МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Волжский политехнический институт (филиал)

федерального государственного бюджетного образовательного учреждения высшего образования

«Волгоградский государственный технический университет»

(ВПИ (филиал) ВолгГТУ)

|  |  |
| --- | --- |
| Факультет | Инженерно-экономический факультет |
| Кафедра | Информатика и технология программирования |

##### **КОНТРОЛЬНАЯ РАБОТА**

|  |  |
| --- | --- |
| по дисциплине: | Машинно-зависимые языки |
| на тему: | Разработка приложения на языке Assembler |
| Вариант № 12 | |

|  |  |
| --- | --- |
| Студент | Редняков Вячеслав Александрович |
|  | (фамилия, имя, отчество) |

|  |  |
| --- | --- |
| Группа | ВВТ-206 |
|  |  |

|  |  |
| --- | --- |
| Оценка |  |
|  | (баллы) |

|  |  |  |  |
| --- | --- | --- | --- |
| Проверил |  |  | доцент, Абрамова О.Ф. |
|  | (подпись и дата подписания) |  | (долж., фамилия и инициалы) |

|  |  |  |  |
| --- | --- | --- | --- |
| Специалист по УМР |  |  | Инкарбекова М.Р. |
|  | (подпись и дата подписания) |  | (фамилия и инициалы) |

Волжский, 2019

# Задание на контрольную работу

Программа должна быть разработана с использованием WinAPI функций в среде WinASM. Текст программы должен содержать комментарии, описывающие основные функции.

1. Создать заголовок программы, содержащий название дисциплины, фамилию и имя автора, номер варианта.
2. Разработать процедуры отображения графика форма которого указана в таблице вариантов. На форме должно присутствовать изображение формулы, определяющей форму графика. Диапазон изменения переменной должен задаваться в строке ввода. Также должен быть реализован выбор вида графика: точки или непрерывная линия.
3. Создать цветную диаграмму () на основе методов рисования в соответствии с параметрами варианта. Элементы диаграммы должны отличаться цветом. Диаграмма должна состоять из 3-4-х элементов, образованных элементами заданной формы.

**Вариант №12**

1) 

2) 

3) Реализовать управление позицией круга (лево-право) клавишами с клавиатуры

**Исходный текст программного средства**

**Задание № 1**

.686

.model flat,stdcall

option casemap:none

WinMain proto :DWORD,:DWORD,:DWORD,:DWORD

count proto

include windows.inc

include user32.inc

include kernel32.inc

include gdi32.inc

include masm32.inc

include shlwapi.inc

includelib user32.lib

includelib kernel32.lib

includelib gdi32.lib

includelib masm32.lib

includelib shlwapi.lib

.data

ps PAINTSTRUCT <>

rc RECT <>

brushBack dd ?

hdc dd ?

ClassName db "SimpleWinClass",0

ClassName1 db "OtherWinClass",0

AppName db "Контрольная работа по дисциплине Машинно-зависимые языки, Редняков Вячеслав, Вариант №12",0

AppNam1e db "График",0

LineName db "Линия",0

PointName db "Точки",0

ButtonClassName db "button",0

ButtonText db "Построить",0

EditClassName db "edit",0

STATIC db "STATIC",0

ame1 db "Минимум x:",0

ame2 db "Максимум x:",0

ame3 db "Шаг",0

flg db 0

repp dd ?

step dq ?

ScaleX dd ?

ScaleY dd ?

intX dd ?

intY dd ?

minX dd ?

maxX dd ?

minY dd -250000

maxY dd 250000

OffsetX dd ?

hwnd1 dd ?

OffsetY dd ?

hInstance HINSTANCE ?

CommandLine LPSTR ?

hwndButton HWND ?

hwndEditMinX HWND ?

hwndEditMaxX HWND ?

hwndEditStep HWND ?

tmp dd ?

buffer db 25 dup(?)

a dq ?

x dq ?

