For hw proj 7

If F(x) = x^2 + 3x + 5

F(a)=f(2)=(2)^2 + 3(2) + 5

=1.x^2 + 3.x^1 + 5.x^0

T and F-> [coef|pow|next]->[3|1|next]->[5|0|/]

1.x ^2 + 3.x^1 + 5x^0

struct node

{

int coef;

int pow;

node \*next;

}//end struct node

node \*t;

t = new(node); //we have t->[t->coef|t->pow|t->next]

a=3

Value=0

T=f;

While(t!=NULL)

{

//every time we get to node

value+=t->coef\*pow(a,t->pow)

//1 \*a^2

//go through this loop over until value of. . .

}

//to display

while(t!=NULL)

{

//make sure it’s positive or negative

cout<<t->coef<<”.x^”<<t->pow;

}

//note don’t use pop method bec it’ll remove ur node

int popq()

{

//example t->[5|2|-]->[3|1|-]->[-5|0|/]

x

}//end popq

//Wednesday will have long quiz to do insertion deletion of nodes etc.

class POLY

{

private:

struct node

{

int coef, pwr;

node \*next;

}//end struct node

public:

node \*front, \*rear;

POLY(){front=NULL; rear=NULL;}

void insert(int c, int pw)

{

node \*t = new(noed); //t->[||]

t->coef = c;

t->pwr = pw;

t->next=NULL; //t->[c|pw|/]

if (front==NULL)

{front = t; rear=t;}

else

{rear->next=t; rear=t;}

//--------construct your linked list

//ex 5x^3+4x^2-10x+2 asks you to enter coef and pwr of each term

/\*you put in 5 and 3 u now have

front and rear ->[5|3|/]

next one is 4 and 2, t->[4|2|/], rear next points to t and assign rear to where t is pointing so u have front->[5|3|-]->rear->[4|2|/] etc. . . better to trace \*/

}//end insert

int value(int a)

{

node \*t = front; //pointer t pointing to where front is pointing

int val=0;

while(t!=NULL) //t goes through the nodes

{

val+=t->coef\*pow(a, t->pwr)

t=t->next;

}//endwhile

return val; //returns your polynomial

}//end value

//we also need something to display the linked list

void display()

{

node \*t = front; //pointer t pointing to where the front is pointing to

while(t!=NULL)

{

if(t->coef>0)

{cout<<”+”<<t->coef;}

else {cout<<”-“<<t->coef;}

cout<<”x^”<<t->pwr;

t=t->next;

}

}//end display

}//end POLY

int main()

{

POLY p; //we now have front and rear pointing to NULL

int c, pw; //for coef and powr

cout<<”Enter coef and pwr. .0,0 to stop: “;

cin>>c>>pw;

while(c!=0 && pw!=0)

{

p.insert(c,pw);

cout<<”Enter coef . . .etc. “;

cin>>c>>pw;

}

//display polynomial

p.display();

int a;

cout<<”Enter a value to evaluate the polynomial: ”;

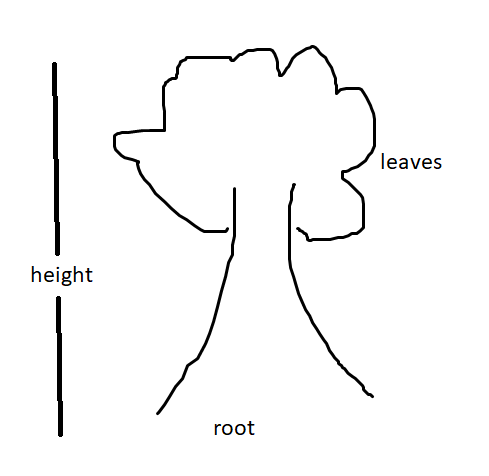
cin>>a;

