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 лабораторной работе №6

Тема работы: Типы пользователя (диапозон, перечисление)

Выполнил

студент: гр. 251003 Панкратьев Е.С.

Проверил: Фадеева Е.П.

Минск 2022

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc122111952)

[2 Методика решения 4](#_Toc122111953)

[2.1 Описание оператора try..except..end 4](#_Toc122111954)

[2.2 Условия ввода 4](#_Toc122111955)

[2.3 Проверка введенных данных 4](#_Toc122111956)

[2.4 Краткое описание алгоритма решения задачи 4](#_Toc122111957)

[2.4.1 Использование пользовательских типов 4](#_Toc122111958)

[2.4.2 Нахождение количество понедельников на интервале 5](#_Toc122111959)

[3 Текстовый алгоритм решения задачи 6](#_Toc122111960)

[4 Структура данных 8](#_Toc122111961)

[5 Схема алгоритма решения задачи по ГОСТ 19.701-90 10](#_Toc122111962)

[Приложение А 17](#_Toc122111963)

[Приложение Б 25](#_Toc122111964)

# Постановка задачи

Дан временной период (число, месяц, год – число, месяц, год). Вычислить количество понедельников, приходящееся на 13 число.

# [Методика решения](#_Toc83996305)

## 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:

* даты должны быть целочисленными и положительными;
* количество дней должно соотвествовать месяцу;
* месяца должны быть в диапазоне [1..12]
* года должны быть в диапазоне [1..3000]

## 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 2.1](#_Описание_оператора_try..except..end)) проверяем целочисленность; оператором if проверяет принадлежность заданному диапазону.

## Краткое описание алгоритма решения задачи

### Использование пользовательских типов

Для удобства в решении данной задачи воспользуемся пользовательским типом данных (типы, хранящие допустимые дни в месяце, допустимые года, допустимые месяцы и допустимые дни недели). Также для ввода и вывода создадим константы ArrMonth для ввода и MonthName и DayOfWeekName для вывода. Также для удобства создадим массив AmountDaysInMonth, в котором номер индекса соответствует номеру месяца, а в элементе хранится количество дней в этом месяце.

Type

Tday = 1..31;

Tyear = 1..3000;

Tmonth = (Jan = 1, Feb, Mar, Apr, May, Jun, Jul, Aug,

Sep, Oct, Nov, Dec);

Tweek = (Monday = 1, Tuesday, Wednesday, Thursday,

Friday, Saturday, Sunday);

Const

AmountDaysInMonth:Array [Tmonth] of Tday =

(31,28,31,30,31,30,31,31,30,31,30,31);

ArrMonth:Array [Ord(Low(Tmonth))..Ord(High(Tmonth))]

of Tmonth = (Jan, Feb, Mar, Apr, May,

Jun, Jul, Aug, Sep, Oct, Nov, Dec);

MonthName:Array [Tmonth] of string = ('January', '

February', 'March', 'April', 'May',

'June', 'July', 'August', 'September',

'October', 'November', 'December');

DayOfWeekName: Array [Tweek] of string = ('Monday', '

Tuesday', 'Wednesday', 'Thursday',

'Friday', 'Saturday', 'Sunday');

### Нахождение количество понедельников на интервале

Перед нахождением количества понедельников на интервале найдем стартовый год Year, с которого будем начинать проход. Так как в григорианском календаре цикл равен 400 лет, то Year = 1+400\*(n-1) (где n-1 – это введенный год, уменьшенный на единицу, div 400). Так как первое января такого года – понедельник, дойдем до года и месяца из первой даты в интервале.

Идти будем помесячно, то есть будем находить первый понедельник каждого месяца. Чтобы это сделать, сначала с помощью цикла будем доходить до года из первой даты, ну а во вложенном цикле будем перебирать все месяца текущего года до конца, или если текущий год равен году из первой даты, то до месяца из этой же даты. Далее с помошью следующего цикла дойдем до понедельника, номер дня которого будет больше, чем номер дня из первой даты.

