CS 2263 - FR01A

Assignment 6

By Ngoc Phuong Anh Nguyen - 3712361

November 2021

Question 1:

Create a stack ADT module to manage Point2D data. The module should handle stack creation, push, pop, peek, reporting the stack contents and stack destruction. You should use your current Point2D module wherever possible. Test your module by implementing it with the playStack.c program.

a. Source Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Point2D.h"

Point2D* mallocPoint2D()
{
  Point2D* pPtThis;
  pPtThis = (Point2D*) malloc(sizeof(Point2D));
  return pPtThis;
}

Point2D* createPoint2D(double x, double y)
{
  Point2D* pPtThis;
  pPtThis = mallocPoint2D();
  if(pPtThis == NULL)
  {
    return pPtThis;
  }
  pPtThis->x = x;
  pPtThis->y = y;
  return pPtThis;
}

void freePoint2D(Point2D* pThis)
{
  free(pThis);
}

double getXPoint2D(Point2D* pThis)
{
  return pThis->x;
}

double getYPoint2D(Point2D* pThis)
{
  return pThis->y;
}

void setPoint2D(Point2D* pPt, double x, double y)
{
  pPt->x = x;
  pPt->y = y;
  return;
}

double getDistancePoint2D(Point2D* pThis, Point2D* pThat)
{
  return sqrt(pow((getXPoint2D(pThis) - getXPoint2D(pThat)), 2)
   + pow((getYPoint2D(pThis) - getYPoint2D(pThat)), 2));
}
```

Figure 1: Source Code of Point2D.c

```c
#ifndef POINT2D_H
#define POINT2D_H

typedef struct point2D
{
  double x;
  double y;
}
Point2D;

typedef struct pt2link
{
  Point2D* payload;
  struct pt2link* next;
}
PtLink, *pPtLink;

typedef struct pointList
{
  pPtLink head;
  pPtLink tail;
  pPtLink iterator; /* not implemented */
  int nLinks;
}PointList;

Point2D* mallocPoint2D();
Point2D* createPoint2D(double x, double y);
void freePoint2D(Point2D* pThis);
double getXPoint2D(Point2D* pThis);
double getYPoint2D(Point2D* pThis);
void setPoint2D(Point2D* pPt, double x, double y);
double getDistancePoint2D(Point2D* pThis, Point2D* pThat);

#endif
```

Figure 2: Source Code of Point2D.h

```c
#include <stdio.h>
#include <stdlib.h>
#include "Point2D.h"
#include "stack.h"
// Other supported functions
pPtLink createPointLink(Point2D* point)
{
  pPtLink pPointLink = (pPtLink) malloc(sizeof(PtLink));
  if(pPointLink == (pPtLink)NULL)
  {
    return pPointLink;
  }
  pPointLink->payload = point;
  return pPointLink;
}

void freePointLink(pPtLink pPointLink)
{
    free(pPointLink);
    return;
}

// stack creation
PointList* mallocPointList()
{
  PointList* pointListThis = (PointList*) malloc(sizeof(PointList));
  if(pointListThis == (PointList*)NULL )
  {
    return pointListThis;
  }
  pointListThis->head = (pPtLink)NULL;
  pointListThis->tail = (pPtLink)NULL;
  pointListThis->iterator = (pPtLink)NULL;
  pointListThis->nLinks = 0;

  return pointListThis;
}

// push
int push(Point2D* newPoint, PointList* pointList)
{
  pPtLink pPointLink = createPointLink(newPoint);
  if(pPointLink == (pPtLink)NULL)
  {
    return 1;
  }
  if(pointList->head == pointList->tail && pointList->head == NULL)
  {
    pointList->head = pPointLink;
    pointList->tail = pointList->head;
  }
  else
  {
    pPointLink->next = pointList->head;
    pointList->head = pPointLink;
  }
  pointList->nLinks++;
  return 0; //success
}
```

Figure 3: Source Code of stack.c (1)

```c
// pop
int pop(PointList* pointList)
{
  pPtLink temp;
  if(pointList->head == (pPtLink)NULL)
  {
    return EOF;
  }
  temp = pointList->head;
  pointList->head = pointList->head->next;
  pointList->nLinks--;
  printf("Popped point: (%.1lf, %.1lf)\n", getXPoint2D(temp->payload), getYPoint2D(temp->payload));
  return 0;//success
}