.const

ButtonID equ 1

LINE\_RB equ 131

POINT\_RB equ 132

EditID equ 2

IDM\_EXIT equ 4

.code

start:

invoke GetModuleHandle, NULL

mov hInstance,eax

invoke WinMain, hInstance,NULL,NULL,NULL

invoke ExitProcess,eax

WinMain proc hInst:HINSTANCE,hPrevInst:HINSTANCE,CmdLine:LPSTR,CmdShow:DWORD

LOCAL wc:WNDCLASSEX

LOCAL msg:MSG

LOCAL hwnd:HWND

LOCAL wc1:WNDCLASSEX

;Регистрируем класс доп. окна

mov wc1.cbSize,SIZEOF WNDCLASSEX

mov wc1.style, CS\_HREDRAW or CS\_VREDRAW

mov wc1.lpfnWndProc, OFFSET WndProc1

mov wc1.cbClsExtra,NULL

mov wc1.cbWndExtra,NULL

push hInstance

pop wc1.hInstance

mov wc1.hbrBackground, 6

mov wc1.lpszMenuName,0

mov wc1.lpszClassName,OFFSET ClassName1

invoke LoadIcon,NULL,IDI\_APPLICATION

mov wc1.hIcon,eax

invoke LoadCursor,NULL,IDC\_ARROW

mov wc1.hCursor,eax

invoke RegisterClassEx, addr wc1

;Регистрируем класс основного окна

mov wc.cbSize,SIZEOF WNDCLASSEX

mov wc.style, CS\_HREDRAW or CS\_VREDRAW

mov wc.lpfnWndProc, OFFSET WndProc

mov wc.cbClsExtra,NULL

mov wc.cbWndExtra,NULL

push hInst

pop wc.hInstance

mov wc.hbrBackground, 5

mov wc.lpszMenuName,0

mov wc.lpszClassName,OFFSET ClassName

invoke LoadIcon,NULL,IDI\_APPLICATION

mov wc.hIcon,eax

invoke LoadCursor,NULL,IDC\_ARROW

mov wc.hCursor,eax

invoke RegisterClassEx, addr wc

invoke CreateWindowEx,WS\_EX\_CLIENTEDGE,ADDR ClassName,ADDR AppName,\

WS\_OVERLAPPEDWINDOW,CW\_USEDEFAULT,\

CW\_USEDEFAULT,400,300,NULL,NULL,\

hInst,NULL

mov hwnd,eax

invoke ShowWindow, hwnd,SW\_SHOWNORMAL

invoke UpdateWindow, hwnd

.WHILE TRUE

invoke GetMessage, ADDR msg,NULL,0,0

.BREAK .IF (!eax)

invoke TranslateMessage, ADDR msg

invoke DispatchMessage, ADDR msg

.ENDW

mov eax,msg.wParam

ret

WinMain endp

WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM

.IF uMsg==WM\_DESTROY

invoke PostQuitMessage,NULL

ret

.ELSEIF uMsg==WM\_CREATE

; надпись "Минимум x:"

invoke CreateWindowEx,WS\_EX\_LEFT, ADDR STATIC,ADDR ame1,\

WS\_CHILD or WS\_VISIBLE,\

5,5,100,25,hWnd,NULL,hInstance,NULL

; надпись "Максимум x:"

invoke CreateWindowEx,WS\_EX\_LEFT, ADDR STATIC,ADDR ame2,\

WS\_CHILD or WS\_VISIBLE,\

5,35,100,25,hWnd,NULL,hInstance,NULL

; надпись "Шаг"