Далее будем идти с полученной даты до второй введенной даты. Перебирать будем так же, но уже будем перебирать каждый понедельник. Если текущая дата будет равна 13, то выводим ее, и увеличиваем счетчик количества найденных дней на единицу.

# Текстовый алгоритм решения задачи

Таблица 1 – Алгоритм решения

|  |  |
| --- | --- |
| Номер  шага | Назначение шага |
|  | Week = 7 |
|  | FourWeeks = Week\*4 |
|  | FiveWeeks = Week\*5 |
|  | AmountDaysInMonth[Tmonth] = (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31) |
|  | ArrMonth[Ord(Low(Tmonth))..Ord(High(Tmonth))] = (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec) |
|  | SearchedDayOfWeek = Monday |
|  | SearchedDayNum = 13 |
|  | CounterAmountSDN:= 0 |
|  | Ввод Day1, NumMonth, Year1 |
|  | Month1:= ArrMonth[NumMonth] |
|  | Ввод Day2, NumMonth, Year2 |
|  | Month2:= ArrMonth[NumMonth] |
|  | CurrDayNum:= Ord(SearchedDayOfWeek) |
|  | IsLeapYear:= False |
|  | CurrYear:= (400\*((Year1-1) div 400) + 1) |
|  | Начало цикла А1. Проверка выполнения условия (CurrYear <= Year1). Если условие истинно, перейти к шагу 17, иначе – к шагу 31 |
|  | CurrMonth:= Low(Tmonth) |
|  | IsLeapYear:= (CurrYear mod 4 = 0) and ((CurrYear mod 100 <> 0) or (CurrYear mod 400 = 0)) |
|  | Начало цикла А1.1. Проверка выполнения условий (CurrMonth <= High(Tmonth)) and ((CurrYear <> Year1) or (CurrMonth < Month1)). Если условия истинны, перейти к шагу 21, иначе – к шагу 29 |
|  | AmountDaysInCurrMonth:= AmountDaysInMonth[CurrMonth] |
|  | Если условия (IsLeapYear and (CurrMonth = Feb)) истинны, перейти к шагу 23. Иначе перейти к шагу 24 |
|  | Inc(AmountDaysInCurrMonth) |
|  | Если условие (AmountDaysInCurrMonth - CurrDayNum + 1 > FourWeeks) истинно, перейти к шагу 24. Иначе перейти к шагу 26 |
|  | CurrDayNum:= FiveWeeks - AmountDaysInCurrMonth + CurrDayNum |
|  | Перейтик шагу 27 |
|  | CurrDayNum:= CurrDayNum - (AmountDaysInCurrMonth - FourWeeks) |
|  | CurrMonth:= Succ(CurrMonth) |

Продолжение таблицы 1

|  |  |
| --- | --- |
|  | Конец цикла А1.1. Вернуться к шагу 19 |
|  | CurrYear:= CurrYear + 1 |
|  | Конец цикла А1. Вернуться к шагу 16 |
|  | Начало цикла А2. Проверка выполнения условия (CurrDayNum < Day1). Если условие истинно, перейти к шагу 32, иначе – к шагу 34 |
|  | Inc(CurrDayNum, Week) |
|  | Конец цикла А2. Вернуться к шагу 31 |
|  | CurrYear:= Year1 |
|  | Начало цикла А3. Проверка выполнения условия (CurrYear <= Year2). Если условие истинно, перейти к шагу 36, иначе – к шагу 53 |
|  | IsLeapYear:= (CurrYear mod 4 = 0) and ((CurrYear mod 100 <> 0) or (CurrYear mod 400 = 0)) |
|  | Начало цикла А3.1. Проверка выполнения условий (CurrMonth <= High(Tmonth)) and ((CurrYear <> Year2) or (CurrMonth <= Month2)). Если условия истинны, перейти к шагу 38, иначе – к шагу 50 |
|  | AmountDaysInCurrMonth:= AmountDaysInMonth[CurrMonth] |
|  | Если условия (IsLeapYear and (CurrMonth = Feb)) истинны, перейти к шагу 40. Иначе перейти к шагу 41 |
|  | Inc(AmountDaysInCurrMonth) |
|  | Начало цикла А3.2. Проверка выполнения условий (CurrDayNum <= AmountDaysInCurrMonth) and ((CurrYear <> Year2) or (CurrMonth <> Month2) or (CurrDayNum <= Day2)). Если условия истинны, перейти к шагу 42, иначе – к шагу 47 |
|  | Если условие (CurrDayNum = SearchedDayNum) истинно, перейти к шагу 43. Иначе перейти к шагу 45 |
|  | Вывод Ord(CurrMonth), CurrYear |
|  | Inc(CounterAmountSDN) |
|  | Inc(CurrDayNum, Week) |
|  | Конец цикла А3.2. Вернуться к шагу 41 |
|  | CurrDayNum:= CurrDayNum - AmountDaysInCurrMonth |
|  | CurrMonth:= Succ(CurrMonth) |
|  | Конец цикла А3.1. Вернуться к шагу 37 |
|  | CurrMonth:= Low(Tmonth) |
|  | CurrYear:= CurrYear + 1 |
|  | Конец цикла А3. Вернуться к шагу 35 |
|  | Вывод CounterAmountSDN |
|  | Останов. |

