ТЕКСТ ПРОГРАММЫ “Easter”

Автор: Сагалов Даниил БПИ-196

```
format PE console
entry start

include 'win32a.inc'

;--------------------------------------------------------------------------
section '.data' data readable writable
    ; Информационные строки, выводимые на экран консоли.
    strInfo1     db 'The Grigorean calendar was first introduced in October 1582.',10,0
    strInfo2     db 'From that point of time, the calendar date of Easter could be determined correctly.',10,0
    strAskYear   db 'Please enter a valid year from 1582 to 2600.',10,0
    strWrongInput db 'Wrong year format. Please try again.',10,0
    strWrongFormat db 'Wrong input format, expected an integer. Terminating process...',10,0
    strReadKey     db 'Press any key to exit...',10,0
    strEasterApril db 'The Easter date for this year is the %dth of April.',10,0
    strEasterMarch db 'The Easter date for this year is the %dth of March.',10,0
    ; Эта строка является входным параметром для импортированной функции scanf
    ; и позволяет считать беззнаковое целое число, введённое пользователем.
    strReadYear  db '%u',0


    ; В этой переменной лежат исходные данные - год, введённый пользователем.
    year        dd ?
    ; В объявленных ниже переменных будут находится промежуточные данные алгоритма.
    G           dd ?
    C           dd ?
    X           dd ?
    Z           dd ?
    D           dd ?
    E           dd ?
    N           dd ?
```

```
;--------------------------------------------------------------------------

section '.code' code readable executable
start:
        ; Считываем год с консоли.
        call ReadInput
        ; Запускаем выполнение алгоритма.
        call CalculateEaster

finish:
        ; Выводим сообщение об ожидании ввода пользователя перед выходом из программы.
        push strReadKey
        call [printf]
        add esp, 4
        ; Ждём нажатия клавиши от пользователя, чтобы окно не закрылось мгновенно.
        call [getch]
        push 0
        call [ExitProcess]

;--------------------------------------------------------------------------
ReadInput:
        ; Выводим справочную информацию на экран.
        push strInfo1
        call [printf]
        add esp, 4
        push strInfo2
        call [printf]
        add esp, 4

askForInput:
        ; Запрашиваем ввод у пользователя.
        push strAskYear
        call [printf]
        add esp, 4
        ; Считываем пользовательский ввод.
```

```
        push year
        push strReadYear
        call [scanf]
        add esp, 8

        ; Проверка корректности входных данных.
        ; scanf возвращает число корректно считанных значений.
        ; Возвращаемое функцией scanf значение лежит в регистре eax.
        ; То есть, если scanf вернула 0, значит ввод был произведён некорректно
        ; (например, была введена строка, когда ожидалось число).
        cmp eax, 0
        je wrongFormat

        ; Делаем проверку входных данных. Год должен быть в промежутке [1582, 2600].
        cmp [year], 1582
        jl wrongInput ; Если введённый год < 1582.

checkUpperBound:
        ; Проверяем верхнюю границу.
        cmp [year], 2600
        jg wrongInput ; Если введённый год > 2600, то он находится вне допустимых значений.
        jmp endInput ; Если всё хорошо - выходим из подпрограммы.

wrongInput:
        ; Выводим сообщение об ошибке, если год находится вне допустимых значений.
        push strWrongInput
        call [printf]
        add esp, 4
        jmp askForInput

wrongFormat:
        ; Если ввод был произведён некорректно, то выводим сообщение об ошибке
        ; и завершаем выполнение программы.
        push strWrongFormat
        call [printf]
        add esp, 4
```

```
        jmp finish

endInput:
        ret

;-------------------------------------------------------------------------
CalculateEaster:

        call CalculateG ; 1-ый шаг алгоритма.
        call CalculateC ; 2-ой шаг алгоритма.
        call CalculateX ; 3-ий шаг алгоритма.
        call CalculateZ ; 4-ый шаг алгоритма.
        call CalculateD ; 5-ый шаг алгоритма.
        call CalculateE ; 6-ой шаг алгоритма.
        call StepSeven  ; 7-ой шаг алгоритма.
        call CalculateN ; 8-ой, 9-ый, 10-ый и 11-ый шаги алгоритма.

endAlg:
        ; Выход из подпрограммы по завершении выполнения алгоритма.
        ret

;-------------------------------------------------------------------------
CalculateG:

        mov eax, [year]
        mov ebx, 19
        xor edx, edx
        div ebx ; Делим year на 19. Остаток от деления будет лежать в edx.
        inc edx ; Увеличиваем остаток от деления на 1.
        mov [G], edx ; Записываем результат в G.

        ret

;-------------------------------------------------------------------------
CalculateC:
```

```
        mov eax, [year]
        mov ebx, 100
        xor edx, edx
        div ebx ; Делим значение в переменной year на 100.
        inc eax ; Прибавляем 1.
        mov [C], eax ; Сохраняем номер века в переменной C.

