CS2263 Assignment 6

Stephen Cole

3553803

QUEUE

Queue.c

#include <stdio.h>

#include <stdlib.h>

#include "Point2D.h"

#include "Queue.h"

PtQ\* mallocPtQ()

{

PtQ\* queue = (PtQ\*)malloc(sizeof(PtQ));

if(queue == (pPtQ)NULL )

return queue;

queue->head = (pPtLink)NULL;

queue->tail = (pPtLink)NULL;

queue->length = 0;

return queue;

}

void enqueue(PtQ\* queue, Point2D\* pt)

{

pPtLink newLink = createPointLink(pt);

//Empty

if(queue->head == (pPtLink)NULL)

{

queue->head = newLink;

queue->tail = newLink;

}

else if(queue->head == queue->tail)

{

queue->tail = newLink;

queue->head->next = newLink;

}

else

{

queue->tail->next = newLink;

queue->tail = newLink;

}

queue->length = queue->length + 1;

}

void dequeue(PtQ\* queue)

{

if(queue->length == 0)

return;

else if(queue->head->next == (pPtLink)NULL)

{

queue->head = (pPtLink)NULL;

queue->length = queue->length-1;

}

else

{

queue->head = queue->head->next;

queue->length = queue->length-1;

}

}

void listQueue(PtQ\* queue)

{

pPtLink link = (pPtLink)malloc(sizeof(PtLink));

link = queue->head;

for(int i=0; i<queue->length; i++)

{

printf("%lf %lf\n", link->payload->x, link->payload->y);

link = link->next;

}

}

pPtLink createPointLink(Point2D\* pt){

pPtLink ptLink = (pPtLink)malloc(sizeof(PtLink));

ptLink->payload = pt;

ptLink->next = (pPtLink)NULL;

return ptLink;

}

Queue.h

#ifndef QUEUE\_H

#define QUEUE\_H

#include <stdlib.h>

#include "Point2D.h"

typedef struct pt2link {

Point2D\* payload;

struct pt2link\* next;

}PtLink, \*pPtLink;

typedef struct pointqueue {

pPtLink head;

pPtLink tail;

int length;

}PtQ, \*pPtQ;

PtQ\* mallocPtQ();

void enqueue(PtQ\* queue, Point2D\* pt);

void dequeue(PtQ\* queue);

void listQueue(PtQ\* queue);

pPtLink createPointLink(Point2D\* pt);

#endif

playQueue.c

// playQueue.c

#include <stdio.h>

#include <stdlib.h>

#include "Queue.h"

#define ENQUEUE 1

#define DEQUEUE 0

#define LIST 2

#define PEEK 3

int main(int argc, char \* \* argv)

{

int iChoice;

int iNRead;

pPtQ queue = mallocPtQ();

/\* Processing loop \*/

printf("Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): ");

iNRead = scanf("%d", &iChoice);

while(iNRead == 1)

{

switch(iChoice)

{

case ENQUEUE:

printf("Point value to add:"); // Obviously you need to read the x and y values

// Read the element, add it to the queue

double x;

double y;

scanf("%lf %lf", &x, &y);

Point2D\* pt = createPoint2D(x,y);

enqueue(queue, pt);

break;

case DEQUEUE:

// dequeue the Point2D and print it out.

if(queue->head != NULL)

{

printf("Point x=%lf, y=%lf\n", queue->head->payload->x, queue->head->payload->y);

dequeue(queue);

}

else

printf("Queue is empty!\n");

break;

case LIST:

// Print out the stack elements,

listQueue(queue);

break;

case PEEK:

// Print out the next value to be dequeue.

printf("Point x=%lf, y=%lf\n", queue->head->payload->x, queue->head->payload->y);

break;

default:

return 0;

}

printf("Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): ");

iNRead = scanf("%d", &iChoice);

}

return EXIT\_SUCCESS;

}

A close up of text on a black background

Description automatically generated

STACK

Stack.c

#include <stdio.h>

#include <stdlib.h>

#include "Point2D.h"

#include "Stack.h"

PtStack\* mallocPtStack()

{

PtStack\* stack = (PtStack\*)malloc(sizeof(PtStack));

if(stack == (pPtStack)NULL )

return stack;

stack->head = (pPtLink)NULL;

stack->length = 0;

return stack;

}

void push(PtStack\* stack, Point2D\* pt)

{

pPtLink newLink = createPointLink(pt);

newLink->next = stack->head;

stack->head = newLink;

stack->length = stack->length + 1;

}

void pop(PtStack\* stack)

{

pPtLink temp;

if(stack->length == 0)

return;

temp = stack->head->next;

freePoint2D(stack->head->payload);

free(stack->head);

stack->head = temp;

stack->length--;

}

void list(PtStack\* stack)

{

pPtLink link = (PtLink\*)malloc(sizeof(PtLink));

link = stack->head;

for(int i=0; i<stack->length; i++)

{

printf("%lf %lf\n", link->payload->x, link->payload->y);

link = link->next;

}

}

pPtLink createPointLink(Point2D\* pt){

pPtLink ptLink = (pPtLink)malloc(sizeof(PtLink));

ptLink->payload = pt;

ptLink->next = (pPtLink)NULL;

return ptLink;

}

Stack.h

#ifndef STACK\_H

#define STACK\_H

#include <stdlib.h>

#include "Point2D.h"

typedef struct pt2link {

Point2D\* payload;

struct pt2link\* next;

}PtLink, \*pPtLink;

typedef struct pointstack {

pPtLink head;

int length;

}PtStack, \*pPtStack;

PtStack\* mallocPtStack();

void push(PtStack\* stack, Point2D\* pt);

void pop(PtStack\* stack);

void list(PtStack\* stack);

pPtLink createPointLink(Point2D\* pt);

#endif

playStack.c

// playStack.c

#include <stdio.h>

#include <stdlib.h>

#include "Stack.h"

#define PUSH 1

#define POP 0

#define LIST 2

#define PEEK 3

int main(int argc, char\* argv[])

{

int iChoice;

int iNRead;

pPtStack stack = mallocPtStack();

/\* Processing loop \*/

printf("Choice (1=add, 0=remove, 2=list, 3=peek): ");

iNRead = scanf("%d", &iChoice);

while(iNRead == 1)

{

switch(iChoice)

{

case PUSH:

printf("Point value to add:"); // Obviously you need to read the x and y values

// Read the element, add it to the stack

double x;

double y;

scanf("%lf %lf", &x, &y);

Point2D\* pt = createPoint2D(x,y);

push(stack, pt);

break;

case POP:

// Pop the Point2D and print it out.

if(stack->head != NULL)

{

printf("Point x=%lf, y=%lf\n", stack->head->payload->x, stack->head->payload->y);

pop(stack);

}

else

printf("Stack is empty!\n");

break;

case LIST:

// Print out the stack elements

list(stack);

break;

case PEEK:

// Print out the next value to be popped.

printf("Point x=%lf, y=%lf\n", stack->head->payload->x, stack->head->payload->y);

break;

default:

return 0;

}

printf("Choice (1=add, 0=remove, 2=list, 3=peek): ");

iNRead = scanf("%d", &iChoice);

}

return EXIT\_SUCCESS;

}

A screenshot of a cell phone screen with text

Description automatically generated

Makefile

comp: queueTest stackTest

queueTest: playQueue.c Queue.c Point2D.c

gcc -o queueTest -Wall playQueue.c Queue.c Point2D.c

stackTest: playStack.c Stack.c Point2D.c

gcc -o stackTest -Wall playStack.c Stack.c Point2D.c