Chapter 9 – Operator Types dan Operator Overloading

Agenda

Pada chapter ini kita akan membahas beberapa topik tentang penggunaan Operator Types dan Operator Overloading, adapun topik yang akan dibahas adalah

- Menggunakan operator keyword
- Notasi prefik dan postfix
- Menggunakan unary dan binary operator
- Conversion operator
- Subscript operator
- Function operator

Operator pada C++

Pada bab awal kita sudah mempelajari berbagai macam operator (+, -, /, >, <) yang dapat digunakan pada tipe data yang sudah ada di C++ seperti int, float, bool, dll. Namun jika anda ingin menggunakan operator tersebut pada tipe data yang anda definisikan sendiri seperti tipe data Class, maka anda dapat menggunakan keyword operator .

```
return_type operator operator_symbol (...parameter list...);
```

Penggunaan keyword operator sebenarnya mirip dengan penggunaan fungsi , hanya anda dapat menggunakan operator symbol seperti (+, -, >,< , =, dll). Mungkin anda bertanya kenapa harus menggunakan keyword operator jika anda dapat menggunakan fungsi ? Ilustrasi dibawah ini akan menunjukan kenapa kita membutuhkan operator.

```
CKataString strKata1(“Hello”);
CKataString strKata2(“World”);
```

Jika anda menginginkan untuk menggabungkan kedua kata tersebut anda dapat membuat function Concatenate seperti berikut:

```
CKataString strGabung;
strGabung = strKata1.Concatenate(strKata2);
```

Selain cara seperti diatas akan lebih natural jika anda menulis kode sebagai berikut:

```
CKataString strGabung;
strGabung = strKata1 + strKata2;
```

Walaupun hasil dari kedua cara penulisan diatas sama, namun penggunaan operator + untuk menggabungkan string akan lebih intuitif dan mudah dipahami.

Ada 2 macam operator yang terdapat di C++ yaitu unary dan binary.

Unary Operator

Unary operator hanya mempunyai single operand saja, cara penulisan unary operator adalah sebagai berikut.

```
return_type operator operator_type (parameter_type)
{
// ... implementation
}
```

Tipe dari unary operator yang dapat digunakan adalah

Operator	Name
++	Increment
--	Decrement
*	Pointer dereference
->	Member selection
!	Logical NOT
&	Address-of
~	One's complement
+	Unary plus
–	Unary negation
Conversion operators	Conversion operators

Pada **Labs1** akan ditunjukan penggunaan operator increment. Pada contoh dibawah ini kita akan membuat Class Kalender yang mempunyai tiga class member yang merepresentasikan hari, bulan, dan tahun (tipe integer), anda dapat menggunakan operator ++ untuk menambahkan hari .

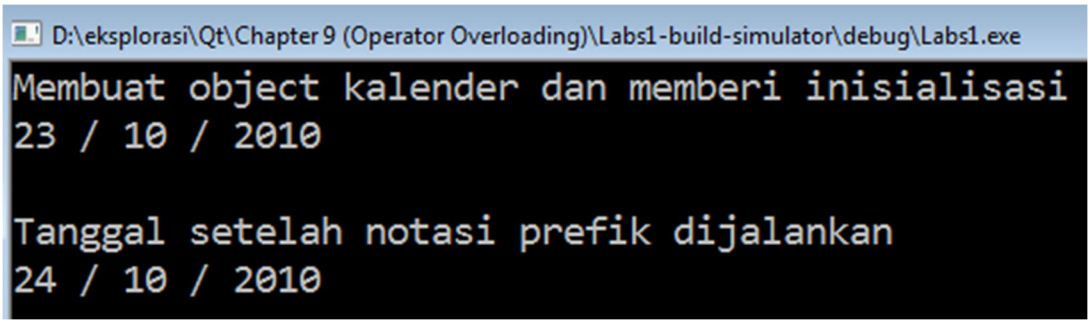
Labs.1 Menggunakan Increment Operator (Notasi prefix)

