Variabel Class

Pertemuan 5

Outline

- Classes
  - Implementing Member Functions
  - Constructors Provided by Compiler
  - Inline Implementation
  - Classes with Other Classes as Member Data

Deklarasi Class

- Untuk mendeklarasikan sebuah class dapat menggunakan kata kunci class diikuti oleh informasi tentang variabel anggota dan fungsi anggota class.
- Sebuah objek dapat diciptakan dari kelas dengan menentukan class dan nama variabel, menggunakan operator titik (.) untuk mengakses fungsi-fungsi anggota dan variabel dari objek itu.

```
class Tricycle{
        public:
                unsigned int speed;
                unsigned int wheelSize;
                pedal();
                brake();
};
```

```
Tricycle wichita;
wichita.speed = 6;
wichita.pedal();
```

Private vs Public Access

- Kata kunci “public” membuat bagian-bagian dari class dapat diakses untuk umum, class dan program lainnya dapat menggunakan objek dalam class tersebut.
- Kata kunci “private” membuat bagian-bagian dari class hanya dapat diakses dalam fungsi dari class itu sendiri. Semua variabel dan fungsi anggota secara default adalah “private”.

Implementasi Member Function

- Definisi fungsi anggota (*member function*) dimulai dengan nama class diikuti oleh operator double titik dua (: :) dan nama fungsi. Fungsi dalam class memiliki kemampuan yang sama seperti fungsi pada umumnya; mereka dapat memiliki parameter dan mengembalikan nilai.

```
void Tricycle::pedal()
{
        std::cout << "Pedaling trike\n";
}
```

Membuat Objek dengan **Constructor**

- Constructor disebut ketika sebuah objek dari kelas yang dipakai. Tugas dari constructor adalah membuat objek yang valid dari class, yang sering kali berisi inisialisasi data dari anggotanya. Ada beberapa cara untuk memanggil konstruktor saat membuat sebuah objek. Salah satunya adalah untuk mengatur sebuah objek tanpa menentukan parameter atau satu atau lebih parameter.

```
Tricycle::Tricycle(int initialSpeed)
{
        setSpeed(initialSpeed);
}
```

```
Tricycle wichita;
```

```
Tricycle wichita(5);
```

Menghapus Objek dengan **Destructor**

- Destructors membersihkan setelah objek selesai digunakan dan membebaskan setiap memori yang dialokasikan untuk objek. Sebuah destructor selalu memiliki nama class diawali dengan tilde (~), tidak ada parameter dan tidak memiliki nilai kembali. Destructor untuk kelas tidak memerlukan tindakan khusus untuk membebaskan memori, sehingga hanya berisi komentar.

```
Tricycle::~Tricycle()
{
        // do nothing
}
```

1. Implementing Member Functions

Tricycle.cpp

```
1: #include <iostream>
2:
3: class Tricycle
4: {
5:         public:
6:         int getSpeed();
7:         void setSpeed(int speed);
8:         void pedal();
9:         void brake();
10:        private:
11:        int speed;
12: };
13:
14: // get the trike's speed
15: int Tricycle::getSpeed()
16: {
17:        return speed;
18: }
19:
20: // set the trike's speed
21: void Tricycle::setSpeed(int newSpeed)
22: {
23:        if (newSpeed >= 0)
24:        {
25:                   speed = newSpeed;
26:        }
27: }
28:
```

…

```
29: // pedal the trike
30: void Tricycle::pedal()
31: {
32:       setSpeed(speed + 1);
33:       std::cout << "\nPedaling; tricycle speed " << speed << " mph\n";
34: }
35:
36: // apply the brake on the trike
37: void Tricycle::brake()
38: {
39:       setSpeed(speed - 1);
40:       std::cout << "\nBraking; tricycle speed " << speed << " mph\n";
41: }
42:
43: // create a trike and ride it
44: int main()
45: {
46:       Tricycle wichita;
47:       wichita.setSpeed(0);
48:       wichita.pedal();
49:       wichita.pedal();
50:       wichita.brake();
51:       wichita.brake();
52:       wichita.brake();
53:       return 0;
54: }
```

