НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЕТ

по лабораторной работе №2

по дисциплине

Программирование на языках низкого уровня в задачах защиты информации

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дмитриев Д.В.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Голубев Д.А.

М18-ИСТ-1

Нижний Новгород

2019 г.

# Цель работы:

Вариант 2. В числовом двумерном массиве вычислить среднее арифметическое максимальных элементов столбцов матрицы.

# Код программы:

# .686P

# .model flat, stdcall

# option casemap:none

# include ..\irvine\Irvine32.inc

# include ..\masm32\include\msvcrt.inc

# includelib ..\irvine\User32.lib

# includelib ..\irvine\Kernel32.lib

# includelib ..\irvine\Irvine32.lib

# includelib ..\masm32\lib\msvcrt.lib

# ; ограничение на размер массива

# maxSize = 100

# ; инициализация массива

# InitMat PROTO,

# lpArray:PTR DWORD

# ; сумма максимальных элементов по столбцам

# ArithmeticMeanOfColumns PROTO,

# lpArray:PTR DWORD,

# mCols:DWORD,

# mRows:DWORD

# ; печать матрицы

# PrintMat PROTO,

# lpArray:PTR DWORD,

# mRows:DWORD,

# mCols:DWORD

# 

# .data?

# rows DWORD ? ; число строк

# cols DWORD ? ; число столбцов

# matrixSize DWORD ? ; количество элементов

# vSum DWORD ?

# vAmountOfEl DWORD ?

# 

# vArithMeanInt DWORD ?

# vArithMeanRem DWORD ?

# Array dword maxSize dup(?)

# .const

# maxRandNumb dword 100

# sConsoleTitle byte "Lab 2 ASM", 13, 10, 0

# sSize byte "Enter a Mat size: ", 0

# sMat byte "Matrix [%d x %d]: ", 13, 10, 0

# sFormat byte " %d",0

# sSpace byte " ", 0

# sError byte "Err", 13, 10, 0

# sSum byte "Sum: %d", 13, 10, 0

# sAmountOfEl byte "Amount of elements: %d", 13, 10, 0

# 

# sArithMean byte "Arithmetic Mean Of Columns: %d.%d", 13, 10, 0

# sExit byte "Press any key to Exit", 13, 10, 0

# 

# fm db '%d, ',0

# 

# 

# .code

# main PROC

# ; заголовок(title) консоли

# invoke SetConsoleTitleA, addr sConsoleTitle

# invoke InitMat, addr Array

# invoke crt\_printf, addr sMat, rows, cols

# invoke PrintMat, addr Array, rows, cols

# invoke ArithmeticMeanOfColumns, addr Array, cols, rows

# ; exit

# invoke crt\_printf, addr sExit

# ; \_getch()

# invoke crt\_\_getch

# invoke ExitProcess, 0

# 

# main ENDP

# InitMat PROC,

# lpArray: PTR DWORD

# invoke crt\_printf, addr sSize

# invoke ReadInt

# ; проверки ввода

# jo error ; ошибка в вводе(проверка на переполнение)

# cmp eax, 0

# jle error ; меньше или равно нулю

# cmp eax, maxSize

# jg error ; превышает maxSize

# 

# mov rows, eax

# mov cols, eax

# 

# mul eax ; умножение нa себя(получаем количество элементов)

# mov matrixSize, eax ; заносим размер матрицы

# ; готовим регистры для цикла заполнения

# mov ebx, lpArray ; заносим указатель на первый элемент в регистр

# mov esi, 0 ; устанавливаем на 0

# mov ecx, matrixSize ; заносим размер в регистр-счетчик

# 

# ; ГПСЧ

# ;invoke Randomize ; инициализурем генератор случайных чисел

# InitVal:

# mov eax, maxRandNumb ; параметр для RandomRange

# invoke RandomRange ; генерируем случайное число

# mov [ebx + esi], eax ; сохраняет сгенерированное значение

# add esi, TYPE lpArray ; к след элементу

# loop InitVal

# ret

# ; переход при ошибке

# error:

# invoke crt\_printf, addr sError

# invoke ExitProcess, 0

# 

# InitMat ENDP

# ArithmeticMeanOfColumns PROC,

# lpArray:PTR DWORD, ; адрес массива

# mCols:DWORD, ; количество элементов

# mRows:DWORD

# LOCAL shift

# mov eax, mCols

# imul eax, TYPE lpArray

# mov shift, eax

# mov eax, 0 ; обнуляем сумму

# mov ebx, 0 ; смещение элемента столбца в строке

# mov ecx, mCols ; количество столбцов

# cycleCols:

# push ecx ; сохраняем счетчик

# mov ecx, mRows ; счетчик элементов в столбце

# mov edx, Array[ebx] ; заносим первый элемент столбца

# mov esi, shift ; смещение второго элемента столбца(cols\*TYPE arr)

# cycleEl:

# cmp edx, Array[ebx] + [esi] ; сравниваем

# jge next ; если больше или равно - к следующему

# mov edx, Array[ebx] + [esi] ; если меньше, то сохранили

# next:

# add esi, shift ; переходим к следующему элементу

# loop cycleEl ; цикл по элементам столбца

# add eax, edx ; просуммировали максимальный элемент

# pop ecx ; восстановили счетчик

# add ebx, TYPE lpArray ; перешли к следующему столбцу

# loop cycleCols ; цикл по столбцам

# mov vSum, eax ; заносим сумму

# mov edx, 0 ; хранит старшие биты делимого, в eax младшие

# mov ebx, mCols ; количество элементов, на которые будем делить

# idiv ebx

# 

# mov vArithMeanRem, edx ; остаток

# mov vArithMeanInt, eax ; целое

# 

# 

# ; вывод

# invoke crt\_printf, ADDR sSum, vSum

# invoke crt\_printf, ADDR sAmountOfEl, mCols

# invoke crt\_printf, ADDR sArithMean, vArithMeanInt, vArithMeanRem

# ret

# ArithmeticMeanOfColumns ENDP

# PrintMat PROC,

# lpArray:PTR DWORD,

# mRows:DWORD,

# mCols:DWORD

# pushad

# mov ebx, lpArray ; указатель на первый элемент

# mov esi, 0 ; устанавливаем на 0

# mov ecx, mRows ; счетчик внешнего цикла

# loopCols:

# push ecx ; сохраняем счетчик внешнего цикла

# mov ecx, mCols ; счетчик вложенного цикла

# 

# LoopRows:

# ; выводит число

# mov eax,[ebx + esi]

# invoke WriteInt

# ; следующее число

# add esi, TYPE lpArray

# ; печатает пробел между числами массива

# mov edx, offset sSpace

# invoke WriteString

# loop LoopRows

# 

# invoke Crlf ; переход на новую строку

# pop ecx ; восстанавливаем внешний счетчикX

# loop loopCols

# popad

# ret

# PrintMat ENDP

# END main

# Результат выполнения программы