# Структура данных

Таблица 2 – Описанные скалярные типы

|  |  |  |
| --- | --- | --- |
| Описанные типы | Множество значений | Разновидность типа |
| Tday | 1..31 | Тип диапазон |
| Tyear | 1..3000 | Тип диапазон |
| Tmonth | (Jan = 1, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec) | Перечислимый скалярный тип |
| Tweek | (Monday = 1, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday) | Перечислимый скалярный тип |

Таблица 3 – Данные

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| Week | Byte | Количество дней в недели |
| FourWeeks | Byte | Количество дней в четырех неделях |
| FiveWeeks | Byte | Количество дней в пяти неделях |
| AmountDaysInMonth | Array [Tmonth] of Tday | Количество дней в месяце. |
| ArrMonth | Array [Ord(Low(Tmonth)).. Ord(High(Tmonth))] of Tmonth | Переводить из номера месяца в тип Tmonth |
| MonthName | Array [Tmonth] of string | Выводить пользователю месяц |
| DayOfWeekName | Array [Tweek] of string | Выводить пользователю день недели |
| SearchedDayOfWeek | Tweek | Искомый день недели |
| SearchedDayNum | Byte | Искомое число дня |
| Day1 | Tday | Введенный день из первой даты |
| Day2 | Tday | Введенный день из второй даты |
| Month1 | Tmonth | Введенный месяц из первой даты |
| Month2 | Tmonth | Введенный месяц из второй даты |
| Year1 | Tyear | Введенный год из первой даты |
| Year2 | Tyear | Введенный год из второй даты |

Продолжение таблицы 2

|  |  |  |
| --- | --- | --- |
| CurrMonth | Tmonth | Текущий месяц |
| CurrDayNum | Byte | Текущий день |
| AmountDaysIn  CurrMonth | Byte | Количество дней в текущем месяце |
| NumMonth | Byte | Переменная для ввода номера месяца |
| CounterAmount  SDN | Word | Счетчик для подсчета количество понедельников, приходящих на 13-ое число |
| CurrYear | Word | Текущий год |
| flag | Boolean | Идентификатор правильного ввода |
| IsLeapYear | Boolean | Идентификатор високосного года |

# Схема алгоритма решения задачи по ГОСТ 19.701-90



Рисунок 1 – Схема алгоритма решения задачи по ГОСТ 19.701-90 (часть 1)



Рисунок 2 – Схема алгоритма решения задачи по ГОСТ 19.701-90 (часть 2)



Рисунок 3 – Схема алгоритма решения задачи по ГОСТ 19.701-90 (часть 3)



Рисунок 4 – Схема алгоритма решения задачи по ГОСТ 19.701-90 (часть 4)



Рисунок 5 – Схема алгоритма решения задачи по ГОСТ 19.701-90 (часть 5)



Рисунок 6 – Схема алгоритма решения задачи по ГОСТ 19.701-90 (часть 6)



Рисунок 7 – Схема алгоритма решения задачи по ГОСТ 19.701-90 (часть 7)

