**COAL PROJECT**



**CODE**

**COMBINED CALCULATOR**

**MEMBER**

* **Fuzail Ahmed** 😊 01-134231-025

INCLUDE Irvine32.inc

.data

welcomemsg BYTE " ============ ==== WELCOME TO THE COMBINED CALCULATOR ==== ============= ",0 ; Required Menu promptss

optionmsg1 BYTE " ADDITION (A) ",0

optionmsg2 BYTE " SUBTRACTION (S) ",0

optionmsg3 BYTE " MULTIPLICATION (M) ",0

optionmsg4 BYTE " DIVISION (D) ",0

optionmsg5 BYTE " REMAINDER (R) ",0

optionmsg6 BYTE " PRIME OR NOT CHECKER (O) ",0

optionmsg7 BYTE " FACTORIAL (F) ",0

optionmsg8 BYTE " CONTINIOUS (C) ",0

optionmsg9 BYTE " END PROGRAME (E) ",0

askmsg BYTE " Press the char of operation u want to perform ......... ",0

charInput BYTE ?

val1 sdword ? ; calculation variables

val2 sdword ?

askagainmsg0 BYTE " \* \* \* W R O N G O P T I O N S E L E C T E D \* \* \* ",0

askagainmsg BYTE " Do you want to try again this operation (y) : ",0

askagainmsg2 BYTE " Do you want to go back to main menu (y) : ",0

addmsg1 BYTE " Enter First value : ",0 ; prompts for Addition procedure

addmsg2 BYTE " Enter Second value : ",0

addansmsg BYTE " Added value is equal to : ",0

submsg1 BYTE " Enter to Substract from : ",0 ; prompts for Subtraction procedure

submsg2 BYTE " Enter to Substract : ",0

subansmsg BYTE " Subbed value is equal to : ",0

mulansmsg BYTE " Multiplied value is equal to : ",0 ; prompts for Multiplication procedure

scaleFactor DWORD 1000

divmsg1 BYTE " Enter numerator value : ",0 ; prompts for Division procedure

divmsg2 BYTE " Enter divisor value : ",0

divansmsg BYTE " Divided value is equal to : ",0

divsepmsg BYTE ".",0

quotient sdword ? ; prompts for Remainder procedure

remainder sdword ?

remmsg1 BYTE " Enter dividend : ",0

remmsg2 BYTE " Enter divisor : ",0

remansmsg1 BYTE " Quotient is equal to : ",0

remansmsg2 BYTE " Remainder is equal to : ",0

orpmsg BYTE " Enter the value : ",0 ; prompts for prime or not procedure

orpprimsg BYTE " This value is : Prime ",0

orpnotmsg BYTE " This value is : Not Prime ",0

factmsg BYTE " Enter the value : ",0

factansmsg BYTE " Factorial of entered value is : ",0

con sdword ? ; prompts for Continous procedure

opin sdword ?

conmsg1 BYTE " Enter a Value : ",0

conmsg2 BYTE " Enter the operator : " ,0

conans BYTE " Answer is equal to : ",0

wrongop BYTE " \* \* \* I N V A L I D O P E R A T O R \* \* \* ",0

askagaincon BYTE " Do you want to perform again on top (y) : ",0

thanksmsg BYTE " ============= ==== THANKS FOR USING ==== ============= ",0

.code

main PROC

start:

call clrscr

mov edx , offset welcomemsg ; displaying the menu prompts

call writestring

call crlf

mov edx , offset optionmsg1

call writestring

call crlf

mov edx , offset optionmsg2

call writestring

call crlf

mov edx , offset optionmsg3

call writestring

call crlf

mov edx , offset optionmsg4

call writestring

call crlf

mov edx , offset optionmsg5

call writestring

call crlf

mov edx , offset optionmsg6

call writestring

call crlf

mov edx , offset optionmsg7

call writestring

call crlf

mov edx , offset optionmsg8

call writestring

call crlf

mov edx , offset optionmsg9

call writestring

call crlf

mov edx , offset askmsg

call writestring

mov edx, OFFSET charInput ; taking the char for procedure selection

mov ecx, 2 ; ecx = 2 is used to avoid errors and it will only take first alpha

call ReadString

movzx eax, byte ptr [charInput]

cmp eax , 'a' ; comparing the char input to run the desird procedure

je addt

cmp eax , 'A'

je addt

cmp eax , 's'

je subt

cmp eax , 'S'

je subt

cmp eax , 'm'

je mult

cmp eax , 'M'

je mult

cmp eax , 'd'

je divt

cmp eax , 'D'

je divt

cmp eax , 'r'

je remt

cmp eax , 'R'

je remt

cmp eax , 'o'

je orpt

cmp eax , 'O'

je orpt

cmp eax , 'f'

je fact

cmp eax , 'F'

je fact

cmp eax , 'c'

je cont

cmp eax , 'C'

je cont

cmp eax , 'e'

je end\_end

cmp eax , 'E'