1. Buka Qt Creator dan buat project Qt Console Application baru dengan nama **Labs1**, kemudian tulis kode berikut.

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;

class Kalender
{
private:
  int _hari;
  int _bulan;
  int _tahun;
  void AddHari(int hari)
  {
      _hari += hari;
      if(_hari > 30)
      {
          AddBulan(_hari/30);
          _hari %= 30;
```

```
        }
    }
    void AddBulan(int bulan)
    {
        _bulan += bulan;
        if(_bulan > 12)
        {
            AddTahun(_bulan/12);
            _bulan %= 12;
        }
    }
    void AddTahun(int tahun)
    {
        _tahun += tahun;
    }
public:
    Kalender(int hari,int bulan, int tahun):_hari(hari),_bulan(bulan),_tahun(tahun){ }
    Kalender &operator ++ ()
    {
        AddHari(1);
        return *this;
    }
    void TampilData()
    {
        cout << _hari << " / " << _bulan << " / " << _tahun << endl;
    }
};

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    Kalender objKal(23,10,2010);
    cout << "Membuat object kalender dan memberi inisialisasi" << endl;
    objKal.TampilData();
    cout << endl;

    ++objKal;
    cout << "Tanggal setelah notasi prefik dijalankan " << endl;
    objKal.TampilData();
    return a.exec();
}
```

2. Kemudian jalankan kode diatas dengan menekan tombol Ctrl+R, outputnya adalah sebagai berikut.



Analisa:

- Class Kalender berisi tiga member variabel yaitu **_hari**, **_bulan**, dan **_tahun** yang merepresentasikan waktu tertentu.
- **operator ++** digunakan untuk menambahkan 1 hari kedalam objek **Kalender**, dengan menggunakan **operator ++** penulisan menjadi lebih intuitif dan mudah dipahami, misal untuk menambahkan 1 hari kedalam objek **Kalender** anda dapat menuliskan **++objKal**.
- Untuk memanggil **operator ++** pada program diatas digunakan notasi prefix (tanda ++ dituliskan sebelum nama objek).
- Karena kode diatas menggunakan notasi prefix maka pada operator ++ akan mengakses objek *by reference*.

Ada perbedaan penulisan notasi prefix dan postfix, sebagai contoh anda dapat melihat kode dibawah ini.

