Project: BallotBoxes

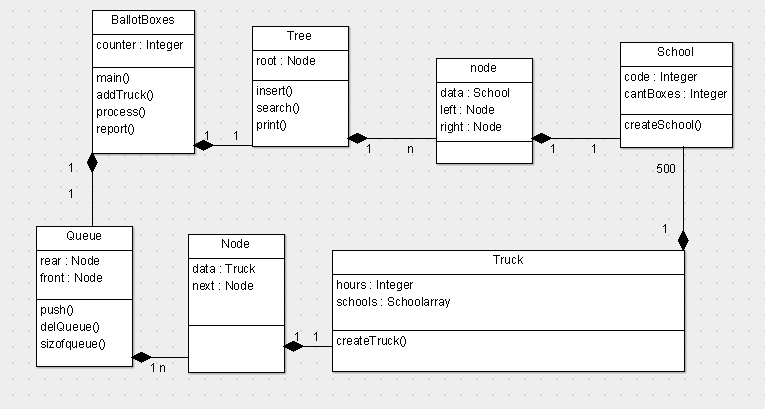
Author: Tomás Lucas Vilaboa

Hours: 6

Code Lines: 228

Files: Main.c, tree.h,queue.h,school.h,truck.h

***UML***



***Main.c***

#include <stdio.h>

#include <stdlib.h>

#include "School.h"

#include "tree.h"

#include "queue.h"

int counter=0;

int main()

{

front=NULL;

node \*root;

root=NULL;

int menu=0;

addTruck(); //lets suposse there is already 1 truck waiting

do{

counter+=1;

addTruck();

if(counter %3==0){

process(&root);

}

printf("Enter 1 to continue the simulation and enter 2 to finish and print the report.\n");

scanf("%d",&menu);

} while (menu==1);

report(root);

return 0;

}

//add truck to queue with 60% prob

void addTruck(){

if(rand()%10 <=6){

createTruck(counter);

}

}

//process the first truck stored in queue and adds the schools from that truck to a tree in order to print them ordenated

void process(node \*\*root){

struct truck truck;

struct Node \*node=delQueue();

truck=node->Data;

printf("Truck %d waited %d hours\n",counter/3,counter - truck.hours);

int i=0;

if(sizeofqueue()>0){

for(i;i<500;i++){

struct school school=truck.schools[i];

struct node \*tmp;

tmp = search(root, school);

if (tmp)

{

createSchool(&((\*root)->data),(\*root)->data.code,(\*root)->data.cantboxes+1);

}

else

{

insert(root,school);

}

}

}

}

//print the tree inorden= ordenated report

void report(node \*root){

print\_inorder(root);

}

***Tree.h***

struct bin\_tree

{

struct school data;

struct bin\_tree \* right, \* left;

};

typedef struct bin\_tree node;

//insert

void insert(node \*\* tree, struct school val)

{

node \*temp = NULL;

if(!(\*tree))

{

temp = (node \*)malloc(sizeof(node));

temp->left = temp->right = NULL;

temp->data = val;

\*tree = temp;

return;

}

if(val.code < (\*tree)->data.code)

{

insert(&(\*tree)->left, val);

}

else if(val.code > (\*tree)->data.code)

{

insert(&(\*tree)->right, val);

}

}

void print\_preorder(node \* tree)

{

if (tree)

{

printschool(tree);

print\_preorder(tree->left);

print\_preorder(tree->right);

}

}

void print\_inorder(node \* tree)

{

if (tree)

{

print\_inorder(tree->left);

printschool(tree);

print\_inorder(tree->right);

}

}

void print\_postorder(node \* tree)

{

if (tree)

{

print\_postorder(tree->left);

print\_postorder(tree->right);

printschool(tree);

}

}

void printschool(node \* tree)

{

printf("\nCode:%d\n",tree->data.code);

printf("Ballot Boxes: %d\n",tree->data.cantboxes);

}

//delete whole tree

void deltree(node \* tree)

{

if (tree)

{

deltree(tree->left);

deltree(tree->right);

free(tree);

}

}

//search a school and returns it

node\* search(node \*\* tree, struct school val)

{

if(!(\*tree))

{

return NULL;

}

if(val.code < (\*tree)->data.code)

{

search(&((\*tree)->left), val);

}

else if(val.code > (\*tree)->data.code)

{

search(&((\*tree)->right), val);

}

else if(val.code == (\*tree)->data.code)

{

return \*tree;

}

}

***Queue.h***

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include "truck.h"

struct Node

{

struct truck Data;

struct Node\* next;

}\*rear, \*front;

int siz=0;

//equals to pop, delete first value from queue and returns it

struct Node\* delQueue()

{

struct Node \*temp, \*var=rear;

if(var!=NULL)

{

rear = rear->next;

siz-=1;

return var;

}

else

printf("\nQueue Empty");

}

void push(struct truck value)

{

struct Node \*temp;

temp=(struct Node \*)malloc(sizeof(struct Node));

temp->Data=value;

if (front == NULL)

{

front=temp;

front->next=NULL;

rear=front;

}

else

{

front->next=temp;

front=temp;

front->next=NULL;

}

siz+=1;

}

void display()

{

struct Node \*var=rear;

if(var!=NULL)

{

printf("\nElements are as: ");

while(var!=NULL)

{

printf("un truck");

var=var->next;

}

printf("\n");

}

else

printf("\nQueue is Empty");

}

int sizeofqueue(){

return siz;

}

***School.h***

struct school

{

int code;

int cantboxes;

}s;

void createSchool(struct school \*school,int code,int cant)

{

school->cantboxes=cant;

school->code=code;

}

***Truck.h***

#include <time.h>

struct truck{

struct school schools[500];

int hours;

};

//create a truck with 500 schools,each with random code between 1 and 9999 and initialize hours value

void createTruck(int hour){

struct truck truck;

int i=0;

for(i;i<500;i++){

int random\_number = rand() % 9999;

struct school school;

createSchool(&school,random\_number,1);

truck.schools[i]=school;

}

truck.hours=hour;

push(truck);

}