Praktikum Programmieren, AI SS09

Übung-Nr. *8*
Aufgabe: *Speichern einer Menge von Punkten*

Bearbeiter: *F. Rosenkranz* Matr. Nr. *19895*
T. Merkel Matr. Nr. *19868*
T. Reinsch Matr. Nr. *19861*

Abgegeben am *1.7.09*

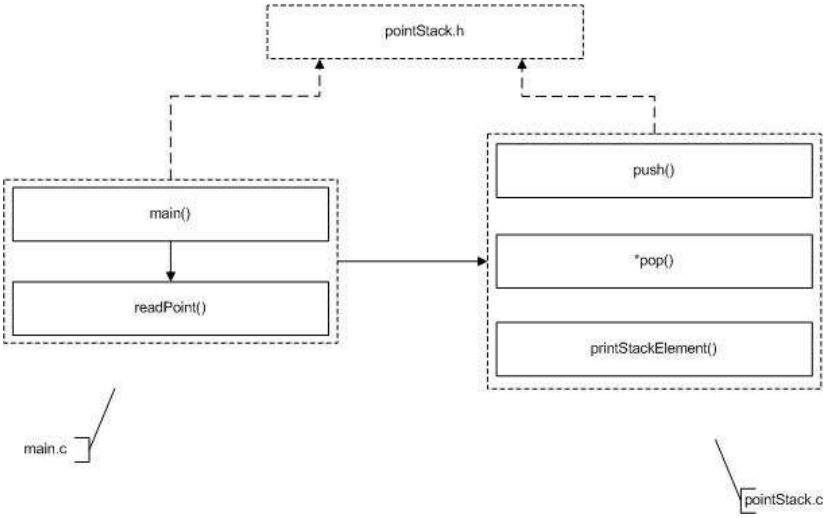
- ◯ Aufgabe erfüllt
- ◯ Bitte korrigieren:
  - Einrückung des Quellcodes
  - Sprechende Namen(Bezeichner) für Variable und Funktionen
  - Nur eine Variable pro Zeile definieren
  - Nur eine Anweisung pro Zeile
  - Ablaufdiagramm
  - Testfälle
  - weitere Anmerkungen

---

1 Beschreibung der Lösung

Das Programm liest beliebig viele Punkte ein und speichert diese. Nach jeder Eingabe wird der Benutzer gefragt, ob er einen weiteren Punkt eingeben oder die gespeicherten Eingaben ansehen möchte. Die Ausgabe der Punkte erfolgt in umgekehrter Reihenfolge wie die Eingabe, das heißt der zuletzt eingegebene Punkt wird zuerst angezeigt.

2 Ablaufdiagramm



3 Quellcode

```
1  /*-----------------------------------------------------------------------
2   * $Id: main.c,v 0.1 2009/06/17 20:14:01 drscream Exp $
3   * Copyright 2009 HS-Weingarten (thomas.merkel@hs-weingarten.de)
4   *-----------------------------------------------------------------------
5   */
6
7  #include <stdio.h>
8  #include "pointStack.h"
9  #include <stdio_ext.h>
10
11 void readPoint( point *punkt) {
12   /* read from stdin and write to stack */
13   printf( "Bitte x-Koordinate eingeben: " );
14   scanf( " %f", &punkt->x );
15   __fpurge( stdin );
16   printf( "Bitte y-Koordinate eingeben: " );
17   scanf( " %f", &punkt->y );
18   __fpurge( stdin );
19   printf( "Bitte z-Koordinate eingeben: " );
20   scanf( " %f", &punkt->z );
21   __fpurge( stdin );
22 }
23
24 int main( void ) {
25   char cmd = 'p';
26   point punkt;
27   point *ppunkt;
28
29   ppunkt = &punkt;
30
31   /* read input waiting for q => quit */
32   while( cmd != 'q' ) {
33     /* call readPoint */
34     readPoint( ppunkt );
35     push( ppunkt );
36     printf( "Neuer Punkt? (‘p‘ fuer Punkt eingeben, ‘q‘ fuer Ausgabe): " );
37     scanf( " %c", &cmd );
38   }
39
40   ppunkt = pop( );
41
42   /* print the output from stack */
43   while ( ppunkt != NULL ) {
44     printStackElement( ppunkt );
45     ppunkt = pop( );
46   }
47   return 0;
48 }
```

