Übungsblatt 1

(13. April 2007)

Aufgabe 1 In dieser Aufgabe sollen Sie Klassen zur Beschreibung geometrischer Objekte im 2-dimensionalen Raum entwickeln. Schreiben Sie für jede Klasse geeignete Konstruktoren und eine Methode `print`.

Lösen Sie die Aufgaben jeweils einmal, indem Sie sie in C programmieren und geeignete Strukturen objektorrientiert verwenden, und einmal, indem Sie mit Klassen in C++ arbeiten.

a) Schreiben Sie eine Klasse (Struktur) `Vertex`, die Punkte im 2-dimensionalen Raum darstellt. Betrachten Sie diese Punkte als Vektoren und definieren Sie Methoden zur Addition und Subtraktion von Punkten, sowie zur Multiplikation mit einem Skalar und zur Betragsberechnung. Diese Methoden sollen das Objekt unverändert lassen, und ein Ergebnis zurückliefern.
Testen Sie die Klasse `Vertex` in einer `main`-Methode.

b) Ergänzen Sie jetzt Ihre Klasse `Vertex` um Varianten der Addition und Subtraktion, die kein Ergebnis liefern, aber das `this`-Objekt verändern.

c) Schreiben Sie eine Klasse `GeometricObject`. Ein geometrisches Objekt soll eine Weite und Höhe, sowie eine Position der oberen rechte Ecke im 2-dimensionalen Raum enthalten.

d) Implementieren Sie die folgende Methoden für Ihre Klasse `GeometricObject`:

- `bool hasWithin(Vertex* p)`, die wahr ist, wenn der Punkt `p` innerhalb der geometrischen Figur liegt.
- `bool touches(GeometricObject* that)` sei wahr, wenn es mindestens einen Punkt gibt, der in beiden Objekten liegt.

Testen Sie auch diese Methoden.

Lösung

GeometricObjectsC.h

```
1  typedef enum {false,true} bool;
2
3  typedef struct {
4    double x;
```

```
5   double y;
6 } Vertex;
7
8 Vertex* newVertex(double x, double y);
9 void printVertex(Vertex* this);
10
11 Vertex* add(Vertex* this,Vertex* that);
12 Vertex* sub(Vertex* this,Vertex* that);
13 Vertex* mult(Vertex* this,double d);
14 double length(Vertex* this);
15
16 void addMod(Vertex* this,Vertex* that);
17 void subMod(Vertex* this,Vertex* that);
18 void multMod(Vertex* this,double d);
19
20 typedef struct{
21   Vertex* pos;
22   double width;
23   double height;
24 } GeometricObject;
25
26 GeometricObject* newGeometricObject
27                      (Vertex* v,double width,double height);
28
29 void printGeometricObject(GeometricObject* this);
30
31 bool hasWithin(GeometricObject* this,Vertex* p);
32 bool touches(GeometricObject* this,GeometricObject* that);
```

GeometricObjectsC.c

```
1  #include "GeometricObjectsC.h"
2  #include <stdlib.h>
3  #include <math.h>
4  #include <stdio.h>
5
6  Vertex* newVertex(double x, double y){
7    Vertex* this = (Vertex*)malloc(sizeof(Vertex));
8    this->x=x;
9    this->y=y;
10   return this;
11 }
12 void printVertex(Vertex* this){
13   printf("(%f,%f)",this->x,this->y);
```

```
14 }
15 Vertex* add(Vertex* this,Vertex* that){
16   return newVertex(this->x+that->x,this->y+that->y);
17 }
18 Vertex* sub(Vertex* this,Vertex* that){
19   return newVertex(this->x-that->x,this->y-that->y);
20 }
21 Vertex* mult(Vertex* this,double d){
22   return newVertex(this->x*d,this->y*d);
23 }
24 double length(Vertex* this){
25   return sqrt(this->x*this->x +this->y*this->y);
26 }
27 void addMod(Vertex* this,Vertex* that){
28   this->x=this->x+that->x;
29   this->y=this->y+that->y;
30 }
31 void subMod(Vertex* this,Vertex* that){
32   this->x=this->x-that->x;
33   this->y=this->y-that->y;
34 }
35 void multMod(Vertex* this,double d){
36   this->x=this->x*d;
37   this->y=this->y*d;
38 }
39 GeometricObject* newGeometricObject
40                    (Vertex* pos,double width,double height){
41   GeometricObject* this
42        =(GeometricObject*)malloc(sizeof(GeometricObject));
43   this->pos=pos;
44   this->width=width;
45   this->height=height;
46   return this;
47 }
48 void printGeometricObject(GeometricObject* this){
49   printf("GeometricObject(");
50   printVertex(this->pos);
51   printf(",%f,%f)",this->width,this->height);
52 }
53 bool hasWithin(GeometricObject* this,Vertex* p){
54   return p->x >= this->pos->x
55              && p->x <= this->pos->x+ this->width
56       && p->y >= this->pos->y
```

```
57              && p->y <= this->pos->y+ this->height;
58  }
59  bool touches(GeometricObject* this,GeometricObject* that){
60    if (this->pos->x > that->pos->x + that -> width)
61      return false;
62    if (this->pos->x + this->width < that->pos->x)
63      return false;
64    if (this->pos->y > that->pos->y + that -> height)
65      return false;
66    if (this->pos->y + this->height < that->pos->y)
67      return false;
68    return true;
69  }
70  int main(){
71    Vertex* v1 = newVertex(1,2);
72    Vertex* v2 = newVertex(22,40);
73    printVertex(v1);  printf("\n");
74    printVertex(v2);  printf("\n");
75
76    Vertex* v3 = add(v1,v2);
77    printVertex(v3);  printf("\n");  free(v3);
78
79    v3 = sub(v1,v2);
80    printVertex(v3);  printf("\n");  free(v3);
81
82    v3 = mult(v1,2.3);
83    printVertex(v3);  printf("\n");  free(v3);
84
85    printf("%f\n",length(v1));
86
87    addMod(v1,v2);  printVertex(v1);  printf("\n");
88    subMod(v1,v2);  printVertex(v1);  printf("\n");
89    multMod(v1,2.3);  printVertex(v1);  printf("\n");
90
91    GeometricObject* g1 = newGeometricObject(v1,40,68);
92    GeometricObject* g2 = newGeometricObject(v2,19,67);
93
94    printGeometricObject(g1);  printf("\n");
95    printGeometricObject(g2);  printf("\n");
96    printf(touches(g1,g2)überuehren sich\n"
97                         :"beruehren sich nicht\n");
98
99    printf(hasWithin(g1,v2)ÿ2 liegt in g1\n"
```

```
100                          :"v2 liegt nicht in g1\n");
101
102   free(g1);free(g2);free(v1);free(v2);
103
104   return 0;
105 }
```