je end\_end

call crlf

call crlf

mov edx , offset askagainmsg0

call writestring

call crlf

call crlf

call waitmsg

call crlf

jmp start ; else go back to start

addt:

call addition

jmp endprogram ; calling the desired procedure

subt:

call subtraction

jmp endprogram

mult:

call multiplication

jmp endprogram

divt:

call division

jmp endprogram

remt:

call remaind

jmp endprogram

orpt:

call primeornot

jmp endprogram

fact:

call factorial

jmp endprogram

cont:

call continuous

jmp endprogram

endprogram: ; tasks to perform when exiting the programe

call clrscr

call crlf

mov edx , offset askagainmsg2

call writestring

mov edx, OFFSET charInput

mov ecx, 2

call ReadString

movzx eax, byte ptr [charInput]

cmp eax , 'y'

je start

cmp eax , 'Y'

je start

end\_end:

call clrscr

call crlf

call crlf

xor edx,edx

mov edx , offset thanksmsg

call writestring

call crlf

call crlf

call waitmsg

exit

main ENDP

addition proc

adds:

call clrscr

call crlf

mov edx , offset addmsg1 ; asking user to input first value

call writestring

call readint

mov val1 , eax

call crlf

mov edx , offset addmsg2 ; asking user to input second value

call writestring

call readint

mov val2 , eax

call crlf

add eax ,val1 ; adding both and displaying

mov edx , offset addansmsg

call writestring

call writeint

call crlf

call crlf

mov edx , offset askagainmsg

call writestring

mov edx, OFFSET charInput

mov ecx, 2

call ReadString

movzx eax, byte ptr [charInput]

cmp eax , 'y'

je adds

cmp eax , 'Y'

je adds

ret

addition endp

subtraction proc

adds:

call clrscr

call crlf

mov edx , offset submsg1 ; asking user to input first value

call writestring

call readint

mov val1 , eax

call crlf

mov edx , offset submsg2 ; asking user to input second value

call writestring

call readint

mov val2 , eax

call crlf

mov eax , val1

sub eax ,val2 ; subtracting both and displaying

mov edx , offset subansmsg

call writestring

call writeint

call crlf

call crlf

mov edx , offset askagainmsg

call writestring

mov edx, OFFSET charInput

mov ecx, 2

call ReadString

movzx eax, byte ptr [charInput]

cmp eax , 'y'

je adds

cmp eax , 'Y'

je adds

ret

subtraction endp

multiplication proc

adds:

call clrscr

call crlf

mov edx , offset addmsg1 ; asking user to input first value

call writestring

call readint

mov val1 , eax

call crlf

mov edx , offset addmsg2 ; asking user to input second value

call writestring

call readint

mov val2 , eax

call crlf

mov eax , val1

mul val2 ; multiplying both and displaying

mov edx , offset mulansmsg

call writestring

call writeint

call crlf

call crlf

mov edx , offset askagainmsg

call writestring

mov edx, OFFSET charInput

mov ecx, 2

call ReadString

movzx eax, byte ptr [charInput]

cmp eax , 'y'

je adds

cmp eax , 'Y'

je adds

ret

multiplication endp

division proc

adds:

call clrscr

call crlf

mov edx , offset divmsg1 ; asking user to input first value

call writestring

call readint

mov val1 , eax

call crlf

mov edx , offset divmsg2 ; asking user to input second value

call writestring

call readint

mov val2 , eax

call crlf

mov eax , val1

mov ebx , val2

xor edx, edx ; reseting edx

div ebx ; dividing both and we get quotient in eax and remainder in edx

mov quotient , eax

mov eax, edx

mul scaleFactor ; multiplying the remainder by scale factor that is 1000 in this case

div ebx ; then dividing to get the decimal value of the main dividion

mov remainder , eax

mov eax , quotient

mov edx , offset divansmsg ; diplaying integer

call writestring

call Writedec

mov edx , offset divsepmsg ; displaying seperation '.'

call writestring

mov eax , remainder ; displaying decimal value

call writedec

call crlf

call crlf

mov edx , offset askagainmsg

call writestring

mov edx, OFFSET charInput

mov ecx, 2

call ReadString

movzx eax, byte ptr [charInput]

cmp eax , 'y'

je adds

cmp eax , 'Y'

je adds

ret

division endp

remaind proc

adds:

call clrscr

call crlf

mov edx , offset remmsg1 ; asking user to input first value

call writestring

call readint

mov val1 , eax

call crlf

mov edx , offset remmsg2 ; asking user to input second value

call writestring

call readint

mov val2 , eax

call crlf

mov eax , val1

mov ebx , val2

xor edx , edx ; reseting edx

div ebx ; dividing both and we get quotient in eax and remainder in edx

mov quotient , eax

mov remainder , edx

mov eax , quotient

mov edx , offset remansmsg1 ; displaying the quotient

call writestring

call Writedec

call crlf

call crlf

mov eax , remainder

mov edx , offset remansmsg2 ; displaying remainder

call writestring

call writedec

call crlf

call crlf

mov edx , offset askagainmsg

call writestring

mov edx, OFFSET charInput

mov ecx, 2

call ReadString

movzx eax, byte ptr [charInput]