```
int bil1 = 22;
int bil2 = bil1++;
//mengcopy nilai lama dari bil1
cout << "bil2 : " << bil2;
//nilai bil1 setelah di increment
cout << "bil1 : " << bil1;
```

Nilai **bil2** adalah 22, karena yang dimasukan kedalam **bil2** adalah nilai lama dari **bil1**, baru setelah itu **bil1** diincrement.

Untuk contoh dibawah ini kita akan mencoba menggunakan notasi postfix, dengan notasi postfix yang dilakukan adalah menduplikat objek yang diinputkan, melakukan increment dan mengembalikan objek tersebut *by value*.

Labs.2 Menggunakan Operator Increment (notasi postfix)

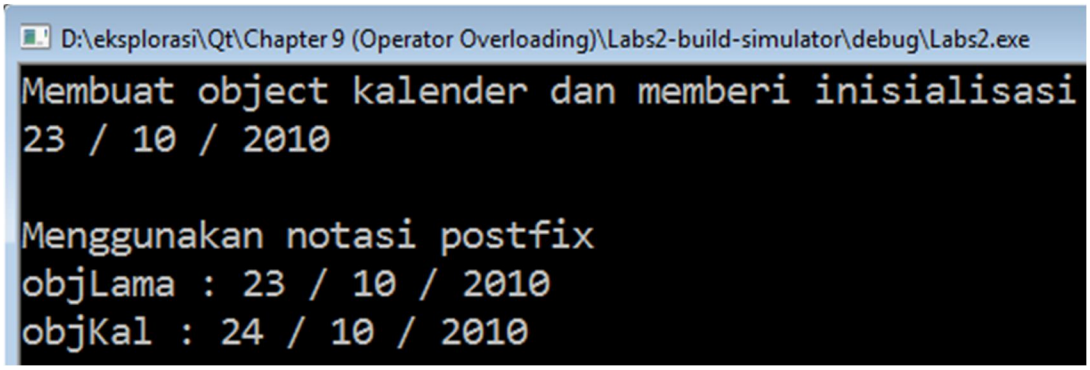
1. Buka Qt Creator dan buat project Qt Console Application baru dengan nama **Labs2**, kemudian tulis kode berikut.

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;

class Kalender
{
private:
  int _hari;
```

```
  int _bulan;
  int _tahun;
  void AddHari(int hari)
  {
      _hari += hari;
      if(_hari > 30)
      {
          AddBulan(_hari/30);
          _hari %= 30;
      }
  }
  void AddBulan(int bulan)
  {
      _bulan += bulan;
      if(_bulan > 12)
      {
          AddTahun(_bulan/12);
          _bulan %= 12;
      }
  }
  void AddTahun(int tahun)
  {
      _tahun += tahun;
  }
public:
  Kalender(int hari,int bulan, int tahun):_hari(hari),_bulan(bulan),_tahun(tahun){ }
  Kalender operator ++ (int)
  {
      Kalender objKal(_hari,_bulan,_tahun);
      AddHari(1);
      return objKal;
  }
  void TampilData()
  {
      cout << _hari << " / " << _bulan << " / " << _tahun << endl;
  }
};

int main(int argc, char *argv[])
{
  QCoreApplication a(argc, argv);
  Kalender objKal(23,10,2010);
  cout << "Membuat object kalender dan memberi inisialisasi" << endl;
  objKal.TampilData();
  cout << endl;
  cout << "Menggunakan notasi postfix" << endl;

  //menggunakan notasi postfix
  Kalender objLama(objKal++);
  cout << "objLama : ";
  objLama.TampilData();
  cout << "objKal : ";
  objKal.TampilData();
  return a.exec();
```

```
}
```

2. Kemudian jalankan kode diatas dengan menekan tombol Ctrl+R, outputnya adalah sebagai berikut.



Analisa:

- Output yang dihasilkan akan sama dengan kode sebelumnya, hanya saja penulisan operator ++ menggunakan notasi postfix.
- Karena menggunakan postfix maka yang dilakukan pada operator ++ adalah mengkopi objek yang lama, menambahkan data, kemudian mengembalikan objek tersebut *by value*.

Conversion Operator

Bagaimana jika anda menginginkan statement **`int bil = Kalender(23,10,2010)`** memiliki arti? Untuk itu anda perlu mengkonversi dari objek **`Kalender`** menjadi tipe data integer, dengan demikian anda akan dengan mudah mengirimkan data **`Kalender`** ke module lain yang hanya menerima parameter bertipe integer.

Anda dapat melakukan konversi diatas dengan menggunakan conversion operator yang mempunyai syntax sebagai berikut:

```
operator conversion_type();
```

Jadi jika anda menghendaki mengkonversi tipe Kalender menjadi int anda dapat menggunakan operator berikut.

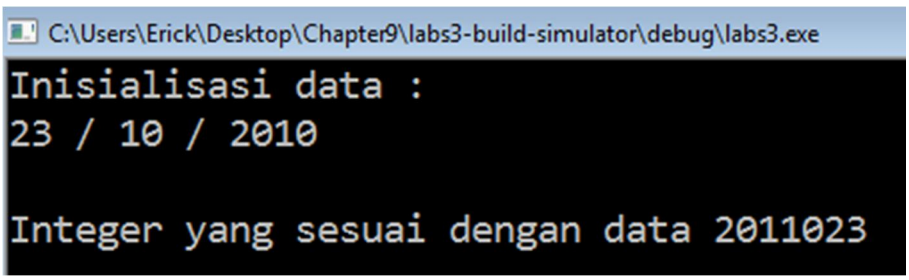
```
operator int()
{
      // implementation
      return intValue;
}
```

Contoh dibawah ini akan menunjukan bagaimana penggunaan conversion operator untuk mengkonversi tipe **`Kalender`** menjadi **`int`**.