Приложение А

(обязательное)

Исходный код программы

Program Lab6;

{

Enter the time period (Day1, Month1, Year1 - Day2, Month2, Year2). Count and print Mondays which coming on the 13th

}

{$APPTYPE CONSOLE}

uses

System.SysUtils;

Type

Tday = 1..31;

Tyear = 1..3000;

Tmonth = (Jan = 1, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep,

Oct, Nov, Dec);

Tweek = (Monday = 1, Tuesday, Wednesday, Thursday, Friday,

Saturday, Sunday);

// Tday - allowed amount of days in a month

// Tyear - allowable amount of years

// Tmonth - available months

// Tweek - available day of the week

Const

// !Immutable constants

Week = 7;

FourWeeks = Week\*4;

FiveWeeks = Week\*5;

AmountDaysInMonth:Array [Tmonth] of Tday =

(31,28,31,30,31,30,31,31,30,31,30,31);

ArrMonth:Array [Ord(Low(Tmonth))..Ord(High(Tmonth))] of

Tmonth = (Jan, Feb, Mar, Apr, May, Jun, Jul,

Aug, Sep, Oct, Nov, Dec);

MonthName:Array [Tmonth] of string = ('January', '

February', 'March', 'April', 'May', 'June',

'July', 'August', 'September','October',

'November', 'December');

DayOfWeekName: Array [Tweek] of string = ('Monday', '

Tuesday', 'Wednesday', 'Thursday', '

Friday', 'Saturday', 'Sunday');

// Week - amount of days in a week

// FourWeeks - amount of days in four weeks

// FiveWeeks - amount of days in five weeks

// AmountDaysInMonth - the element number corresponds to

// the month, and the content corresponds to the amount of

// days in the month

// ArrMonth - the element number corresponds to the month

// number, and the content corresponds to months (of Tmonth)

// MonthName - the element number corresponds to the month,

// and the content corresponds to the name of the month

// DayOfWeekName - the element number corresponds to the

// day of the week, and the content corresponds to the name

// of the day of the week

// !Mutable constants

SearchedDayOfWeek = Monday;

SearchedDayNum = 13;

// SearchedDayOfWeek - searched day of the week

// SearchedDayNum - the number of the searched day

Var

Day1, Day2 : Tday;

Month1, Month2 : Tmonth;

Year1, Year2 : Tyear;

CurrMonth : Tmonth;

CurrDayNum, AmountDaysInCurrMonth, NumMonth: Byte;

CounterAmountSDN, CurrYear: Word;

flag, IsLeapYear : boolean;

// Day1 - number for from the first date of the period

// Day2 - number for from the second date of the period

// Month1 - month for from the first date of the interval

// Month2 - month for from the second date of the interval

// Year1 - year for from the first date of the interval

// Year2 - year for from the second date of the interval

// NumMonth - entered month number

// CounterAmountSDN - counter amount searched day number

// CurrMonth - current month

// CurrDayNum - current day number

// AmountDaysInCurrMonth - amount of days in the current

// month

// CurrYear - current year

// flag - flag to confirm the correctness of entering

//numbers

// IsLeapYear - flag to determine if a month is found by

// its number+

Begin

Writeln('The day must match the month. The month must belong

to the interval ', Ord(Low(Tmonth)) , '..' , Ord

(High(Tmonth)),'.');

Writeln('The year must belong to the interval ',Ord(

Low(Tyear)),'..',Ord(High(Tyear)),'. Also, the

second date must be greater than the first');

Writeln('Enter your dates like this:');

Writeln('day month year');

Writeln('-');

Writeln('day month year');

Writeln;

// Initialization CounterAmountSDN

CounterAmountSDN:= 0;

// Cycle with postcondition for entering correct data.

repeat

// Initialize the flag and IsLeapYear

flag:= False;

IsLeapYear:= False;

// Validating the correct input data type

Try

Readln(Day1, NumMonth, Year1);

Except

Writeln('Incorrect type entry');

flag:= True;

End;

