Lab Course (Basics & Lab)

Universität Bonn
Institut für Informatik II
May 23, 2016

Summer term 2016
Prof. Dr. Reinhard Klein
Stefan Krumpen, Alexander Dieckmann

Sheet 4 - Basic Classes

Submit theory via email[1] to **all** tutors, until *Mo 30/5 9:00*.
Practical assignments are corrected in the exercises on *Mo 30/5 14:30*.
Single submissions!

This sheet you will use basic structures/classes with member functions and variables, simple constructors. Details can be found e.g.

a) C++ Tutorial[2] Pages: 86 - 93.

b) Another reference that might be interesting: MIT-C-2011[3] and MIT-C-2013[4]

Assignment 1 (Structures, *2 points*)

Structures allow to combine several variables into a logical entity. For example `Vector3` (from the listing) combines x, y and z coordinates of a Vector in $\mathbb{R}^3$. Once defined structures can be used anywhere where basic types might be used (e.g. to create a variable or an array of structures). By convention we start structure names (and class names) uppercase contrary to variables that start lowercase.

a) Create implementations for the following functions:

```
struct Vector3
    {
    double x;
    double y;
    double z;
}
```

```
void print(const Vector3& a);
Vector3 add(const Vector3& a, const Vector3& b);
Vector3 substract(const Vector3& a, const Vector3& b);
Vector3 multiply(double s, const Vector3& b);
double dot_product(const Vector3& a, const Vector3& b);
double length(const Vector3& a);
Vector3 normalize(const Vector3& a);
vector3 cross_product(const Vector3& a, const Vector3& b);
```

b) Write test cases for each function. Therefore define $s = -2$, $a = (1, 2, 3)$ and $b = (-1, 2, -2)$. Output results of a call to each function. Check that results are valid.

Assignment (Theory) 2 (Basic Classes and Scopes, *2 points*)

a) Shortly describe the following terms: member variable, member function, constructor, destructor, default argument, public, private.

b) What is the output of the following code ? Please explain !

[1] Please send emails to *all* tutors and with subject: '[lab-graphics] submission'.
[2] `http://www.cplusplus.com/files/tutorial.pdf`
[3] `http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-096-introduction-to-c-january-iap-2011`
[4] `http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-s096-introduction-to-c-and-c-january-iap-2013`

```
class A {
public:
    int counter;

    A(int a = 1) {
        counter = a;
        cout << "construct A: " << counter << endl;
    }
    ~A() {
        cout << "destruct A: " << counter << endl;
    }

    void use() {
        if( -- counter > 0 ) {
            cout << counter << " times left" << endl;
        }
    }
}
```

```
void main(int argc, char
    ** argv) {
    A a1(4), a2(2);
    a1.use();
    {
        A a3;
        a3.use();
        a1.use();
        a3.use();
    }
    a1.use();
}
```

Assignment 3 (Basic Geometry, *4 points*)

Utilize the structure `Vector3` to implement the following classes. This might help you:

- For a line through $\vec{a}$ and $\vec{b}$ the closest point to c is: $\vec{a} - \left(\vec{b} - \vec{a}\right) \frac{\langle \vec{a}-\vec{c}, \vec{b}-\vec{a} \rangle}{\langle \vec{b}-\vec{a}, \vec{b}-\vec{a} \rangle}$
- Hesse normal form (HNF) describes a plane via its normal $\vec{n}$ ($\|\vec{n}\| = 1$) and the distance d of the origin. A point $\vec{x}$ is on the plane iff $\langle \vec{x}, \vec{n} \rangle - d = 0$.
- For a plane through points $\vec{a}$, $\vec{b}$ and $\vec{c}$, the hesse-normal form is: $\vec{n} = \frac{(\vec{a}-\vec{c}) \times (\vec{b}-\vec{c})}{\|(\vec{a}-\vec{c}) \times (\vec{b}-\vec{c})\|}$ and $d = -\langle \vec{a}, \vec{n} \rangle$
- For a plane $\langle \vec{x}, \vec{n} \rangle - d = 0$ and $\|\vec{n}\| = 1$ the closest point to $\vec{c}$ is: $\vec{c} - \vec{n} \cdot (d + \langle \vec{c}, \vec{n} \rangle)$
- The intersection of a line through $\vec{a}$ and $\vec{b}$ and a plane $(\vec{n}, d)$ is $\vec{a} - (\vec{b} - \vec{a}) \frac{d + \langle \vec{a}, \vec{n} \rangle}{\langle \vec{b}-\vec{a}, \vec{n} \rangle}$.

Hint: Vectors are written as $\vec{a}$. We do not differ between points and vectors. The vector from the origin to a point is identified with this point. $< \vec{a}, \vec{b} >$ is the *dot-product* of $\vec{a}$ and $\vec{b}$.

```
class Line {
  Vector3 point1, point2;

public:
  Line(const Vector3& p1,
      const Vector3& p2);

  const Vector3& get_point1() const;
  const Vector3& get_point2() const;

  Vector3 closest_point(const Vector3& p);
  double distance_to(const Vector3& p);
}
```

```
class Plane {
  Vector3 point, normal;

public:
  Plane(Vector3 p1,
      Vector3 p2, Vector3 p3);

  const Vector3& get_point() const;
  const Vector3& get_normal() const;
  double get_hnf_d() const;

  Vector3 closest_point(const Vector3& p);
  double distance_to(const Vector3& p);

  Vector3 intersect_line(const Line &l);
}
```

$$\vec{o} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \vec{p_1} = \begin{pmatrix} \sqrt{1/8} \\ \sqrt{1/8} \\ \sqrt{3/4} \end{pmatrix}, \vec{p_2} = \begin{pmatrix} 0 \\ 2\sqrt{1/8} \\ 0 \end{pmatrix}, \vec{p_3} = \begin{pmatrix} \sqrt{1/8} + \sqrt{3/8} \\ \sqrt{1/8} + \sqrt{3/8} \\ \sqrt{3/4} - \sqrt{1/4} \end{pmatrix}, \vec{q_1} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \vec{q_2} = \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}$$

a) Create the plane spanned by the points $\vec{p_1}$, $\vec{p_2}$ and $\vec{p_3}$. Output the distance of the plane to $\vec{p_1}$, $\vec{p_2}$, $\vec{p_3}$, $(\vec{p_1} + \vec{p_2} + \vec{p_3})/3$ and $\vec{p_1} + \vec{p_2} + \vec{p_3}$.

b) We want to calculate the distance from the origin to the plane. First output the 'd' parameter.

c) Recalculate using the `Plane::distance_to` function.

d) Finally use the `Plane::closest_point` function to get the closest point to the origin $\vec{o}$ on the plane $\mathcal{P}$. Output its coordinates and its distance from the origin $\vec{o}$.

e) To which sphere centered in the origin is the plane tangent?

f) Create the line through $\vec{q_1}$ and $\vec{q_2}$). Output its distance to the points $\vec{p_1}$, $\vec{p_2}$ and $\vec{p_3}$.

g) Intersect the previous line with the plane. Output the intersection point $\vec{w}$ *and* distance of $\vec{w}$ to both, the plane and the line.

Good luck !