Labs.3 Conversion Operator untuk konversi class Kalender ke integer

1. Buka Qt Creator dan buat project Qt Console Application baru dengan nama **Labs3**, kemudian tulis kode berikut.

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;

class Kalender
{
private:
    int _hari;
    int _bulan;
    int _tahun;
public:
    Kalender(int hari,int bulan,int tahun) : _hari(hari), _bulan(bulan),
_tahun(tahun) {}
    operator int()
    {
        return ((_tahun*1000) + (_bulan*100) + _hari);
    }
    void TampilData()
    {
        cout << _hari << " / " << _bulan << " / " << _tahun << endl;
    }
};

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    Kalender objKal(23,10,2010);
    cout << "Inisialisasi data : " << endl;
    objKal.TampilData();
    cout << endl; cout << endl;
    int nData = objKal;
    cout << "Integer yang sesuai dengan data " << nData << endl;
    return a.exec();
}
```



Analisa:

- Pada operator int(), variabel _tahun, _bulan, dan _hari dikalikan dengan bilangan tertentu sehingga menghasilkan kembalian berupa int.
- Statement **`int nData = objKal`** akan menjalankan operator int dan mengembalikan nilai integer dari objek **`Kalender`**.

- Dengan menggunakan operator int akan lebih mudah membandingkan dua objek **`Kalender`**, karena objek tersebut dapat mengembalikan satu nilai integer.

Binary Operator

Operator yang mengoperasikan dua operand disebut dengan binary operator, cara penulisan binary operator sama dengan penulisan oprator yang sebelumnya.

`return_type operator_type (parameter);`

Ada beberapa macam binary operator yang dapat digunakan pada C++, diantaranya :

Operator	Name
+	Addition
+=	Addition/Assigment
-	Substraction
-=	Substraction/Assigment
<	Less Than
>	Greater Than
<=	Less thar or equal to
>=	Greater than or equal to
==	Equal to
!=	Inequality

Contoh program dibawah ini menggunakan operator Addition (+) untuk menambahkan hari pada objek kalender, anda dapat menambahkan beberapa hari kedepan, misal 5 atau 10 hari dari tanggal sekarang.

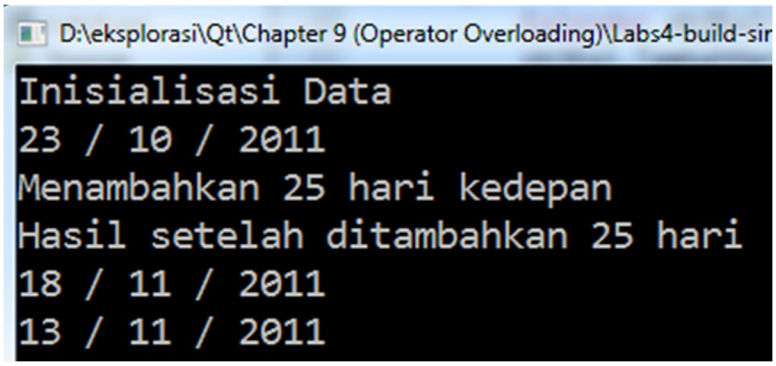
Labs.4 Menggunakan Binary Addition Operator

1. Buka Qt Creator dan buat project Qt Console Application baru dengan nama **Labs3**, kemudian tulis kode berikut.

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
class Kalender
{
private:
    int _hari;
    int _bulan;
    int _tahun;
    void TambahHari(int hari)
    {
        _hari += hari;
        if(_hari>30)
        {
            TambahBulan(_hari/30);
            _hari %= 30;
        }
    }
    void TambahBulan(int bulan)
    {
        _bulan += bulan;
        if(_bulan>12)
```

```
        {
            TambahTahun(_bulan/12);
            _bulan %= 12;
        }
    }
    void TambahTahun(int tahun)
    {
        _tahun += tahun;
    }
public:
    Kalender(int hari, int bulan, int tahun) : _hari(hari), _bulan(bulan),
_tahun(tahun) {}
    Kalender operator + (int hari)
    {
        Kalender objKal(_hari,_bulan,_tahun);
        objKal.TambahHari(hari);
        return objKal;
    }
    void TampilTanggal()
    {
        cout << _hari << " / " << _bulan << " / " << _tahun << endl;
    }
};
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    Kalender objKal(23,10,2011);
    cout << "Inisialisasi Data " << endl;
    objKal.TampilTanggal();
    cout << "Menambahkan 25 hari kedepan " << endl;
    Kalender objKalBaru(objKal + 25);
    cout << "Hasil setelah ditambahkan 25 hari " << endl;
    objKalBaru.TampilTanggal();

    //cara penulisan yang lain
    objKal=objKal+20;
    objKal.TampilTanggal();
    return a.exec();
}
```