// flag check

if not flag then

begin

// Validate Range

if not (Day1 in [Low(Tday)..High(Tday)]) or not

(NumMonth in [Ord(Low(Tmonth))..Ord(High(Tmonth))])

or (Year1 < Low(Tyear)) or (Year1 > High(Tyear))

then

begin

Writeln('Incorrect input! They dont match the

required interval');

flag:= True;

end

// Otherwise, check the correspondence between the num

// ber of days and the month

else

begin

// Transfer from month number to month

Month1:= ArrMonth[NumMonth];

// Leap year definition

IsLeapYear := (Year1 mod 4 = 0) and ((Year1 mod 100

<> 0) or (Year1 mod 400 = 0));

// Check the correspondence between the number of

// days and the month

if IsLeapYear and (Month1 = Feb) and (Day1 > Amount

DaysInMonth[Month1] + 1) then

begin

Writeln('Since ',Year1,' is a leap year,

',MonthName[Month1],' has ',AmountDay

sInMonth[Month1] + 1,' days');

flag:= True;

end

else if Day1 > AmountDaysInMonth[Month1] then

begin

Writeln(MonthName[Month1],' has ', AmountDay

sInMonth[Month1],' days');

flag:= True;

end;

end;

end;

until not flag;

Writeln('-');

// Cycle with postcondition for entering correct data.

repeat

// Initialize the flag and IsLeapYear

flag:= False;

IsLeapYear:= False;

// Validating the correct input data type

Try

Readln(Day2, NumMonth, Year2);

Except

Writeln('Incorrect type entry');

flag:= True;

End;

// flag check

if not flag then

begin

// Validate Range

if not (Day1 in [Low(Tday)..High(Tday)]) or not

(NumMonth in [Ord(Low(Tmonth))..Ord(High ( Tmo

nth))])or (Year1 < Low(Tyear)) or (Year1 >

High(Tyear)) then

begin

Writeln('Incorrect input! They dont match the

required interval');

flag:= True;

end

// Otherwise, check the correspondence between the

//number of days and the month

else

begin

// Transfer from month number to month

Month2:= ArrMonth[NumMonth];

// Leap year definition

IsLeapYear:= (Year2 mod 4 = 0) and ((Year2 mod 100 <>

0) or (Year2 mod 400 = 0));

// Check the correspondence between the number of

// days and the month

if IsLeapYear and (Month2 = Feb) and (Day2 > Amount

DaysInMonth[Month2] + 1) then

begin

Writeln('Since ',Year2,' is a leap year,

',MonthName[Month2],' has ',AmountDay

sInMonth[Month2] + 1,' days');

flag:= True;

end

else if Day1 > AmountDaysInMonth[Month1] then

begin

Writeln(MonthName[Month2],' has ',AmountDay

sInMonth[Month2],' days');

flag:= True;

end

// Checking if the second date is greater than the

// first

else if (Year2 < Year1) or ((Year2 = Year1) and

(Ord(Month2) < Ord(Month1))) or ((Year2 =

Year1) and (Ord(Month2) = Ord(Month1)) and

(Day2 < Day1)) then

begin

Writeln('Time interval is wrong! The second data is

greater than the first');

flag:= True;

end;

end;

end;

until not flag;

// Decide on which days of the week will search for the

// dates of the days

CurrDayNum:= Ord(SearchedDayOfWeek);

// Initialize IsLeapYear

IsLeapYear:= False;

// Starting from the year (in which first of January is

// Monday), iterate through all the years

// up to the firs entered year (In this cycle, find the

// first SearchedDayOfWeek in Month1 Year1)

for CurrYear:= (400 \* ((Year1-1) div 400) + 1) to Year1 do

begin

// Reset CurrMonth to the first month

CurrMonth:= Low(Tmonth);

// Leap year definition

IsLeapYear:= (CurrYear mod 4 = 0) and ((CurrYear mod 100

<> 0) or (CurrYear mod 400 = 0));

// Iterate over all possible months or, if it is Year1,

// before the first entered month

while (CurrMonth <= High(Tmonth)) and ((CurrYear <>

Year1) or (CurrMonth < Month1)) do

begin

// Found the amount of days in the current month

AmountDaysInCurrMonth:= AmountDaysInMonth[CurrMonth];

if IsLeapYear and (CurrMonth = Feb) then

Inc(AmountDaysInCurrMonth);