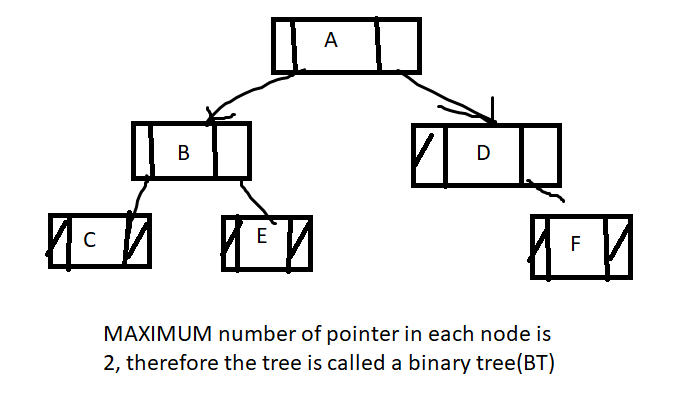
cout<<”f{“<<a<<”) is ”<<p.value(a)<<endl;

}//end mian

NEW TOPIC

Trees





each [ ] is called a node

declaration

struct node

{

int info;

node\* left;

node \*right;

}//end struct node

node \*r;

r=new node; //r->[r->left|r->info|r->right]

consider [|B|]

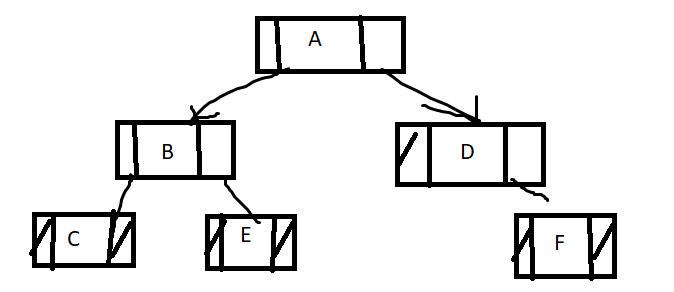
/\

[|C|] [|E|] //-B is called the PARENT of C and E

//C is the LEFT CHILD, E is the RIGHT CHILD of B

//C and E are SIBLINGS

//UNDERLINED are technical terms



-the nodes without children are called LEAF : C, E, F

-The height of a binary tree BT is the longest distance from root to a leaf.. we count the number of arrows or pointers. the height of the tree is therefore 2

ALGORITHM to visit all nodes in BT

There ar 3 METHODS to visit all the nodes

1. pre-order traversal
2. in-order traversal
3. post-order traversal

Given the following BT.

apply all 3 traversal algorithms to visit all nodes

root [A]

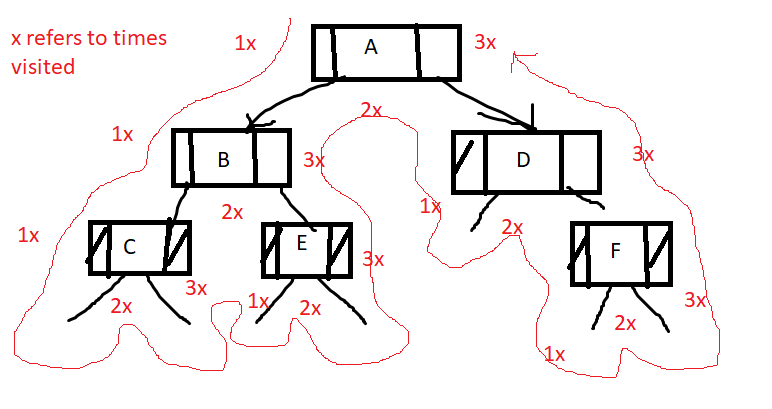
/\

[B] [E]

/\ /\

[C] [D] [F] //we ensure each node has 2 children

/\ /\ /\

//we visit them all of em 

1. pre-oder (1) : A B C D E F //saw first time first to last
2. in-order (2) : C B D A E F //saw 2nd time first to last
3. postoder(3) : C D B F E A //saw 3rd time first to last