GeometricObjects.hpp

```
1  class Vertex{
2   public:
3    double x;
4    double y;
5
6    Vertex(double x, double y);
7    virtual void print();
8
9    virtual Vertex* add(Vertex* that);
10   virtual Vertex* sub(Vertex* that);
11   virtual Vertex* mult(double d);
12   virtual double length();
13
14   virtual void addMod(Vertex* that);
15   virtual void subMod(Vertex* that);
16   virtual void multMod(double d);
17 };
18
19 class GeometricObject{
20  public:
21   Vertex* pos;
22   double width;
23   double height;
24
25   GeometricObject(Vertex* v,double width,double height);
26   virtual void print();
27
28   virtual bool hasWithin(Vertex* p);
29   virtual bool touches(GeometricObject* that);
30
31 } ;
```

GeometricObjects.cpp

```
1  #include "GeometricObjects.hpp"
2  #include <iostream>
```

```
3  #include <cmath>
4
5  Vertex::Vertex(double x, double y){
6    this->x=x;
7    this->y=y;
8  }
9
10 void Vertex::print(){
11   std::cout<<"("<<this->x<<","<<this->y<<")";
12 }
13
14 Vertex* Vertex::add(Vertex* that){
15   return new Vertex(this->x+that->x,this->y+that->y);
16 }
17 Vertex* Vertex::sub(Vertex* that){
18   return new Vertex(this->x-that->x,this->y-that->y);
19 }
20 Vertex* Vertex::mult(double d){
21   return new Vertex(this->x*d,this->y*d);
22 }
23 double Vertex::length(){
24   return sqrt(this->x*this->x +this->y*this->y);
25 }
26 void Vertex::addMod(Vertex* that){
27   this->x=this->x+that->x;
28   this->y=this->y+that->y;
29 }
30 void Vertex::subMod(Vertex* that){
31   this->x=this->x-that->x;
32   this->y=this->y-that->y;
33 }
34 void Vertex::multMod(double d){
35   this->x=this->x*d;
36   this->y=this->y*d;
37 }
38 GeometricObject::GeometricObject
39                    (Vertex* pos,double width,double height){
40   this->pos=pos;
41   this->width=width;
42   this->height=height;
43 }
44 void GeometricObject::print(){
45   std::cout<<"GeometricObject(";
```

```
46    this->pos->print();
47    std::cout<<","<<this->width<<","<<this->height<<")";
48  }
49  bool GeometricObject::hasWithin(Vertex* p){
50    return p->x >= this->pos->x
51               && p->x <= this->pos->x+ this->width
52        && p->y >= this->pos->y
53               && p->y <= this->pos->y+ this->height;
54  }
55  bool GeometricObject::touches(GeometricObject* that){
56    if (this->pos->x > that->pos->x + that -> width)
57      return false;
58    if (this->pos->x + this->width < that->pos->x)
59      return false;
60    if (this->pos->y > that->pos->y + that -> height)
61      return false;
62    if (this->pos->y + this->height < that->pos->y)
63      return false;
64    return true;
65  }
66  int main(){
67    Vertex* v1 = new Vertex(1,2);
68    Vertex* v2 = new Vertex(22,40);
69    v1->print();  std::cout<<std::endl;
70    v2->print();  std::cout<<std::endl;
71
72    Vertex* v3 = v1->add(v2);
73    v3->print();  std::cout<<std::endl;  delete v3 ;
74
75    v3 = v1->sub(v2);
76    v3->print();  std::cout<<std::endl;  delete v3;
77
78    v3 = v1->mult(2.3);
79    v3->print();  std::cout<<std::endl;  delete v3 ;
80
81    std::cout<<v1->length()<<std::endl;
82
83    v1->addMod(v2);  v1->print();   std::cout<<std::endl;
84    v1->subMod(v2);  v1->print();   std::cout<<std::endl;
85    v1->multMod(2.3);  v1->print(); std::cout<<std::endl;
86
87    GeometricObject* g1 = new GeometricObject(v1,40,68);
88    GeometricObject* g2 = new GeometricObject(v2,19,67);
```

```
89
90    g1->print();  std::cout<<std::endl;
91    g2->print();  std::cout<<std::endl;
92
93    std::cout << (g1->touches(g2)ëberuehren sich"
94                                    :"beruehren sich nicht")
95              << std::endl;;
96    std::cout << (g1->hasWithin(v2)ÿ2 liegt in g1"
97                                      :"v2 liegt nicht in g1")
98              << std::endl;;
99
100   delete g1;delete g2;delete v1;delete v2;
101   return 0;
102 }
```