// Depending on the amount of days between the current

// day and the amount of days

// in the current month, find the date of the first

// SearchedDayOfWeek of the next month

if AmountDaysInCurrMonth - CurrDayNum + 1 > FourWeeks

then

CurrDayNum:= FiveWeeks - AmountDaysInCurrMonth +

CurrDayNum

else

CurrDayNum:= CurrDayNum - (AmountDaysInCurrMonth –

FourWeeks);

// Find the current month for which found the date of

// the first SearchedDayOfWeek

CurrMonth:= Succ(CurrMonth);

end;

end;

// In this cycle, find the firts SearchedDayOfWeek after

// Day1

while CurrDayNum < Day1 do

Inc(CurrDayNum, Week);

Writeln;

Writeln('Found the following data coming on ',

DayOfWeekName[SearchedDayOfWeek],' the ',Searched

DayNum,'th:');

// Iterate from year to year 2 (in this cycle find all

// SearchedDayNum SearchedDayOfWeek

for CurrYear := Year1 to Year2 do

begin

// Leap year definition

IsLeapYear:= (CurrYear mod 4 = 0) and ((CurrYear mod 100

<> 0) or (CurrYear mod 400 = 0));

// Iterate over all possible months or, if it is Year2,

// before the second entered month

while (CurrMonth <= High(Tmonth)) and ((CurrYear <>

Year2) or (CurrMonth <= Month2)) do

begin

// Found the amount of days in the current month

AmountDaysInCurrMonth:= AmountDaysInMonth[CurrMonth];

if IsLeapYear and (CurrMonth = Feb) then

Inc(AmountDaysInCurrMonth);

// Iterate over all possible months or, if it is Year2

// and Month2, before the second entered month

while (CurrDayNum <= AmountDaysInCurrMonth) and ((Cur

rYear <> Year2) or (CurrMonth <> Month2) or

(CurrDayNum <= Day2)) do

begin

// The current number corresponds to SearchedDayNum,

// then display the current date

if CurrDayNum = SearchedDayNum then

begin

writeln(CurrDayNum,' ',Ord(CurrMonth),' ', Cur

rYear);

Inc(CounterAmountSDN);

end;

// Incrementing the current date by a week

Inc(CurrDayNum, Week);

end;

// Finding the first SearchedDayOfWeek in the next

// month

CurrDayNum:= CurrDayNum - AmountDaysInCurrMonth;

// Modernize CurrMonth

CurrMonth:= Succ(CurrMonth);

end;

// Reset CurrMonth to the first month

CurrMonth:= Low(Tmonth);

end;

Writeln('Finally found ',CounterAmountSDN,' dates');

Readln;

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:

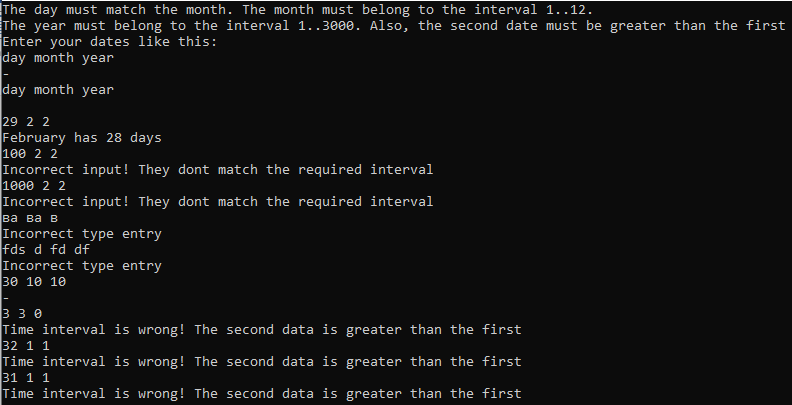


Рисунок 8 – Результаты расчетов

**Тестовая ситуация: корректный ввод данных**

**Тест 2**

Исходные данные: 1 1 1 – 10 4 10

Ожидаемый результат: 14 дат

Полученный результат:

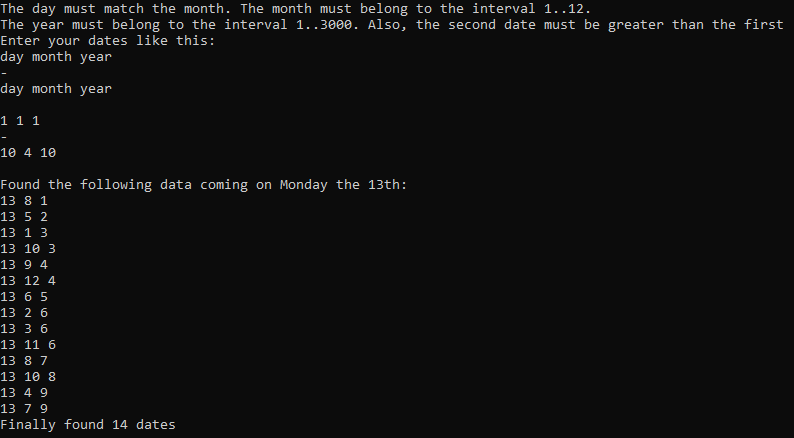


Рисунок 9 – Результаты расчетов

**Тест 3**

Исходные данные: 12 10 200 – 25 4 202

Ожидаемый результат: 3 дат

Полученный результат:

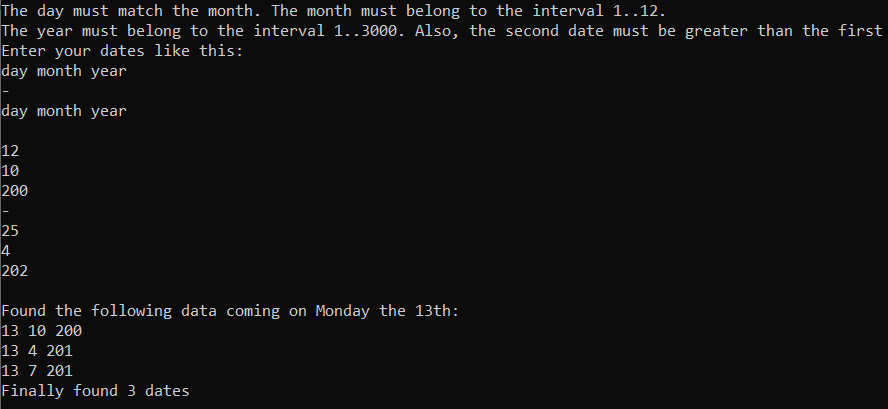


Рисунок 10 – Результаты расчетов

**Тест 4**

Исходные данные: 10 12 2000 – 10 4 2025

Ожидаемый результат: 41 дат

Полученный результат:

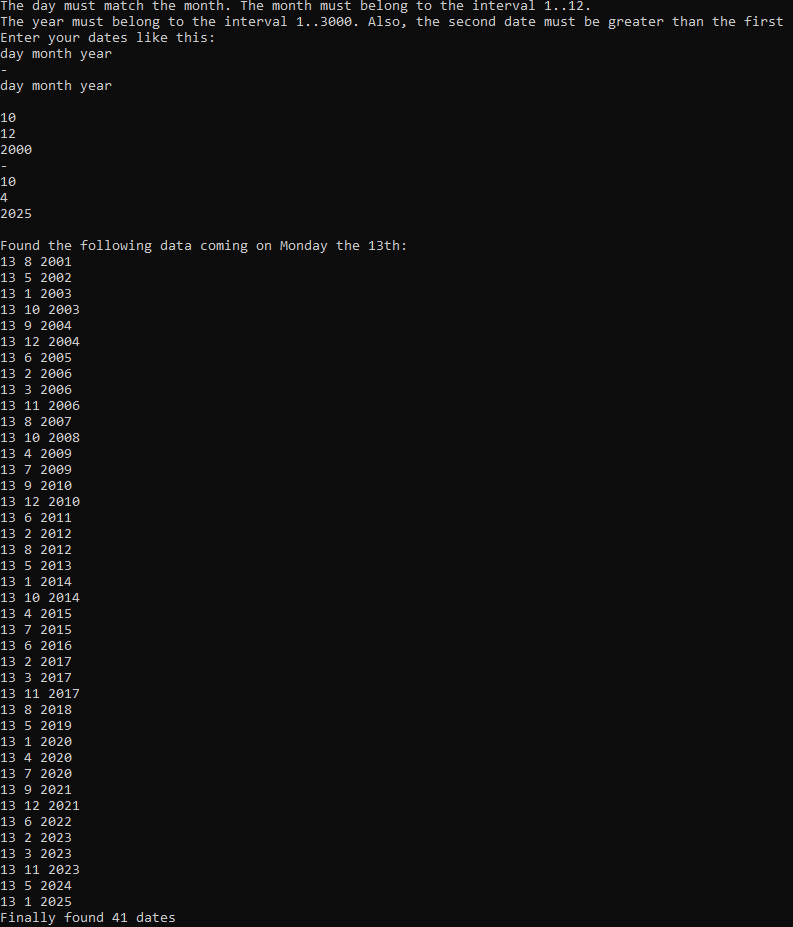


Рисунок 11 – Результаты расчетов

**Тест 5**

Исходные данные: 20 10 20 – 26 10 20

Ожидаемый результат: 0 дат

Полученный результат:

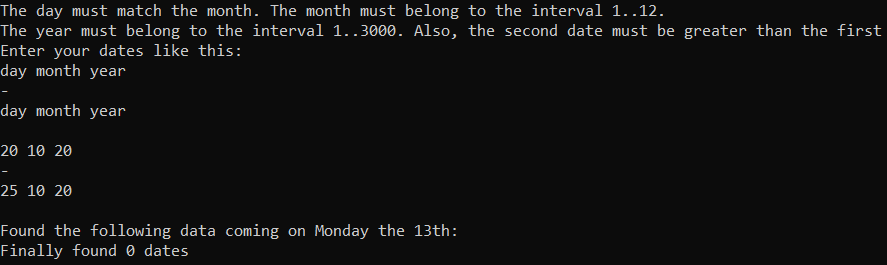


Рисунок 12 – Результаты расчетов

**Тест 6**

Исходные данные: 10 2 2800 – 12 3 2810

Ожидаемый результат: 16 дат

Полученный результат:

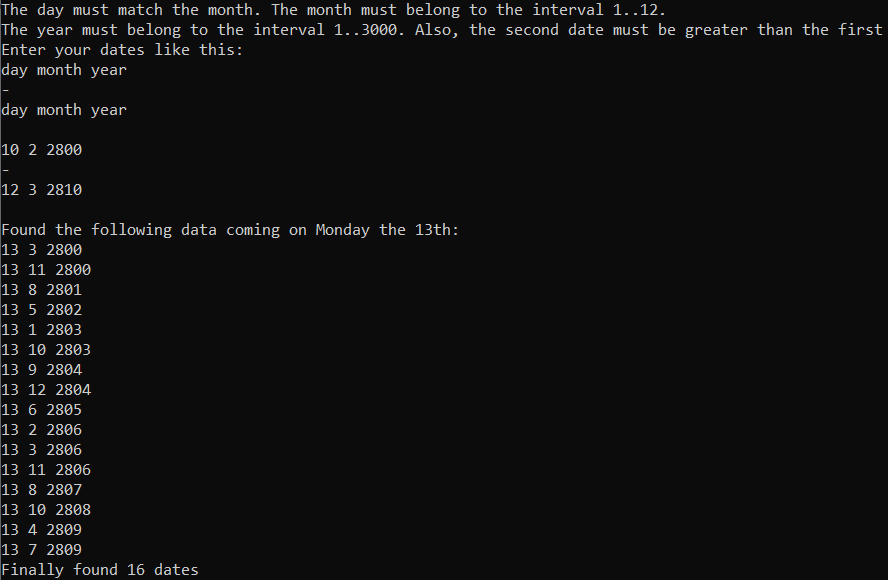


Рисунок 13 – Результаты расчетов