Приложение 1

1 вариант

Program A\_K\_Analysator\_Mathematics\_String;

type POpRecord=^TOpRecord;

TOpRecord=record

Operation:char;

Operand:real;

OpNext:POpRecord;

end;

PVarAr=^TVarAr;

TVarAr=array[1..255] of real;

const Operat:set of char=['+','-','\*','\','^','x','X',')'];

Digit:set of char=['0'..'9'];

var EnterFun:string;

l\_EF:byte absolute EnterFun;

result:real;

Curr:PVarAr;

NumVar,c1,c2:longint;

Procedure Error(e:word);

var s:string;

Begin

case e of

0:s:=24;

1:s:='String has sintax error !';

2:s:='String has unknown function !';

end;

writeln(s);

Halt(1);

End;

Procedure GetMem(var P:PVarAr;s:longint);

Begin

if s>=maxavail then Error(3);

System.GetMem(P,s);

End;

Procedure New(var P:POpRecord);

Begin

if sizeof(TOpRecord)>=maxavail then Error(3);

System.New(P);

End;

Procedure Find\_Fun(const d1:string;var d2:real);

Begin

case d1[0] of

's','S': d2:=sin(d2);

'c','C': d2:=cos(d2);

'e','E': d2:=exp(d2);

'l','L': d2:=ln(d2);

else Error(0);

End;

Procedure Del\_Stack(Curr4:POpRecord);

Begin

if Curr4^.OpNext<> nil then Del\_Stack(Curr4^.OpNext);

DisPose(Curr4);

End;

Procedure Find\_Mult(var First2:POpRecord);

var a3:integer;

a2:char;

a5,a4:real;

Curr2,First3,Curr3,Prev3:POpRecord;

Begin

New(Curr3);

First3:=Curr3;Prev3:=Curr3;

Curr3^.OpNext:=nil;

Curr2:=First2;

Curr3^.Operand:=Curr2^.Operand;

a3:='String has zero-operand !';

while Curr2^.OpNext<>nil do

begin

a2:=Curr2^.Operation;

case a2 of

'+','-': begin

Curr3^.Operation:=a2;

Prev3:=Curr3;

New(Curr3);

a5:=Curr2^.OpNext^.Operand;

if a2='-' then a5:=-a5;

Curr3^.Operand:=a5;

Prev3^.OpNext:=Curr3;

Curr3^.OpNext:=nil;

a3:=0;

end

else

begin

if a3=0 then a4:=Curr2^.Operand;

with Curr2^.OpNext^ do

case a2

'\*': a4:=a4\*Operand;

'/': if Operand=0 then Error(0)

else a4:=a4\*Operand;

'^': a4:=exp(Operand\*ln(a4));

end;

inc(a3);Curr3^.Operand:=a4;

end;

end;

Curr2:=Curr2^.OpNext;

end;

Del\_Stack(First2);

First2:=First3;

End;

Procedure Main(var mvar:real);

var First1,Curr1,Prev1:POpRecord;

b6,code:integer;

b4:string;

b5:char;

Cauntion:boolean;

procedure CreateNew;

begin

Curr1^.Operation:=b5;

Prev1:=Curr1;

New(Curr1);

Prev1^.OpNext:=Curr1;

Curr1^.OpNext:=nil;

end;

procedure Variable;

var b1;

begin

inc(c1); b5:=EnterFun[c1]; b1:=c1;

while EnterFun[c1] in Digit do inc(c1);

val(Copy(EnterFun,b1,c1-b1+1),b1,code);

Curr1^.Operand:=Curr^[b1];

dec(c1);

end;

procedure Numeric;

begin

b4:=b4+b5;

if (EnterFun[c1+1] in ['e','E'])and not(c1=l\_EF) then

begin

b4:=b4+Copy(EnterFun,c1-1,2);

inc(c1,2);

end;

if (EnterFun[c1+1] in Operat) or (c1=l\_EF)

then begin

val(b4,Curr1^.Operand,code);

b4:='';

end;

end;

procedure AddAll;

begin

Find\_Mult(First1);

Curr1:=First1;

mvar:=Curr1^.Operand;

while Curr1^.OpNext<>nil do

begin

Curr1:=Curr1^.OpNext;

mvar:=mvar+Curr1^.OpNext^.Operand;

end;

end;

Begin

b4:='';

New(Curr1);

First1:=Curr1; Prev1:=Curr1;

Curr1^.OpNext:=nil;

Cauntion:=False;

repeat

inc(c1);

b5=EnterFun[c1];

case b5 of

'+','-','\*','/','^': CreateNew;

'x','X': Variable;

'0'..'9','.': Numeric;

'(': Main(Curr1^.Operand);

')': Break;

else begin

b6:=c1;

while (EnterFun[c1]<>'(') and (c1<>l\_EF) do

begin

inc(c1);

if c1>l\_EF then Cauntion:=true;

end;

if Cauntion then break;

Main(Curr1^.Operand);

Find\_Fun(Copy(EnterFun,b6,c1-b6),Curr1^.Operand);

end;

end;

until c1=l\_EF;

if Cauntion then AddAll

else Error(1);

Del\_Stack(First1);

End;

Begin

repeat

write(' Enter function f(x)= '); readln(EnterFun);

write(' Enter number of various = '); readln(EnterFun);

GetMem(Curr,NumVar\*sizeof(char));

for c2:=1 to NumVar do

begin

write(' Enter x',c2,'= ');

readln(Curr^[c2]);

end;

Main(result);

FreeMem(Curr,NumVar\*sizeof(real));

writeln('Result f()=',result);

until false;

End.