invoke CreateWindowEx,WS\_EX\_LEFT, ADDR STATIC,ADDR ame3,\

WS\_CHILD or WS\_VISIBLE,\

5,65,100,25,hWnd,NULL,hInstance,NULL

invoke CreateWindowEx,WS\_EX\_CLIENTEDGE, ADDR EditClassName,NULL,\

WS\_CHILD or WS\_VISIBLE or WS\_BORDER or ES\_LEFT or\

ES\_AUTOHSCROLL,\

95,5,100,25,hWnd,EditID,hInstance,NULL

mov hwndEditMinX,eax

invoke CreateWindowEx,WS\_EX\_CLIENTEDGE, ADDR EditClassName,NULL,\

WS\_CHILD or WS\_VISIBLE or WS\_BORDER or ES\_LEFT or\

ES\_AUTOHSCROLL,\

95,35,100,25,hWnd,EditID,hInstance,NULL

mov hwndEditMaxX,eax

invoke CreateWindowEx,WS\_EX\_CLIENTEDGE, ADDR EditClassName,NULL,\

WS\_CHILD or WS\_VISIBLE or WS\_BORDER or ES\_LEFT or\

ES\_AUTOHSCROLL,\

95,65,100,25,hWnd,EditID,hInstance,NULL

mov hwndEditStep,eax

invoke SetFocus, hwndEditMinX

invoke CreateWindowEx,NULL, ADDR ButtonClassName,ADDR ButtonText,\

WS\_CHILD or WS\_VISIBLE or BS\_DEFPUSHBUTTON,\

95,95,100,25,hWnd,ButtonID,hInstance,NULL

mov hwndButton,eax

invoke CreateWindowEx,NULL, ADDR ButtonClassName,ADDR LineName,\

WS\_CHILD or WS\_VISIBLE or BS\_AUTORADIOBUTTON,\

95,125,100,25,hWnd,LINE\_RB,hInstance,NULL

invoke SendMessage,eax,BM\_SETCHECK,BST\_CHECKED,0

invoke CreateWindowEx,NULL, ADDR ButtonClassName,ADDR PointName,\

WS\_CHILD or WS\_VISIBLE or BS\_AUTORADIOBUTTON,\

95,155,100,25,hWnd,POINT\_RB,hInstance,NULL

.ELSEIF uMsg==WM\_COMMAND

mov eax,wParam

.IF ax==LINE\_RB ;если в построении выбрана линия

mov flg,0

.ELSEIF ax==POINT\_RB ;если в построении выбраны точки

mov flg,1

.ELSE

.IF ax==ButtonID

shr eax,16

.IF ax==BN\_CLICKED

invoke GetWindowText,hwndEditMinX,ADDR buffer,25

invoke StrToInt, ADDR buffer

mov minX,eax

invoke GetWindowText,hwndEditMaxX, ADDR buffer,25

invoke StrToInt, ADDR buffer

mov maxX,eax

invoke GetWindowText,hwndEditStep,ADDR buffer,25

invoke StrToFloat, ADDR buffer, ADDR step

;Создаем доп. окно

invoke CreateWindowEx,WS\_EX\_CLIENTEDGE,ADDR ClassName1,ADDR AppNam1e,\

WS\_OVERLAPPEDWINDOW,CW\_USEDEFAULT,\

CW\_USEDEFAULT,800,600,0,NULL,\

hInstance,NULL

mov hwnd1,eax

invoke ShowWindow, hwnd1, SW\_SHOWNORMAL

invoke UpdateWindow, hwnd1

.ENDIF

.ENDIF

.ENDIF

.ELSE

invoke DefWindowProc,hWnd,uMsg,wParam,lParam

ret

.ENDIF

xor eax,eax

ret

WndProc endp

WndProc1 proc hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM

.IF uMsg==WM\_DESTROY

invoke DestroyWindow,hWnd

ret

.ELSEIF uMsg==WM\_PAINT

invoke GetClientRect, hWnd, offset rc

invoke BeginPaint, hWnd, ADDR ps

mov hdc,eax

invoke CreatePen,PS\_SOLID, 2, 00000000h

mov brushBack, eax

invoke SelectObject,hdc,brushBack

;Считаем масштаб, сдвиг

fild rc.bottom

mov [tmp], 2

fidiv tmp

fistp OffsetY

fild maxX

fisub minX

fidivr rc.right

fstp ScaleX

fild rc.bottom

fild maxY

fisub minY

fdiv

fstp ScaleY

fild maxX

fmul ScaleX

fisubr rc.right

fistp OffsetX

;Рисуем координатные оси

invoke MoveToEx,hdc,OffsetX,0,0

invoke LineTo,hdc,OffsetX,rc.bottom

invoke MoveToEx,hdc,0,OffsetY,0

invoke LineTo,hdc,rc.right,OffsetY

invoke CreatePen,PS\_SOLID, 3, 002020FFh

mov brushBack, eax

invoke SelectObject,hdc,brushBack

;Кол-во повторений, точек

fild maxX

fisub minX

fdiv step

fabs

fistp repp

mov ecx, repp

fild minX

fstp x

invoke count

push ecx

invoke MoveToEx, hdc, intX, intY, 0

pop ecx

cnt:

invoke count

push ecx

.IF flg==1

invoke MoveToEx, hdc, intX, intY, 0

.ENDIF

invoke LineTo, hdc, intX, intY

pop ecx

fld x

fadd step

fstp x

ffree st(3)

ffree st(4)

ffree st(5)

loop cnt

invoke EndPaint, hWnd, ADDR ps

.ELSE

invoke DefWindowProc,hWnd,uMsg,wParam,lParam

ret

.ENDIF

xor eax,eax

ret

WndProc1 endp

jmp @enddd

count proc

;числитель

;x(x-18)=x^2-18x=(1)

mov[tmp], 18

fld x

fisub tmp

fmul x

;sqrt(1)=(2)

fsqrt

;x+(2)=(3)

fadd x

;ln(3)=(4)

fldln2

fxch

fyl2x

;знаменатель

;x^2-12=(5)

mov[tmp], 12

fld x

fmul x

fisub tmp

;e^(5)=(6)

fldl2e

fmul

fld st

frndint

fsub st(1), st

fxch st(1)

f2xm1

fld1

fadd

fscale

;(6)+x=(7)

fadd x

;log(7)=(8)

fldlg2

fxch

fyl2x

;(4)/(8)

fdiv st(2),st(0)

fld st(2)

;Считаем X и Y, как координату пикселя

fmul ScaleY

fild OffsetY

fsubr

fistp intY

fld x

fmul ScaleX

fiadd OffsetX

fistp intX

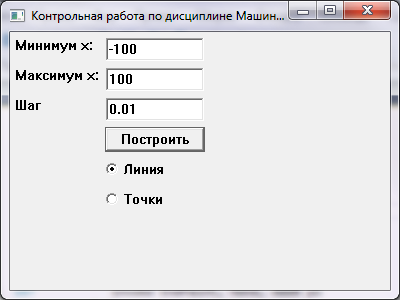
ret

count endp

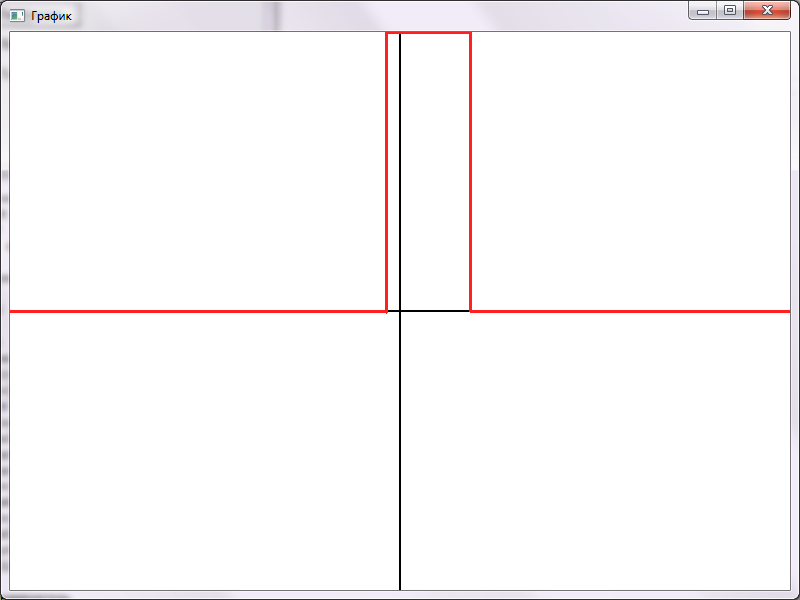
@enddd:

end start

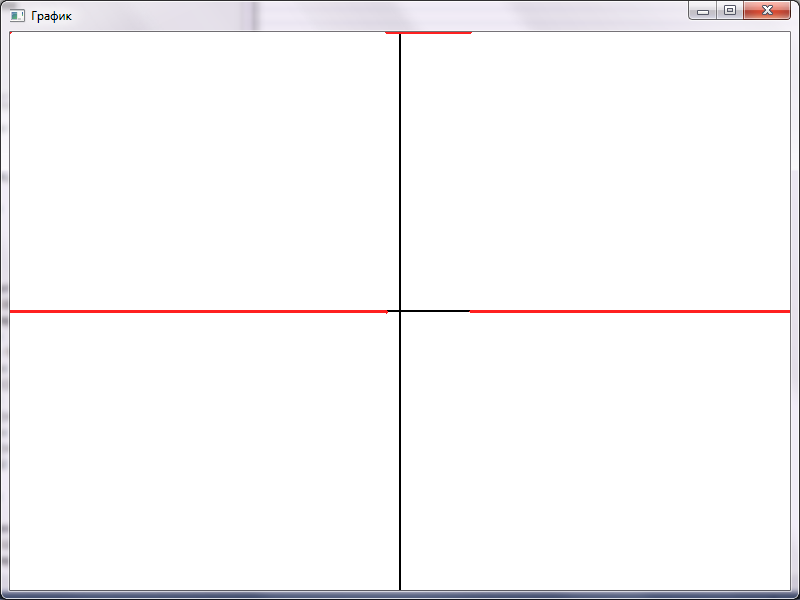
**Пример работы программы:**



**С выбранным параметром Линия**



**С выбранным параметром Точки**



**Задание №2**

.686

.model flat, stdcall

option casemap:none

WinMain proto :DWORD,:DWORD,:DWORD,:DWORD

count proto :DWORD,:DWORD

Paint\_Proc proto :DWORD, :DWORD

include windows.inc

include user32.inc

include kernel32.inc

include gdi32.inc

includelib user32.lib

includelib kernel32.lib

includelib gdi32.lib

.data

polygon POINT 4 dup(<>)

brushBack dd ?

ClassName db "SimpleWinClass", 0

AppName db "Контрольная работа по дисциплине Машинно-зависимые языки, Редняков Вячеслав, Вариант №12", 0

cirX dd 200

cirY dd 50

cirX2 dd 450

cirY2 dd 300

hInstance HINSTANCE ?

RedBrush dd ?

YellowBrush dd ?

AquaBrush dd ?

GreenBrush dd ?

DefPen dd ?

mouseX dd ?

mouseY dd ?

.code

start:

invoke GetModuleHandle, NULL

mov hInstance, eax

invoke WinMain, hInstance, NULL, NULL, NULL

invoke ExitProcess, eax

WinMain proc hInst:HINSTANCE, hPrevInst:HINSTANCE, CmdLine:LPSTR, CmdShow:DWORD

LOCAL wc:WNDCLASSEX

LOCAL msg:MSG

LOCAL hwnd:HWND

mov wc.cbSize, SIZEOF WNDCLASSEX

mov wc.style, CS\_HREDRAW or CS\_VREDRAW

mov wc.lpfnWndProc, OFFSET WndProc

mov wc.cbClsExtra, NULL

mov wc.cbWndExtra, NULL

push hInst

pop wc.hInstance

mov wc.hbrBackground, 0

mov wc.lpszMenuName, 0

mov wc.lpszClassName, OFFSET ClassName

invoke LoadIcon, NULL, IDI\_APPLICATION

mov wc.hIcon, eax

invoke LoadCursor, NULL, IDC\_ARROW

mov wc.hCursor, eax

invoke RegisterClassEx, addr wc

;Создаем кисти и перо

invoke CreateSolidBrush, 00000ffh

mov RedBrush, eax

invoke CreateSolidBrush, 0000ffffh

mov YellowBrush, eax

invoke CreateSolidBrush, 00ffff00h

mov AquaBrush, eax

invoke CreateSolidBrush, 0000ff00h

mov GreenBrush, eax

invoke CreatePen, PS\_SOLID, 3, 00000000h

mov DefPen, eax

INVOKE CreateWindowEx, WS\_EX\_CLIENTEDGE, ADDR ClassName, ADDR AppName,\

WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT,\

CW\_USEDEFAULT, 800, 643, NULL, NULL,\

hInst, NULL

mov hwnd, eax

INVOKE ShowWindow, hwnd, SW\_SHOWNORMAL

INVOKE UpdateWindow, hwnd

.WHILE TRUE

INVOKE GetMessage, ADDR msg, NULL, 0, 0

.BREAK .IF (!eax)