3.1 pointStack.c

```
1  /*-----------------------------------------------------------------------
2   * $Id: pointStack.c,v 0.1 2009/06/17 20:05:21 drscream Exp $
3   * Copyright 2009 HS-Weingarten (thomas.merkel@hs-weingarten.de)
4   *-----------------------------------------------------------------------
5   */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <math.h>
10 #include "pointStack.h"
11
12 stackPoint *startpunkt = NULL;
13
14 void push( point *newPoint ) {
15   /* push to stack / struct (pointStack.h) */
16   stackPoint *punkt;
17   punkt = (stackPoint *) malloc( sizeof( stackPoint ));
18
19   /* check out of memory ... */
20   if( punkt == NULL ) {
21     printf( "Kein freier Speicher mehr!\n");
22     exit( -1 );
23   }
24
25   punkt->p = *newPoint;
26   punkt->next = startpunkt;
27   startpunkt = punkt;
28 }
29
30 point *pop( void ) {
31   /* pop from stack / struct (pointStack.h) */
32   point *punkt;
33   if( startpunkt != NULL ) {
34     stackPoint *spunkt = startpunkt;
35     startpunkt = startpunkt->next;
36     punkt = &spunkt->p;
37     free( spunkt );
38     return punkt;
39   }
40   return 0;
41 }
42
43 void printStackElement( point *punkt) {
44   /* print to stdout */
45   printf( "x = %5.2f ", punkt->x );
46   printf( "y = %5.2f ", punkt->y );
47   printf( "z = %5.2f\n", punkt->z );
48 }
```

3.2 pointStack.h

```
1  /*-----------------------------------------------------------------------
2   * $Id: pointStack.h,v 0.1 2009/06/17 20:01:04 drscream Exp $
3   * Copyright 2009 HS-Weingarten (thomas.merkel@hs-weingarten.de)
4   *-----------------------------------------------------------------------
5   */
6
7  #ifndef POINTSTACK_H
8  #define POINTSTACK_H
9
10 typedef struct point {
11   float x;
12   float y;
13   float z;
14 } point;
15
16 typedef struct stackPoint {
17   point p;
18   struct stackPoint *next;
19 } stackPoint;
20
21 void push( point *newPoint );
22 void printStackElement( point *punkt );
23 point *pop( void );
24
25 #endif
```

4 Testfälle

```
Bitte x–Koordinate eingeben: 1
Bitte y–Koordinate eingeben: 1
Bitte z–Koordinate eingeben: 1
Neuer Punkt? (‘p‘ fuer Punkt eingeben, ‘q‘ fuer Ausgabe): p
Bitte x–Koordinate eingeben: 2
Bitte y–Koordinate eingeben: 2
Bitte z–Koordinate eingeben: 2
Neuer Punkt? (‘p‘ fuer Punkt eingeben, ‘q‘ fuer Ausgabe): p
Bitte x–Koordinate eingeben: 3
Bitte y–Koordinate eingeben: 3
Bitte z–Koordinate eingeben: 3
Neuer Punkt? (‘p‘ fuer Punkt eingeben, ‘q‘ fuer Ausgabe): p
Bitte x–Koordinate eingeben: 4
Bitte y–Koordinate eingeben: 4
Bitte z–Koordinate eingeben: 4
Neuer Punkt? (‘p‘ fuer Punkt eingeben, ‘q‘ fuer Ausgabe): p
Bitte x–Koordinate eingeben: 5
Bitte y–Koordinate eingeben: 5
Bitte z–Koordinate eingeben: 5
Neuer Punkt? (‘p‘ fuer Punkt eingeben, ‘q‘ fuer Ausgabe): q
x =  5.00 y =  5.00 z =  5.00
x =  4.00 y =  4.00 z =  4.00
x =  3.00 y =  3.00 z =  3.00
x =  2.00 y =  2.00 z =  2.00
x =  1.00 y =  1.00 z =  1.00
```