cmp eax , 'y'

je adds

cmp eax , 'Y'

je adds

ret

remaind endp

primeornot PROC

adds:

call clrscr

call crlf

mov edx , offset orpmsg ; asking user to input the value

call writestring

call readint

mov val1 , eax

call crlf

cmp val1, 1 ; Check if the number is 1

je not\_prime

cmp val1, 2 ; Check if the number is 2

je prime

mov ebx , val1

dec ebx

mov ecx, 2 ; Start checking for divisibility from 2

check\_divisibility:

mov eax, val1 ; Move the value to EAX

xor edx, edx ; Clear EDX for division

div ecx ; Divide EAX by ECX

mov eax , edx

call writedec

call crlf

cmp edx, 0 ; Check if the remainder is zero

je not\_prime ; If remainder is zero, the number is not prime

cmp ebx , ecx

je prime

inc ecx ; Increment divisor

jmp check\_divisibility

prime:

mov edx, OFFSET orpprimsg

call WriteString

jmp end\_prog

not\_prime:

mov edx, OFFSET orpnotmsg

call WriteString

end\_prog:

call crlf

call crlf

mov edx , offset askagainmsg

call writestring

mov edx, OFFSET charInput

mov ecx, 2

call ReadString

movzx eax, byte ptr [charInput]

cmp eax , 'y'

je adds

cmp eax , 'Y'

je adds

ret

primeornot endp

factorial proc

adds:

call clrscr

call crlf

mov edx , offset factmsg ; asking user to input the value

call writestring

call readint

mov val1 , eax

cmp eax , 0 ; if 0 exit with 0

je end\_prog

mov ecx , val1 ; else run a loop input times and add each value

mov val1 , 0

l1:

add val1 , ecx

loop l1

end\_prog:

mov eax , val1

mov edx , offset factansmsg

call writestring

call writedec

call crlf

call crlf

call crlf

mov edx , offset askagainmsg

call writestring

mov edx, OFFSET charInput

mov ecx, 2

call ReadString

movzx eax, byte ptr [charInput]

cmp eax , 'y'

je adds

cmp eax , 'Y'

je adds

ret

factorial endp

continuous proc

adds:

call clrscr

call crlf

mov edx , offset conmsg1 ; asking user to input first value

call writestring

call readint

mov con , eax

next\_time:

call crlf

mov edx , offset conmsg2 ; asking to input the operator to perform

call writestring

mov edx, OFFSET opin

mov ecx, 2

call ReadString

movzx eax, byte ptr [opin]

cmp eax , '!' ; if factorial '!' then run that without getting second value

je facti

call crlf

mov edx , offset conmsg1 ; asking the second value

call writestring

call readint

mov val2 , eax

call crlf

movzx eax, byte ptr [opin] ; comparing the operator to perform that specific operation

cmp eax , '+'

je addi

cmp eax , '-'

je subi

cmp eax , '\*'

je multi

cmp eax , '/'

je divi

mov edx , offset wrongop

call writestring

call crlf

call crlf

call waitmsg

jmp adds

addi: ; procedure for add

mov eax , val2

add con , eax

mov eax , con

mov edx , offset addansmsg

call writestring

call writeint

call crlf

call crlf

jmp end\_prog

subi: ; procedure for sub

mov eax , val2

sub con , eax

mov eax , con

mov edx , offset subansmsg

call writestring

call writeint

call crlf

call crlf

jmp end\_prog

multi: ; procedure for multi

mov eax , con

mul val2

mov con , eax

mov edx , offset mulansmsg

call writestring

call writeint

call crlf

call crlf

jmp end\_prog

divi: ; procedure for div

mov eax , con

mov ebx , val2

xor edx, edx

div ebx

mov quotient , eax

mov eax, edx

mul scaleFactor

div ebx

mov remainder , eax

mov eax , quotient

mov edx , offset divansmsg

call writestring

call Writedec

mov con , eax

mov edx , offset divsepmsg

call writestring

mov eax , remainder

call writedec

call crlf

call crlf

jmp end\_prog

facti: ; procedure for factorial

mov eax , con

cmp eax , 0

je end\_fact

mov ecx , con

mov con , 0

l1:

add con , ecx

loop l1

end\_fact:

call crlf

mov eax , con

mov edx , offset factansmsg

call writestring

call writedec

call crlf

call crlf

jmp end\_prog

end\_prog:

mov edx , offset askagaincon

call writestring

mov edx, OFFSET charInput

mov ecx, 2

call ReadString

movzx eax, byte ptr [charInput]

cmp eax , 'y'

je next\_time

cmp eax , 'Y'

je next\_time

ret

continuous endp

END main