// peek
void peek(PointList* pointList)
{
  if(pointList->head == (pPtLink)NULL)
  {
    printf("The list is empty, unable to peek the point\n");
  }
  else
  {
    Point2D* point = pointList->head->payload;
    printf("Peeked point: (%.1lf, %.1lf)\n", getXPoint2D(point), getYPoint2D(point));
  }
}

// reporting the stack contents
void print(PointList* pointList)
{
  pPtLink current = pointList->head;
  int i = 0;
  if(current == (pPtLink)NULL)
  {
    printf("The list is empty\n");
    return;
  }

  while(current != (pPtLink)NULL)
  {
    printf("Point %d: (%.1lf, %.1lf)\n", i, current->payload->x, getYPoint2D(current->payload));
    printf("- PtLink: %p\n- payload: %p\n- next: %p\n\n", current, current->payload, current->next);
    i++;
    current = current->next;
  }
}

// stack destruction
void freePointList(PointList* pointList)
{
  pPtLink current = pointList->head;
  while(pointList->head != (pPtLink)NULL)
  {
    current = pointList->head->next;
    freePointLink(pointList->head);
    pointList->head = current;
  }
  free(pointList);
  return;
}
```

Figure 4: Source Code of stack.c (2)

```c
#ifndef STACK_H
#define STACK_H
#include "Point2D.h"

pPtLink createPointLink(Point2D* point);
void freePointLink(pPtLink pPointLink);
PointList* mallocPointList();
int push(Point2D* newPoint, PointList* pointList);
int pop(PointList* pointList);
void peek(PointList* pointList);
void print(PointList* pointList);
void freePointList(PointList* pointList);

#endif
```

Figure 5: Source Code of stack.h

```c
// playStack.c
#include <stdio.h>
#include <stdlib.h>
#include "Point2D.h"
#include "stack.h"
#define PUSH 1
#define POP 0
#define LIST 2
#define PEEK 3
int main(int argc, char* argv[])
{
  int iChoice, iNRead, failure;
  Point2D* point;
  double x,y;
  Point2D* newPoint;
  PointList* pointList = mallocPointList();

  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: ");
        int check = scanf("%lf%lf", &x,&y);
        while(check != 2)
        {
          printf("Please input the point value again: ");
          check = scanf("%lf%lf", &x,&y);
        }
        newPoint = createPoint2D(x,y);
        failure = push(newPoint, pointList);
        if(failure)
        {
          printf("Unable to add the value to the list.\n");
        }
      break;
      case POP:
        failure = pop(pointList);
        if(failure != 0)
        {
          printf("Unable to pop the value from the list.\n");
        }
      break;
      case LIST:
        print(pointList);
      break;
      case PEEK:
        peek(pointList);
      break;
      default:
        freePoint2D(newPoint);
        freePointList(pointList);
        return 0;
    }
    printf("Choice (1=add, 0=remove, 2=list, 3=peek): ");
    iNRead = scanf("%d", &iChoice);
  }
}
```

Figure 6: Source Code of playStack.c

b. Output:

```
[anguyen5@gc112m30 Assignment 6]$ make playStack
gcc -c stack.c
gcc -c Point2D.c
gcc -lm stack.o Point2D.o playStack.c -o playStack
[anguyen5@gc112m30 Assignment 6]$ ./playStack
Choice (1=add, 0=remove, 2=list, 3=peek): 1
Point value to add: 2.5
3.5
Choice (1=add, 0=remove, 2=list, 3=peek): 2
Point 0: (2.5, 3.5)
- PtLink: 0x1f8d060
- payload: 0x1f8d040
- next: (nil)

Choice (1=add, 0=remove, 2=list, 3=peek): 1
Point value to add: 5.6 8.8
Choice (1=add, 0=remove, 2=list, 3=peek): 2
Point 0: (5.6, 8.8)
- PtLink: 0x1f8d0a0
- payload: 0x1f8d080
- next: 0x1f8d060

Point 1: (2.5, 3.5)
- PtLink: 0x1f8d060
- payload: 0x1f8d040
- next: (nil)