INVOKE TranslateMessage, ADDR msg

INVOKE DispatchMessage, ADDR msg

.ENDW

mov eax, msg.wParam

ret

WinMain endp

WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM

LOCAL hOld:HDC

LOCAL hMemDC: HDC

LOCAL hBmp: HDC

LOCAL ps:PAINTSTRUCT

LOCAL rc:RECT

LOCAL hdc:HDC

.IF uMsg==WM\_DESTROY

invoke PostQuitMessage, NULL

ret

.ELSEIF uMsg==WM\_CREATE ;Создаем фигуру Треугольник и задаем для нее координаты

mov polygon.x, 200

mov polygon.y, 550

mov polygon[type POINT].x, 450

mov polygon[type POINT].y, 550

mov polygon[2\*type POINT].x, 325

mov polygon[2\*type POINT].y, 300

.ELSEIF uMsg==WM\_PAINT

invoke GetClientRect, hWnd, ADDR rc

invoke BeginPaint, hWnd, ADDR ps

mov hdc, eax

;Контекст устройства в памяти

invoke CreateCompatibleDC, hdc

mov hMemDC, eax

;Bitmap для рисования

invoke CreateCompatibleBitmap, hdc, rc.right, rc.bottom

mov hBmp, eax

;Присоединяем bitmap к DC

invoke SelectObject, hMemDC, hBmp

mov hOld, eax

;Заливка рабочей области окна

invoke FillRect, hMemDC, ADDR rc, 0

;Рисуем фигуры

invoke SelectObject, hMemDC, DefPen

;задаем цвет и рисуем треугольник

invoke SelectObject, hMemDC, RedBrush

invoke Polygon, hMemDC, ADDR polygon, 3

;задаем цвет и рисуем прямоугольник с указанием координат

invoke SelectObject, hMemDC, AquaBrush

invoke Rectangle, hMemDC, 50, 550, 200, 300

;задаем цвет и рисуем прямоугольник с указанием координат

invoke SelectObject, hMemDC, GreenBrush

invoke Rectangle, hMemDC, 450, 550, 600, 300

;задаем цвет и рисуем элипс по координатам лежащим в переменных cirX, cirY, cirX2, cirY2

invoke SelectObject, hMemDC, YellowBrush

invoke Ellipse, hMemDC, cirX, cirY, cirX2, cirY2

;Двойная буферизация

invoke BitBlt, hdc, 0, 0, rc.right, rc.bottom, hMemDC, 0, 0, SRCCOPY

invoke SelectObject, hMemDC, hOld

invoke DeleteObject, hBmp

invoke DeleteDC, hMemDC

invoke EndPaint, hdc, ADDR ps

.ELSEIF uMsg==WM\_KEYDOWN ;если нажата клавиша на клавиатуре

.IF wParam==VK\_LEFT ;если это кнопка лево

.IF cirX!=0 ;если круг не на левом краю

;смещаем круг влево

mov eax, cirX

sub eax, 200

mov cirX, eax

mov eax, cirX2

sub eax, 200

mov cirX2, eax

.ENDIF

invoke InvalidateRect, hWnd, 0, FALSE

.ENDIF

.IF wParam==VK\_RIGHT ;если это кнопка право

.IF cirX2!=650 ;если круг не на правом краю

;смещаем круг вправо

mov eax, cirX

add eax, 200

mov cirX, eax

mov eax, cirX2

add eax, 200

mov cirX2, eax

.ENDIF

invoke InvalidateRect, hWnd, 0, FALSE

.ENDIF

.ELSE

invoke DefWindowProc, hWnd, uMsg, wParam, lParam

ret

.ENDIF

xor eax, eax

ret

WndProc endp

end start

**Пример работы программы:**