2. Constructors Provided by Compiler

NewTricycle.cpp

```
1: #include <iostream>
2:
3: class Tricycle
4: {
5:          public:
6:          Tricycle(int initialAge);
7:          ~Tricycle();
8:          int getSpeed();
9:          void setSpeed(int speed);
10:         void pedal();
11:         void brake();
12:         private:
13:         int speed;
14: };
15:
16: // constructor for the object
17: Tricycle::Tricycle(int initialSpeed)
18: {
19:         setSpeed(initialSpeed);
20: }
21:
22: // destructor for the object
23: Tricycle::~Tricycle()
24: {
25:         // do nothing
26: }
27:
28: // get the trike's speed
29: int Tricycle::getSpeed()
30: {
31:         return speed;
32: }
33:
```

...

```
34: // set the trike's speed
35: void Tricycle::setSpeed(int newSpeed)
36: {
37:         if (newSpeed >= 0)
38:         {
39:                     speed = newSpeed;
40:         }
41: }
42:
43: // pedal the trike
44: void Tricycle::pedal()
45: {
46:         setSpeed(speed + 1);
47:         std::cout << "\nPedaling; tricycle speed " << getSpeed() << " mph\n";
48: }
49:
50: // apply the brake on the trike
51: void Tricycle::brake()
52: {
53:         setSpeed(speed - 1);
54:         std::cout << "\nBraking; tricycle speed " << getSpeed() << " mph\n";
55: }
56:
57: // create a trike and ride it
58: int main()
59: {
60:         Tricycle wichita(5);
61:         wichita.pedal();
62:         wichita.pedal();
63:         wichita.brake();
64:         wichita.brake();
65:         wichita.brake();
66:         return 0;
67: }
```

3. Inline Implementation

Tricycle.hpp

```
1: #include <iostream>
2:
3: class Tricycle
4: {
5:         public:
6:         Tricycle(int initialSpeed);
7:         ~Tricycle();
8:         int getSpeed() const { return speed; }
9:         void setSpeed(int speed);
10:        void pedal()
11:        {
12:                  setSpeed(speed + 1);
13:                  std::cout << "\nPedaling " << getSpeed() << " mph\n";
14:        }
15:        void brake()
16:        {
17:                  setSpeed(speed - 1);
18:                  std::cout << "\nPedaling " << getSpeed() << " mph\n";
19:        }
20:        private:
21:        int speed;
22: };
```

Tricycle.cpp

```
1: #include “Tricycle.hpp”
2:
3: // constructor for the object
4: Tricycle::Tricycle(int initialSpeed)
5: {
6:         setSpeed(initialSpeed);
7: }
8:
9: // destructor for the object
10: Tricycle::~Tricycle()
11: {
12:         // do nothing
13: }
14:
15: // set the trike’s speed
16: void Tricycle::setSpeed(int newSpeed)
17: {
18:         if (newSpeed >= 0)
19:         {
20:                     speed = newSpeed;
21:         }
22: }
23:
24: // create a trike and ride it
25: int main()
26: {
27:         Tricycle wichita(5);
28:         wichita.pedal();
29:         wichita.pedal();
30:         wichita.brake();
31:         wichita.brake();
32:         wichita.brake();
33:         return 0;
34: }
```

4. Classes with Other Classes as Member Data

Rectangle.hpp

```
1: #include <iostream>
2:
3: class Point
4: {
5:         // no constructor, use default
6:         public:
7:         void setX(int newX) { x = newX; }
8:         void setY(int newY) { y = newY; }
9:         int getX() const { return x; }
10:        int getY() const { return y; }
11:        private:
12:        int x;
13:        int y;
14: };
15:
16: class Rectangle
17: {
18:        public:
19:        Rectangle(int newTop, int newLeft, int newBottom, int newRight);
20:        ~Rectangle() {}
21:
22:        int getTop() const { return top; }
23:        int getLeft() const { return left; }
24:        int getBottom() const { return bottom; }
25:        int getRight() const { return right; }
26:
```

...

```
27:         Point getUpperLeft() const { return upperLeft; }
28:         Point getLowerLeft() const { return lowerLeft; }
29:         Point getUpperRight() const { return upperRight; }
30:         Point getLowerRight() const { return lowerRight; }
31:
32:         void setUpperLeft(Point location);
33:         void setLowerLeft(Point location);
34:         void setUpperRight(Point location);
35:         void setLowerRight(Point location);
36:
37:         void setTop(int newTop);
38:         void setLeft (int newLeft);
39:         void setBottom (int newBottom);
40:         void setRight (int newRight);
41:
42:         int getArea() const;
43:
44: private:
45:         Point upperLeft;
46:         Point upperRight;
47:         Point lowerLeft;
48:         Point lowerRight;
49:         int top;
50:         int left;
51:         int bottom;
52:         int right;
53: };
```

