

# 1 D4 - TEKKOM B

## Praktikum Class



Nama	:	Septian Bagus Jumanoro
Kelas	:	1 – D4 Teknik Komputer B
NRP	:	3221600039
Dosen	:	Dr Bima Sena Bayu Dewantara S.ST, MT.
Mata Kuliah	:	Praktikum Pemrograman Dasar 2
Hari/Tgl. Praktikum	:	Rabu, 13 april 2022



## 1. Percobaan 1 – Mengakses anggota class

```
#include <iostream>
using namespace std;
class Box
{
public:
    double length; // Length of a box
    double breadth; // Breadth of a box
    double height; // Height of a box
};

int main()
{
    system("cls");
    Box Box1; // Declare Box1 of type Box
    Box Box2; // Declare Box2 of type Box
    double volume = 0.0; // Store the volume of a box here

    // box 1 specification
    Box1.height = 5.0;
    Box1.length = 6.0;
    Box1.breadth = 7.0;

    // box 2 specification
    Box2.height = 10.0;
    Box2.length = 12.0;
    Box2.breadth = 13.0;

    // volume of box 1
    volume = Box1.height * Box1.length * Box1.breadth;
    cout << "Volume of Box1 : " << volume << endl;

    // volume of box 2
    volume = Box2.height * Box2.length * Box2.breadth;
    cout << "Volume of Box2 : " << volume << endl;

    return 0;
}
```

## Output

PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

```
Volume of Box1 : 210
Volume of Box2 : 1560
PS D:\Kuliah\SMT 2\PD 2\Prak6> □
```

## Kesimpulan

Berdasarkan program tersebut, diketahui menggunakan fungsi Class Box yang berisikan beberapa type data .Lalu di dalam int main() akan mendeklarasikan sebuah class dan juga mempunyai variable volume, pada variable volume sendiri dia memiliki fungsi sebagai penampung data yang akan di operasikan, dapat kita lihat bahwa pada variable volume akan mengoperasikan sebuah data.

Lalu data pada Class Box Box1 sendiri memiliki data tertentu yang menyangkut pautkan dengan type data sebelumnya (length, height, breadth). Begitu juga dengan data pada Class Box Box2 yang sama dengan Class Box 1, namun value data yangdi terima adalah berbeda.

## 2. Percobaan 2 – Fungsi sebagai anggota class

```
#include <iostream>
using namespace std;
class Box
{
public:
    double length; // Length of a box
    double breadth; // Breadth of a box
    double height; // Height of a box

    // Member functions declaration
    double getVolume(void);
    void setLength(double len);
    void setBreadth(double bre);
    void setHeight(double hei);
};

// Member functions definitions
double Box::getVolume(void)
{
    return length * breadth * height;
}

void Box::setLength(double len)
{
    length = len;
}

void Box::setBreadth(double bre)
{
    breadth = bre;
}
```

```

void Box::setHeight(double hei)
{
    height = hei;
}

// Main function for the program
int main( )
{
    system("cls");
    Box Box1; // Declare Box1 of type Box
    Box Box2; // Declare Box2 of type Box
    double volume = 0.0; // Store the volume of a box here

    // box 1 specification
    Box1.setLength(6.0);
    Box1.setBreadth(7.0);
    Box1.setHeight(5.0);

    // box 2 specification
    Box2.setLength(12.0);
    Box2.setBreadth(13.0);
    Box2.setHeight(10.0);

    // volume of box 1
    volume = Box1.getVolume();
    cout << "Volume of Box1 : " << volume <<endl;

    // volume of box 2
    volume = Box2.getVolume();
    cout << "Volume of Box2 : " << volume <<endl;

    return 0;
}

```

## Output

PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

```

Volume of Box1 : 210
Volume of Box2 : 1560
PS D:\Kuliah\SMT 2\PD 2\Prak6> 

```

## Kesimpulan

Berdasarkan program tersebut, cara kerjanya hampir sama dengan nomor 1. Hanya terdapat beda pada saat pemanggilan anggota. Untuk nomor 1 menggunakan variabel sebagai anggota class, sedangkan nomor 2 menggunakan fungsi sebagai anggota class.

### 3. Percobaan 3 – Akses anggota class bertipe public

```
#include <iostream>
using namespace std;
class Line
{
    public:
    double length;

    void setLength(double len);
    double getLength(void);
};

// Member functions definitions
double Line:: getLength(void)
{
    return length ;
}

void Line :: setLength(double len)
{
    length = len;
}

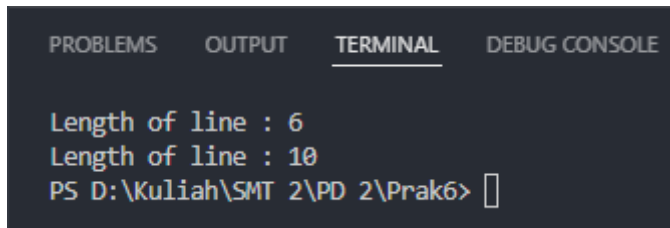
// Main function for the program
int main()
{
    system("cls");
    Line line;

    // set line length
    line.setLength(6.0);
    cout << "Length of line : " << line. getLength() <<endl;

    // set line length without member function
    line.length = 10.0; // OK: because length is public
    cout << "Length of line : " << line. length <<endl;

    return 0;
}
```

## Output



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Length of line : 6
Length of line : 10
PS D:\Kuliah\SMT 2\PD 2\Prak6> 
```

## Kesimpulan

Berdasarkan program tersebut, anggota class yang bertipe public dapat diakses dalam fungsi main. Karena tipe public tersebut merupakan tipe untuk variabel yang dapat diakses dimana saja terutama dalam fungsi main.

### 4. Percobaan 4 – Akses anggota class bertipe private

```
#include <iostream>
using namespace std;
class Box
{
    public:
        double length;
        void setWidth(double wid);
        double getWidth(void);

    private:
        double width;
};

// Member functions definitions
double Box:: getWidth(void)
{
    return width ;
}

void Box:: setWidth(double wid)
{
    width = wid;
}

// Main function for the program
int main()
{
    system("cls");
```

```

Box box;

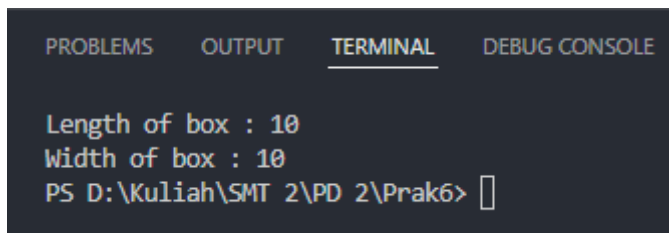
// set box length without member function
box.length = 10.0; // OK: because length is public
cout << "Length of box : " << box.length << endl;

// set box width without member function
// box.width = 10.0; // Error: because width is private
box.setWidth(10.0); // Use member function to set it.
cout << "Width of box : " << box.getWidth() << endl;

return 0;
}

```

## Output



```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Length of box : 10
Width of box : 10
PS D:\Kuliah\SMT 2\PD 2\Prak6> 

```

## Kesimpulan

Berdasarkan program tersebut, merupakan kebalikan dari public yaitu private. Dimana private merupakan tipe variabel yang hanya bisa diakses di dalam class saja. Namun jika ingin memanggil anggota class yang private maka program tersebut akan error.

## 5. Percobaan 5 – Akses anggota class bertipe protected

```

#include <iostream>
using namespace std;
class Box
{
    protected:
    double width;
};

class SmallBox:Box // SmallBox is the derived class.
{

```

```

    public:
    void setSmallWidth(double wid);
    double getSmallWidth();
};

// Member functions of child class
double SmallBox:: getSmallWidth()
{
    return width;
}

void SmallBox:: setSmallWidth(double wid)
{
    width = wid;
}

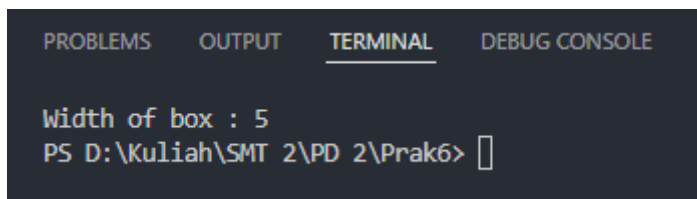
// Main function for the program
int main()
{
    system("cls");
    SmallBox box;

    // set box width using member function
    box. setSmallWidth(5.0);
    cout << "Width of box : " << box.getSmallWidth() << endl;

    return 0;
}

```

## Output



```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Width of box : 5
PS D:\Kuliah\SMT 2\PD 2\Prak6> 

```

## Kesimpulan

Berdasarkan program tersebut, anggota variabel yang bertipe protected dapat diakses sedangkan yang bertipe private tidak. Untuk cara mengaksesnya sendiri dapat menggunakan fungsi class turunan yang masih bisa mengakses anggota class induk.



## 6. Percobaan 6 – Class constructor

```
#include <iostream>
using namespace std;
class Line
{
    public:
    void setLength( double len );
    double getLength( void );
    Line(); // This is the constructor

    private:
    double length;
};

// Member functions definitions including constructor
Line:: Line(void)
{
    cout << "Object is being created" << endl;
}

void Line:: setLength( double len )
{
    length = len;
}

double Line:: getLength()
{
    return length;
}

// Main function for the program
int main( )
{
    system("cls");
    Line line;

    // set line length
    line. setLength(6.0);
    cout << "Length of line : " << line. getLength() << endl;

    return 0;
}
```

## Output

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Object is being created
Length of line : 6
PS D:\Kuliah\SMT 2\PD 2\Prak6> □
```

## Kesimpulan

Berdasarkan program tersebut, class constructor merupakan fungsi anggota yang berada dalam class, dan di eksekusi ketika terpanggil dalam fungsi main. Untuk membuatnya harus menggunakan fungsi diluar class dan inisialisasinya tetap memanggil class utama diikuti tanda ::

### 7. Percobaan 7 – Class constructor dengan parameter

```
#include <iostream>
using namespace std;
class Line
{
    public:
        void setLength(double len);
        double getLength(void);
        Line(double len); // This is the constructor

    private: double length;
};

// Member functions definitions including constructor
Line:: Line(double len)
{
    cout << "Object is being created, length = " << len << endl;
    length = len;
}

void Line:: setLength(double len)
{
    length = len;
}

double Line:: getLength()
{
    return length;
}
```

```
// Main function for the program
int main()
{
    system("cls");
    Line line(10.0);

    // get initially set length.
    cout << "Length of line : " << line.getLength() << endl;

    // set line length again
    line.setLength(6.0);
    cout << "Length of line : " << line.getLength() << endl;

    return 0;
}
```

### Output sebelum diganti

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Object is being created, length = 10
Length of line : 10
Length of line : 6
PS D:\Kuliah\SMT 2\PD 2\Prak6> 
```

### Output setelah diganti

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Object is being created, length = 10
Length of line : 10
Length of line : 6
PS D:\Kuliah\SMT 2\PD 2\Prak6> 
```

### Kesimpulan

Berdasarkan program tersebut, keduanya hampir sama. Namun pada program sebelum diubah menggunakan fungsi khusus diluar class untuk membuat constructor. Setelah program diubah fungsi constructor memiliki parameter yang dapat di parsingkan.

## 8. Percobaan 8 – Class destructor

```
#include <iostream>
using namespace std;
class Line
{
    public:
    void setLength( double len ); double getLength( void );
    Line(); // This is the constructor declaration
    ~Line(); // This is the destructor: declaration

    private:
    double length;
};

// Member functions definitions including constructor
Line::Line()
{
    cout << "Object is being created" << endl;
}

Line::~~Line()
{
    cout << "Object is being deleted" << endl;
}

void Line::setLength(double len)
{
    length = len;
}

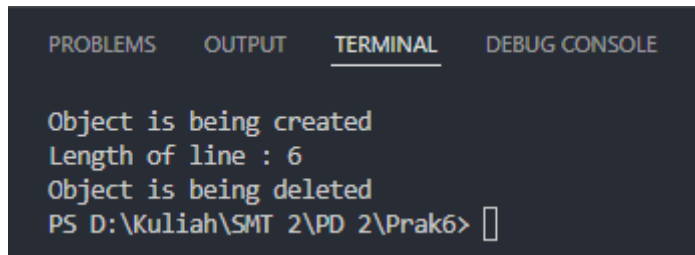
double Line:: getLength()
{
    return length;
}

// Main function for the program
int main()
{
    system("cls");
    Line line;

    // set line length
    line.setLength(6.0);
    cout << "Length of line : " << line.getLength() << endl;
```

```
return 0;  
}
```

## Output

A screenshot of a C++ IDE's terminal window. The terminal has four tabs at the top: 'PROBLEMS', 'OUTPUT', 'TERMINAL' (which is selected and underlined), and 'DEBUG CONSOLE'. The output text in the terminal is: 'Object is being created', 'Length of line : 6', 'Object is being deleted', and a command prompt 'PS D:\Kuliah\SMT 2\PD 2\Prak6>' followed by a cursor. The text is displayed in a light blue monospace font on a dark background.

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  
  
Object is being created  
Length of line : 6  
Object is being deleted  
PS D:\Kuliah\SMT 2\PD 2\Prak6> █
```

## Kesimpulan

Berdasarkan program tersebut, destructor merupakan kebalikan dari constructor dimana fungsi khusus yang akan menghapus sebuah objek ketika program akan berakhir. Fungsi destructor hampir sama dengan fungsi () atau free atau fungsi (void) pada memori dinamis.