2. Kemudian jalankan kode diatas dengan menekan tombol Ctrl+R, outputnya adalah sebagai berikut.



Analisa:

- Dengan menggunakan operator + anda dapat menambahkan hari pada objek Kalender, jumlah hari yang ditambahkan tergantung dari nilai yang diinputkan pada parameter.
- Anda dapat menambah hari dengan menggunakan operator + pada objek Kalender, misal: **`objKal = objKal + 25`** atau dengan membuat objek baru untuk menampung nilai hasil penambahan **`Kalender objKalBaru(objKal+20)`**

Addition-Assignment Operator

Dengan menggunakan Addition-Assignment operator anda dapat menuliskan sintaks **`a += b`**, yang sama artinya dengan **`a = a + b`**. Pada contoh program dibawah ini operator Addition-Assignment akan digunakan untuk menambahkan hari pada objek Kalender.

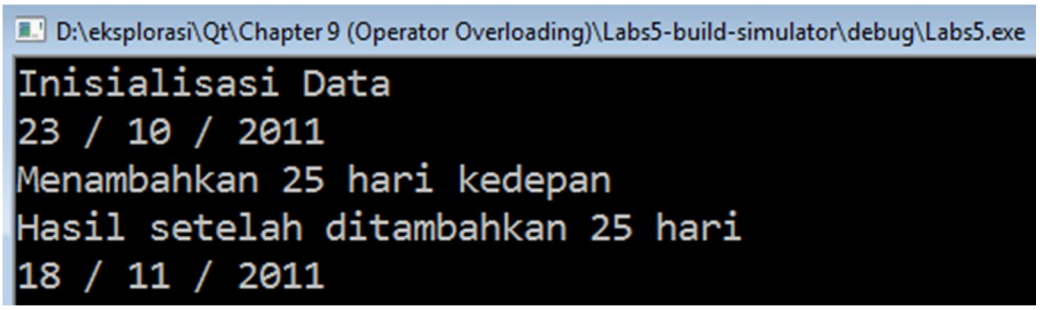
Labs.5 Menggunakan Addition Assigment Operator dan Substraction Assigment Operator

1. Buka Qt Creator dan buat project Qt Console Application baru dengan nama **Labs5**, kemudian tulis kode berikut.

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
class Kalender
{
private:
    int _hari;
    int _bulan;
    int _tahun;
    void TambahHari(int hari)
    {
        _hari += hari;
        if(_hari>30)
        {
            TambahBulan(_hari/30);
            _hari %= 30;
        }
    }
    void TambahBulan(int bulan)
    {
        _bulan += bulan;
        if(_bulan>12)
        {
            TambahTahun(_bulan/12);
            _bulan %= 12;
        }
    }
    void TambahTahun(int tahun)
    {
        _tahun += tahun;
    }
public:
```