Rectangle.cpp

```
1: #include “Rectangle.hpp”
2:
3: Rectangle::Rectangle(int newTop, int newLeft, int newBottom, int newRight)
4: {
5:        top = newTop;
6:        left = newLeft;
7:        bottom = newBottom;
8:        right = newRight;
9:
10:       upperLeft.setX(left);
11:       upperLeft.setY(top);
12:
13:       upperRight.setX(right);
14:       upperRight.setY(top);
15:
16:       lowerLeft.setX(left);
17:       lowerLeft.setY(bottom);
18:
19:       lowerRight.setX(right);
20:       lowerRight.setY(bottom);
21: }
22:
23: void Rectangle::setUpperLeft(Point location)
24: {
25:       upperLeft = location;
26:       upperRight.setY(location.getY());
27:       lowerLeft.setX(location.getX());
28:       top = location.getY();
29:       left = location.getX();
30: }
31:
```

...

```
32: void Rectangle::setLowerLeft(Point location)
33: {
34:         lowerLeft = location;
35:         lowerRight.setY(location.getY());
36:         upperLeft.setX(location.getX());
37:         bottom = location.getY();
38:         left = location.getX();
39: }
40:
41: void Rectangle::setLowerRight(Point location)
42: {
43:         lowerRight = location;
44:         lowerLeft.setY(location.getY());
45:         upperRight.setX(location.getX());
46:         bottom = location.getY();
47:         right = location.getX();
48: }
49:
50: void Rectangle::setUpperRight(Point location)
51: {
52:         upperRight = location;
53:         upperLeft.setY(location.getY());
54:         lowerRight.setX(location.getX());
55:         top = location.getY();
56:         right = location.getX();
57: }
58:
59: void Rectangle::setTop(int newTop)
60: {
61:         top = newTop;
62:         upperLeft.setY(top);
63:         upperRight.setY(top);
64: }
65:
```

...

```
66: void Rectangle::setLeft(int newLeft)
67: {
68:       left = newLeft;
69:       upperLeft.setX(left);
70:       lowerLeft.setX(left);
71: }
72:
73: void Rectangle::setBottom(int newBottom)
74: {
75:       bottom = newBottom;
76:       lowerLeft.setY(bottom);
77:       lowerRight.setY(bottom);
78: }
79:
80: void Rectangle::setRight(int newRight)
81: {
82:       right = newRight;
83:       upperRight.setX(right);
84:       lowerRight.setX(right);
85: }
86:
87: int Rectangle::getArea() const
88: {
89:       int width = right - left;
90:       int height = top - bottom;
91:       return (width * height);
92: }
93:
```

…

```
94: // compute area of the rectangle by finding corners,
95: // establish width and height and then multiply
96: int main()
97: {
98:       // initialize a local Rectangle variable
99:       Rectangle myRectangle(100, 20, 50, 80 );
100:
101:      int area = myRectangle.getArea();
102:
103:      std::cout << "Area: " << area << "\n";
104:      std::cout << "Upper Left X Coordinate: ";
105:      std::cout << myRectangle.getUpperLeft().getX() << "\n";
106:      return 0;
107: }
```

Tugas

- Modifikasi program Tricycle.cpp dengan menambahkan objek lain yaitu "second_trike", berikan nilai inisial dan cobalah method pedal() dan brake() pada nya.
- Modifikasi program NewTricycle.cpp dengan menambahkan sebuah member variable "wheelSize" yang terdiri minimal 4 nilai ketika di-set dengan sebuah accessor.
- Pisahkan class Point dari Rectangle.hpp menjadi file header sendiridan masukkan di dalam Rectangle.hpp. Jelaskan bagaimana perubahan yang terjadi dengan kompilasi Rectangle.cpp? apakah hasilnya berubah?
- Buatlah class Line yang terdiri dari 2 titik yang terhubung dengan menggunakan class Point!