Choice (1=add, 0=remove, 2=list, 3=peek): 3
Peeked point: (5.6, 8.8)
Choice (1=add, 0=remove, 2=list, 3=peek): 0
Popped point: (5.6, 8.8)
Choice (1=add, 0=remove, 2=list, 3=peek): 2
Point 0: (2.5, 3.5)
- PtLink: 0x1f8d060
- payload: 0x1f8d040
- next: (nil)

Choice (1=add, 0=remove, 2=list, 3=peek): 0
Popped point: (2.5, 3.5)
Choice (1=add, 0=remove, 2=list, 3=peek): 2
The list is empty
Choice (1=add, 0=remove, 2=list, 3=peek): 0
Unable to pop the value from the list.
Choice (1=add, 0=remove, 2=list, 3=peek): 1
Point value to add: 5.5 4
Choice (1=add, 0=remove, 2=list, 3=peek): 2
Point 0: (5.5, 4.0)
- PtLink: 0x1f8d0e0
- payload: 0x1f8d0c0
- next: (nil)

Choice (1=add, 0=remove, 2=list, 3=peek): 0
Popped point: (5.5, 4.0)
Choice (1=add, 0=remove, 2=list, 3=peek): 2
The list is empty
Choice (1=add, 0=remove, 2=list, 3=peek): 3
The list is empty, unable to peek the point
Choice (1=add, 0=remove, 2=list, 3=peek): u
[anguyen5@gc112m30 Assignment 6]$
```

Figure 7: Output of playStack.c

Question 2:

Create a queue ADT module to manage Point2D data. The module should handle queue creation, enqueuing, dequeuing, peek (look at the next value to be dequeued), reporting the queue contents and queue destruction. You should use your current Point2D module wherever possible. Test your module by implementing it with the playQueue.c program.

a. Source Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include "Point2D.h"
#include "queue.h"
// Other supported functions
pPtLink createPointLink(Point2D* point)
{
  pPtLink pPointLink = (pPtLink) malloc(sizeof(PtLink));
  if(pPointLink == (pPtLink)NULL)
  {
    return pPointLink;
  }
  pPointLink->payload = point;
  return pPointLink;
}

void freePointLink(pPtLink pPointLink)
{
    free(pPointLink);
    return;
}

// queue creation
PointList* mallocPointList()
{
  PointList* pointListThis = (PointList*) malloc(sizeof(PointList));
  if(pointListThis == (PointList*)NULL )
  {
    return pointListThis;
  }
  pointListThis->head = (pPtLink)NULL;
  pointListThis->tail = (pPtLink)NULL;
  pointListThis->iterator = (pPtLink)NULL;
  pointListThis->nLinks = 0;
  return pointListThis;
}

// enqueue
int enqueue(Point2D* newPoint, PointList* pointList)
{
  pPtLink pPointLink = createPointLink(newPoint);
  if(pPointLink == (pPtLink)NULL)
  {
    return 1;
  }
  if(pointList->nLinks)
  {
    pointList->tail->next = pPointLink;
  }
  else
  {
    pointList->head = pPointLink;
  }
  pointList->tail = pPointLink;
  pointList->nLinks++;
  return 0; //success
}
// dequeuing
int dequeue(PointList* pointList)
{
  if(!pointList->nLinks)
  {
```

Figure 8: Source Code of queue.c (1)

```c
    return EOF;
  }
  pPtLink temp = pointList->head;
  Point2D* tempPoint = temp->payload;
  pointList->head = temp->next;
  printf("Dequeued point: (%.1lf, %.1lf)\n", getXPoint2D(tempPoint), getYPoint2D(tempPoint));
  free(temp);
  pointList->nLinks--;
  return 0;//success
}

// peek (look at the next value to be dequeued)
void peek(PointList* pointList)
{
  if(pointList->head == (pPtLink)NULL)
  {
    printf("The list is empty, unable to peek the point\n");
  }
  else
  {
    Point2D* point = pointList->head->payload;
    printf("Peeked point: (%.1lf, %.1lf)\n", getXPoint2D(point), getYPoint2D(point));
  }
}

// reporting the queue contents
void print(PointList* pointList)
{
  pPtLink current = pointList->head;
  int i = 0;
  if(i == pointList->nLinks)
  {
    printf("The list is empty\n");
    return;
  }

  while(i < pointList->nLinks)
  {
    printf("Point %d: (%.1lf, %.1lf)\n", i, getXPoint2D(current->payload), getYPoint2D(current->payload));
    printf("Address:\n");
    printf("- PtLink: %p\n", current);
    printf("- payload: %p\n", current->payload);
    printf("- next: %p\n\n", current->next);
    i++;
    current = current->next;
  }
}

// and queue destruction
void freePointList(PointList* pointList)
{
  pPtLink current = pointList->head;
  while(pointList->head != (pPtLink)NULL)
  {
    current = pointList->head->next;
    freePointLink(pointList->head);
    pointList->head = current;
  }
  free(pointList);
  return;
}
```