```
    Kalender(int hari, int bulan, int tahun) : _hari(hari), _bulan(bulan),
_tahun(tahun) {}
    void operator += (int hari)
    {
        TambahHari(hari);
    }
    void TampilTanggal()
    {
        cout << _hari << " / " << _bulan << " / " << _tahun << endl;
    }
};
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    Kalender objKal(23,10,2011);
    cout << "Inisialisasi Data " << endl;
    objKal.TampilTanggal();
    cout << "Menambahkan 25 hari kedepan " << endl;
    objKal+=25;
    cout << "Hasil setelah ditambahkan 25 hari " << endl;
    objKal.TampilTanggal();
    return a.exec();
}
```

2. Kemudian jalankan kode diatas dengan menekan tombol Ctrl+R, outputnya adalah sebagai berikut.



Analisa

- Pada kode diatas anda dapat menggunakan operator Addition-Assignment pada objek Kalender, misal anda ingin menambahkan 25 hari pada objek Kalender, anda dapat menuliskan kode **`objKal += 25;`**

Comparison Operator

Pada kasus tertentu dimana anda ingin membandingkan dua objek bertipe Kalender ada dapat menggunakan comparison operator.

```
if (objKal1 == objKal2)
{
      // Do something
}
else
```

```
{
      // Do something else
}
```

Anda dapat menggunakan equality operator (==) atau inequality operator (!=). Anda juga dapat membuat lebih dari satu equality atau inequality operator yang mempunyai return value atau parameter yang berbeda, ini disebut dengan overloading operator.

Labs.6 Overloading Comparison Operator (Equality dan Inequality)

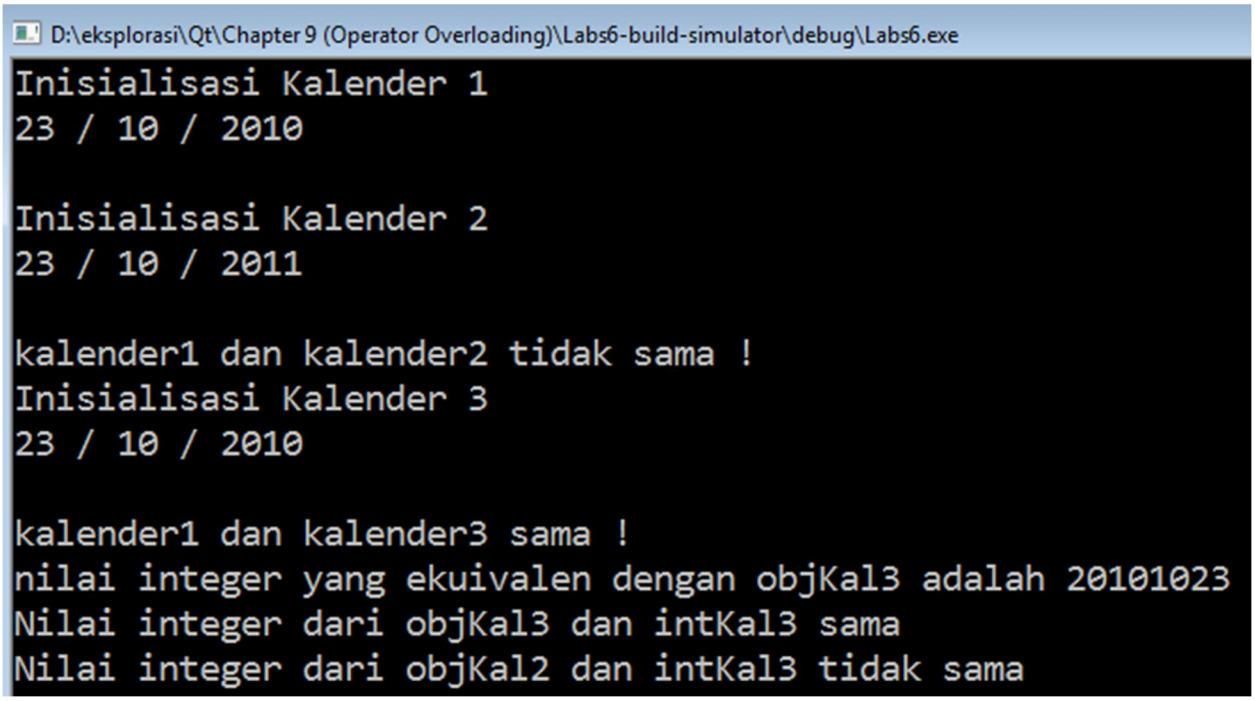
1. Buka Qt Creator dan buat project Qt Console Application baru dengan nama **Labs6**, kemudian tulis kode berikut.

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
class Kalender
{
private:
    int _hari;
    int _bulan;
    int _tahun;
    void TambahHari(int hari);
    void TambahBulan(int bulan);
    void TambahTahun(int tahun);
public:
    Kalender(int hari,int bulan, int tahun) : _hari(hari), _bulan(bulan),
_tahun(tahun) {}
    void TampilTanggal()
    {
        cout << _hari << " / " << _bulan << " / " << _tahun << endl;
    }
    operator int();
    bool operator == (const Kalender &objKal);