        ret

;-----------------------------------------------------------------------
CalculateX:

        mov eax, [C]
        mov ebx, 3
        mul ebx ; Теперь в eax будет лежать 3*C.
        mov ebx, 4
        xor edx, edx
        div ebx ; Делим 3*C на 4. Целая часть лежит в eax.
        sub eax, 12 ; eax -= 12.
        mov [X], eax ; Сохраняем результат в переменной X.

        ret

;-----------------------------------------------------------------------
CalculateZ:

        mov eax, [C]
        mov ebx, 8
        mul ebx ; Теперь в eax будет лежать 8*C.
        add eax, 5
        mov ebx, 25
        xor edx, edx
        div ebx ; (8C + 5)/25
        sub eax, 5 ; Из целой части от деления вычитаем 5.
        mov [Z], eax
```

```
        ret

;--------------------------------------------------------------------------
CalculateD:

        mov eax, [year]
        mov ebx, 5
        mul ebx ; Теперь в eax будет лежать 5*Y.
        mov ebx, 4
        xor edx, edx
        div ebx ; 5Y/4. Целая часть от деления лежит в eax.
        sub eax, [X] ; Вычитаем X.
        sub eax, 10 ; Вычитаем 10.
        mov [D], eax ; Результат записываем в переменную D.

        ret

;--------------------------------------------------------------------------
CalculateE:

        mov eax, [G]
        mov ebx, 11
        mul ebx ; В eax будет лежать 11G.
        add eax, 20 ; 11G + 20
        add eax, [Z] ; 11G + 20 + Z
        sub eax, [X] ; 11G + 20 + Z - X
        mov ebx, 30
        xor edx, edx
        div ebx ; То, что в eax, делим на 30 и берём остаток.
        add edx, 30 ; К остатку от деления прибавляем 30.
        mov eax, edx
        mov ebx, 30
        xor edx, edx
        div ebx ; Последний раз делим на 30 и берём остаток.
        mov [E], edx ; Записываем его в переменную E.
```

```
        ret

;-------------------------------------------------------------------------
StepSeven:

        cmp [E], 24
        je incrementE ; Если E == 24, то нужно увеличить E на 1.
        cmp [E], 25
        je checkG ; Если E == 25, то нужно проверить второе условие.
        jmp finishStepSeven ; Иначе - выходим из подпрограммы.

checkG:
        cmp [G], 11 ; Сравниваем G с 11.
        jb incrementE ; Если G > 11 И E == 25, то увеличиваем E на 1.
        jmp finishStepSeven ; Иначе - выходим из подпрограммы.

incrementE:
        add [E], 1 ; Увеличиваем E на 1 при необходимости.

finishStepSeven:
        ret

;-------------------------------------------------------------------------
CalculateN:

stepEight:
        ; 8-ой шаг алгоритма.
        mov eax, 44
        mov ebx, [E]
        sub eax, ebx ; Вычитаем E из 44.
        mov [N], eax ; Сохраняем результат в N.

stepNine:
        ; 9-ый шаг алгоритма
        cmp [N], 21
        jl addThirty ; Если N < 21, то переходим к нужной метке.
```

```
        jmp stepTen ; Иначе сразу переходим на следующий шаг алгоритма.

addThirty:
        add [N], 30 ; Увеличиваем N на 30.

stepTen:
        ; 10-ый шаг алгоритма
        add [N], 7 ; N += 7.
        mov eax, [D] ; Перемещаем значение переменной D в регистр eax.
        add eax, [N] ; D += N.
        mov ebx, 7
        xor edx, edx
        div ebx ; Делим D + N на 7 и берём остаток.
        sub [N], edx ; Вычитаем остаток от деления из N + 7.

stepEleven:
        ; 11-ый шаг алгоритма
        cmp [N], 31 ; Сравниваем N с 31.
        jg isApril ; Если N > 31, то дата Пасхи будет в апреле.
        jmp isMarch ; Иначе - в марте.

isApril:
        sub [N], 31 ; N -= 31.
        push [N]
        push strEasterApril
        call [printf] ; Выводим информацию на экран.
        add esp, 8
        jmp endAlgorithm ; Выходим из подпрограммы.

isMarch:
        push [N]
        push strEasterMarch
        call [printf] ; Выводим информацию на экран.
        add esp, 8

endAlgorithm:
```

```
        ret

;--------------------------------Import section--------------------------

section '.idata' import data readable
    library kernel, 'kernel32.dll',\
            msvcrt, 'msvcrt.dll',\
            user32,'USER32.DLL'

include 'api\user32.inc'
include 'api\kernel32.inc'
    import kernel,\
           ExitProcess, 'ExitProcess'

  include 'api\kernel32.inc'
    import msvcrt,\
           printf, 'printf',\
           scanf, 'scanf',\
           getch, '_getch'
```