Figure 9: Source Code of queue.c (2)

```c
#ifndef QUEUE_H
#define QUEUE_H
#include "Point2D.h"

pPtLink createPointLink(Point2D* point);
void freePointLink(pPtLink pPointLink);
PointList* mallocPointList();
int enqueue(Point2D* newPoint, PointList* pointList);
int dequeue(PointList* pointList);
void peek(PointList* pointList);
void print(PointList* pointList);
void freePointList(PointList* pointList);

#endif
```

Figure 10: Source Code of queue.h

```c
// playQueue.c
#include <stdio.h>
#include <stdlib.h>
#include "Point2D.h"
#include "queue.h"
#define ENQUEUE 1
#define DEQUEUE 0
#define LIST 2
#define PEEK 3
int main(int argc, char ** argv)
{
  int iChoice, iNRead, failure;
  double x,y;
  Point2D* newPoint;
  PointList* pointList = mallocPointList();

  /* 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: ");
      int check = scanf("%lf%lf", &x,&y);
      while(check != 2)
      {
        printf("Please input the point value again: ");
        check = scanf("%lf%lf", &x,&y);
      }
      newPoint = createPoint2D(x,y);
      failure = enqueue(newPoint, pointList);
      if(failure)
      {
        printf("Unable to add the value to the list.\n");
      }
      break;
      case DEQUEUE:
      failure = dequeue(pointList);
      if(failure != 0)
      {
        printf("Unable to dequeue the value from the list.\n");
      }
      break;
      case LIST:
        print(pointList);
      break;
      case PEEK:
        peek(pointList);
      break;
      default:
        freePoint2D(newPoint);
        freePointList(pointList);
        return 0;
    }
    printf("Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): ");
    iNRead = scanf("%d", &iChoice);
  }
  return EXIT_SUCCESS;
}
```

Figure 11: Source Code of playQueue.c

b. Output:

```
[anguyen5@gc112m30 Assignment 6]$ make playQueue
gcc -c queue.c
gcc -c Point2D.c
gcc -lm queue.o Point2D.o playQueue.c -o playQueue
[anguyen5@gc112m30 Assignment 6]$ ./playQueue
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 1
Point value to add: 2 3
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 1
Point value to add: 4 5
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 2
Point 0: (2.0, 3.0)
Address:
- PtLink: 0x247a060
- payload: 0x247a040
- next: 0x247a0a0

Point 1: (4.0, 5.0)
Address:
- PtLink: 0x247a0a0
- payload: 0x247a080
- next: (nil)

Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 0
Dequeued point: (2.0, 3.0)
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 2
Point 0: (4.0, 5.0)
Address:
- PtLink: 0x247a0a0
- payload: 0x247a080
- next: (nil)

Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 0
Dequeued point: (4.0, 5.0)
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 2
The list is empty
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 1
Point value to add: 4 55
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 1
Point value to add: 9.9 6
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 3
Peeked point: (4.0, 55.0)
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 2
Point 0: (4.0, 55.0)
Address:
- PtLink: 0x247a060
- payload: 0x247a0a0
- next: 0x247a0e0

Point 1: (9.9, 6.0)
Address:
- PtLink: 0x247a0e0
- payload: 0x247a0c0
- next: (nil)

Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 0
Dequeued point: (4.0, 55.0)
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 2
Point 0: (9.9, 6.0)
Address:
- PtLink: 0x247a0e0
- payload: 0x247a0c0
- next: (nil)

Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 0
Dequeued point: (9.9, 6.0)
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 3
The list is empty, unable to peek the point
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 2
The list is empty
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): u
[anguyen5@gc112m30 Assignment 6]$
```

Figure 12: Output of playQueue.c