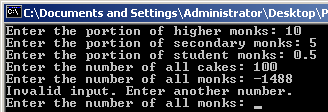
Полученный результат:



Исходные данные: Monks = -1488.

Ожидаемый результат: Invalid input. Enter another number.

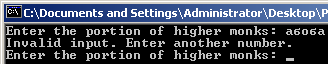
Полученный результат:



Исходные данные: Cakes1 = абоба.

Ожидаемый результат: Invalid input. Enter another number.

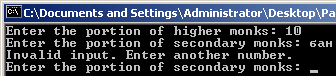
Полученный результат:



Исходные данные: Cakes2 = бан.

Ожидаемый результат: Invalid input. Enter another number.

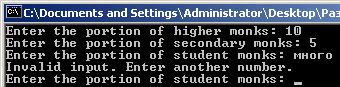
Полученный результат:



Исходные данные: Cakes3 = много.

Ожидаемый результат: Invalid input. Enter another number.

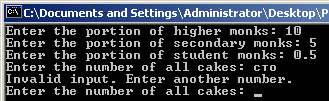
Полученный результат:



Исходные данные: Cakes = сто.

Ожидаемый результат: Invalid input. Enter another number.

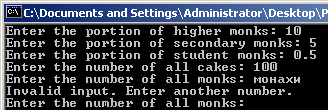
Полученный результат:



Исходные данные: Monks = монахи.

Ожидаемый результат: Invalid input. Enter another number.

Полученный результат:



Тестовая ситуация 3: проверка вычислений

Тестовая ситуация для проверки вывода правильных решений при количестве съеденных высшими монахами пирогов, равном десяти, количестве съеденных средними монахами пирогов, равном пяти, количестве съеденных низшими монахами пирогов, равном половине, количестве всех монахов, равном ста и количестве всех пирогов, равном ста.

Исходные данные: Cakes1 = 10; Cakes2 = 5; Cakes3 = 0.5; Cakes = 100; Monks = 100

Ожидаемый результат: 1 высший монах, 9 обычных и 90 студентов.

Полученный результат:

