

# 1 D4 - TEKKOM B

## PRAKTIKUM 6 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, 20 April 2022



## 9. Percobaan 9 – Copy Constructor

```
#include <iostream>
using namespace std;
class Line{
public:
    int getLength(void);
    Line(int len); // simple constructor
    Line(const Line &obj); // copy constructor
    ~Line(); // destructor

private:
    int *ptr;
};

// Member functions definitions including constructor
Line::Line(int len){
    cout << "Normal constructor allocating ptr" << endl;
    // allocate memory for the pointer;
    ptr = new int;
    *ptr = len;
}

Line::Line(const Line &obj){
    cout << "Copy constructor allocating ptr." << endl;
    ptr = new int;
    *ptr = *obj.ptr; // copy the value
}

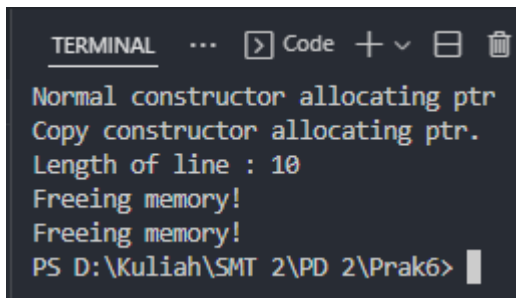
Line::~~Line(void){
    cout << "Freeing memory!" << endl;
    delete ptr;
}

int Line::getLength(void){
    return *ptr;
}

void display(Line obj){
    cout << "Length of line : " << obj.getLength() << endl;
}

// Main function for the program
int main(){
    system("cls");
    Line line(10);
    display(line) ;
    return 0;
}
```

## Output

A terminal window with a dark background and light-colored text. The text shows the execution of a C++ program. It starts with 'Normal constructor allocating ptr', followed by 'Copy constructor allocating ptr.', then 'Length of line : 10'. Finally, it shows 'Freeing memory!' twice, indicating memory deallocation for both objects. The prompt at the bottom is 'PS D:\Kuliah\SMT 2\PD 2\Prak6>'.

## Kesimpulan

Pada program tersebut, memiliki 2 tipe constructor, yaitu simple constructor dan copy constructor. Simple constructor akan mengalokasikan memori seperti biasa, sedangkan untuk copy constructor akan mengcopy constructor tersebut. Karena terdapat 2 constructor yang dibuat, maka pembebasan memorynya terjadi 2 kali pada akhir program.

```
#include <iostream>
using namespace std;
class Line{
public:
    int getLength(void);
    Line(int len); // simple constructor
    Line(const Line &obj); // copy constructor
    ~Line(); // destructor

private:
    int *ptr;
};

// Member functions definitions including constructor
Line::Line(int len){
    cout << "Normal constructor allocating ptr" << endl;
    // allocate memory for the pointer;
    ptr = new int;
    *ptr = len;
}

Line::Line(const Line &obj){
    cout << "Copy constructor allocating ptr." << endl;
    ptr = new int;
    *ptr = *obj.ptr; // copy the value
}

Line::~~Line(void){
```

```

        cout << "Freeing memory!" << endl;
        delete ptr;
    }

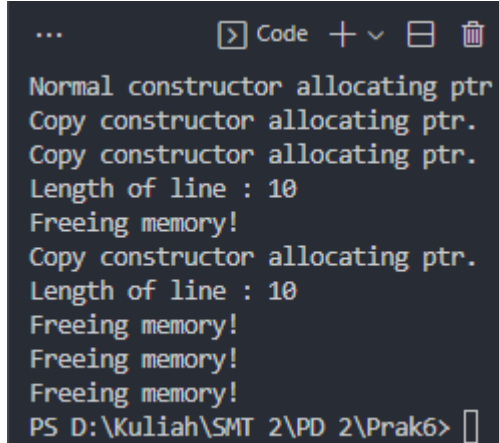
    int Line::getLength(void){
        return *ptr;
    }

    void display(Line obj){
        cout << "Length of line : " << obj.getLength() << endl;
    }

    // Main function for the program
    int main(){
        system("cls");
        Line line1(10);
        Line line2 = line1; // This also calls copy constructor
        display(line1);
        display(line2);
    }

```

## Output



```

... Code + v [ ] [X]
Normal constructor allocating ptr
Copy constructor allocating ptr.
Copy constructor allocating ptr.
Length of line : 10
Freeing memory!
Copy constructor allocating ptr.
Length of line : 10
Freeing memory!
Freeing memory!
Freeing memory!
PS D:\Kuliah\SMT 2\PD 2\Prak6>

```

## Kesimpulan

Pada program tersebut sama dengan percobaan sebelumnya, hanya inisialisasinya dilakukan sebanyak 2 kali, sehingga terdapat 4 memori yang dibuat.

## 10. Percobaan 10 – Friend Function

```
#include <iostream>
using namespace std;

class Box{
    double width;
public:
    friend void printWidth(Box box);
    void setWidth(double wid);
};

// Member function definition
void Box::setWidth(double wid){
    width = wid;
}

// Note: printWidth() is not a member function of any class.
void printWidth(Box box){
    /* Because printWidth() is a friend of Box, it can
    directly access any member of this class */
    cout << "Width of box : " << box.width << endl;
}

// Main function for the program
int main(){
    system("cls");
    Box box;
    // set box width without member function
    box.setWidth(10.0);
    // Use friend function to print the width.
    printWidth(box);
    return 0;
}
```

### Output

```
... Code + v [ ] [X]
Width of box : 10
PS D:\Kuliah\SMT 2\PD 2\Prak6> [ ]
```

## Kesimpulan

Pada program tersebut, menggunakan friend function pada class yang fungsinya dapat mengakses semua member di dalam class dimana ia dipanggil. Sehingga pada program ini, printwidth dapat menampilkan nilai dari variabel box.

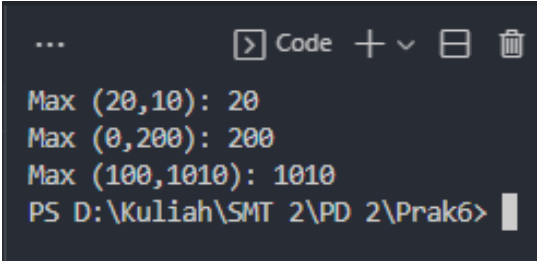
## 11. Percobaan 11 – Inline Functions

```
#include <iostream>
using namespace std;

inline int Max(int x, int y){
    return (x > y)? x : y;
}

// Main function for the program
int main(){
    system("cls");
    cout << "Max (20,10): " << Max(20,10) << endl;
    cout << "Max (0,200): " << Max(0,200) << endl;
    cout << "Max (100,1010): " << Max(100,1010) << endl;
    return 0;
}
```

## Output



The screenshot shows a code editor window with a dark background. At the top, there are icons for 'Code', a plus sign, a minus sign, a window icon, and a trash icon. Below these icons, the output of the program is displayed in a monospaced font. The output consists of three lines: 'Max (20,10): 20', 'Max (0,200): 200', and 'Max (100,1010): 1010'. Below the output, there is a prompt 'PS D:\Kuliah\SMT 2\PD 2\Prak6>' followed by a cursor.

```
... Code + - [ ] [X]
Max (20,10): 20
Max (0,200): 200
Max (100,1010): 1010
PS D:\Kuliah\SMT 2\PD 2\Prak6> |
```

## Kesimpulan

Pada program tersebut, fungsinya dapat digunakan didalam class tetapi terbatas. Sehingga saat dilakukan di luar class, inline fuction akan berjalan seperti biasa.

## 12. Percobaan 12 – This Pointer

```
#include <iostream>
using namespace std;

class Box{
public:
    // Constructor definition
    Box(double l=2.0,double b=2.0,double h=2.0){
        cout <<"Constructor called." << endl;
        length = l;
        breadth = b;
        height = h;
    }

    double Volume(){
        return length * breadth * height;
    }

    int compare(Box box){
        return this->Volume() > box.Volume();
    }

private:
    double length; // Length of a box
    double breadth; // Breadth of a box
    double height; // Height of a box
};

int main(void){
    system("cls");
    Box Box1(3.3, 1.2, 1.5); // Declare box1
    Box Box2(8.5, 6.0, 2.0); // Declare box2
    if(Box1.compare(Box2)){
        cout << "Box2 is smaller than Box1" <<endl;
    }
    else{
        cout << "Box2 is equal to or larger than Box1" <<endl;
    }
    return 0;
}
```

## Output

```
... Code + - [ ] [X]  
Constructor called.  
Constructor called.  
Box2 is equal to or larger than Box1  
PS D:\Kuliah\SMT 2\PD 2\Prak6> [ ]
```

## Kesimpulan

Pada program tersebut menggunakan fungsi compare. Dimana penunjukkan nilai yang terjadi di fungsi compare, dilakukan this dengan tanda panah untuk menunjuk nilai yang telah diberikan. Sehingga dapat disimpulkan dapat dilakukan operasi this pointer di fungsi dalam class.

## 13. Percobaan 13 – Pointer to C++ Class

```
#include <iostream>  
using namespace std;  
  
class Box{  
public:  
    // Constructor definition  
    Box(double l=2.0,double b=2.0,double h=2.0){  
        cout <<"Constructor called." << endl;  
        length = l;  
        breadth = b;  
        height = h;  
    }  
  
    double Volume(){  
        return length * breadth * height;  
    }  
  
private:  
    double length; // Length of a box  
    double breadth; // Breadth of a box  
    double height; // Height of a box  
};  
  
int main(void){  
    system("cls");  
    Box Box1(3.3, 1.2, 1.5); // Declare box1
```



```

Box Box2(8.5, 6.0, 2.0); // Declare box2
Box *ptrBox; // Declare pointer to a class.
// Save the address of first object
ptrBox = &Box1;
// Now try to access a member using member access operator
cout << "Volume of Box1: " << ptrBox->Volume() << endl;
// Save the address of first object
ptrBox = &Box2;
// Now try to access a member using member access operator
cout << "Volume of Box2: " << ptrBox->Volume() << endl;
return 0;
}

```

## Output

```

... Code + - [ ] [X]
Constructor called.
Constructor called.
Volume of Box1: 5.94
Volume of Box2: 102
PS D:\Kuliah\SMT 2\PD 2\Prak6> 

```

## Kesimpulan

Pada program tersebut didalam gungsi main, dilakukan deklarasi pointer ke box yaitu \*ptrbox. Pointer tersebut juga digunakan untuk mengcopy alamat dari box1 dan digunakan untuk menampilkan dengan menunjuk fungsi volume dengan tanda panah. Sehingga dapat disimpulkan pointer dapat dijalankan seperti pointer biasa pada program main.

## 14. Percobaan 14 – Anggota statik dari sebuah class

```

#include <iostream>
using namespace std;
class Box{
public:
    static int objectCount;
    // Constructor definition
    Box(double l=2.0, double b=2.0, double h=2.0){
        cout <<"Constructor called." << endl;
        length = l;
        breadth = b;
        height = h;
        // Increase every time object is created
    }
}

```

```

        objectCount++;
    }
    double Volume(){
        return length * breadth * height;
    }
private:
    double length; // Length of a box
    double breadth; // Breadth of a box
    double height; // Height of a box
};

// Initialize static member of class Box
int Box:: objectCount = 0;

int main(void){
    system("cls");
    Box Box1(3.3, 1.2, 1.5); // Declare box1
    Box Box2(8.5, 6.0, 2.0); // Declare box2
    // Print total number of objects.
    cout << "Total objects: " << Box::objectCount << endl;
    return 0;
}

```

## Output

```

... Code + v [ ] [X]
Constructor called.
Constructor called.
Total objects: 2
PS D:\Kuliah\SMT 2\PD 2\Prak6> 

```

## Kesimpulan

Pada program tersebut didalam anggota statik dari sebuah class ini, dilakukan penginisialisasian variable statik. Penginisialisasian variable statik tersebut digunakan untuk Class Box di luar class. Di dalam class variable tersebut akan bertambah setiap constructor yang telah dibuat. Sehingga dapat disimpulkan class dapat memuat variable statik yang telah diinisialisasikan ke dalam classtersebut.

## 15. Percobaan 15 – Anggota Fungsi Statik

```
#include <iostream>
using namespace std;
class Box{
public:
    static int objectCount;

    // Constructor definition
    Box(double l=2.0, double b=2.0, double h=2.0){
        cout << "Constructor called." << endl;
        length = l;
        breadth = b;
        height = h;
        // Increase every time object is created
        objectCount++;
    }

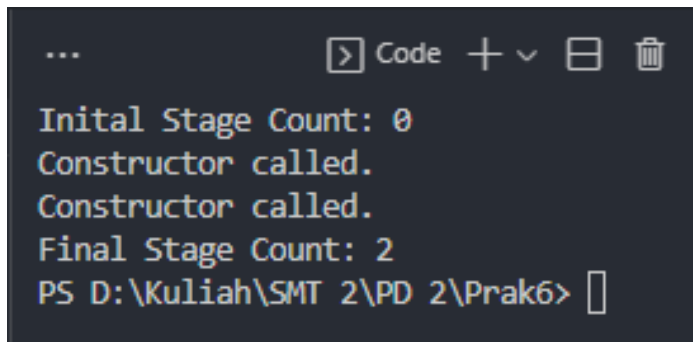
    double Volume(){
        return length * breadth * height;
    }

    static int getCount(){
        return objectCount;
    }
private:
    double length; // Length of a box
    double breadth; // Breadth of a box
    double height; // Height of a box
};

// Initialize static member of class Box
int Box:: objectCount = 0;

int main(void){
    system("cls");
    // Print total number of objects before creating object.
    cout << "Initial Stage Count: " << Box::getCount() << endl;
    Box Box1(3.3, 1.2, 1.5); // Declare box1
    Box Box2(8.5, 6.0, 2.0); // Declare box2
    // Print total number of objects after creating object.
    cout << "Final Stage Count: " << Box::getCount() << endl;
    return 0;
}
```

## Output



```
... Code + v [icon] [icon]
Initial Stage Count: 0
Constructor called.
Constructor called.
Final Stage Count: 2
PS D:\Kuliah\SMT 2\PD 2\Prak6> [ ]
```

## Kesimpulan

Pada percobaan ini, didalam program dilakukan penginisiasilasian variable static objectCount di class box. Kemudian objectCount diakses dalam constructor dan getCount, karena nilai awal dari penginisialisasian adalah 0, maka getCount pertama bernilai 0 karena belum dilakukan pendefinisian class di dalam program main. Setelah dilakukan pendefinisian, nilai dari box menjadi sama dengan constructor yang telah dibuat. Jadi dapat disimpulkan anggota statik dapat diakses oleh member class.