    bool operator == (int tglNumber);
    bool operator != (const Kalender &objKal);
    bool operator != (int tglNumber);
};

Kalender::operator int()
{
    return ((_tahun*10000)+(_bulan*100)+_hari);
}

bool Kalender::operator ==(const Kalender &objKal)
{
    return ((objKal._hari==_hari) && (objKal._bulan==_bulan) &&
(objKal._tahun==_tahun));
}

bool Kalender::operator ==(int tglNumber)
{
    return tglNumber == (int)*this;
}
```

```
bool Kalender::operator !=(const Kalender &objKal)
{
    return !(this->operator ==(objKal));
}
bool Kalender::operator !=(int tglNumber)
{
    return !(this->operator ==(tglNumber));
}
void Kalender::TambahHari(int hari)
{
    _hari += hari;
    if(_hari>30)
    {
        TambahBulan(_hari/30);
        _hari %= 30;
    }
}
void Kalender::TambahBulan(int bulan)
{
    _bulan += bulan;
    if(_bulan>12)
    {
        TambahTahun(_bulan/12);
        _bulan %= 12;
    }
}
void Kalender::TambahTahun(int tahun)
{
    _tahun += tahun;
}
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    Kalender objKal1(23,10,2010);
    cout << "Inisialisasi Kalender 1" << endl;
    objKal1.TampilTanggal();
    cout << endl;

    Kalender objKal2(23,10,2011);
    cout << "Inisialisasi Kalender 2" << endl;
    objKal2.TampilTanggal();
    cout << endl;

    //menggunakan operator tidak sama dengan
    if(objKal1 != objKal2)
        cout << "kalender1 dan kalender2 tidak sama !" << endl;
    Kalender objKal3(23,10,2010);
    cout << "Inisialisasi Kalender 3" << endl;
    objKal3.TampilTanggal();
    cout << endl;

    //menggunakan operator sama dengan
    if(objKal1==objKal3)
        cout << "kalender1 dan kalender3 sama !" << endl;
```

```
    int intKal3 = objKal3;
    cout << "nilai integer yang ekuivalen dengan objKal3 adalah " << intKal3 << endl;

    //menggunakan overloading operator sama dengan untuk membandingkan integer
    if(objKal3 == intKal3)
        cout << "Nilai integer dari objKal3 dan intKal3 sama" << endl;

    //menggunakan overloading operator tidak sama dengan untuk membandingkan integer
    if(objKal2 != intKal3)
        cout << "Nilai integer dari objKal2 dan intKal3 tidak sama" << endl;
    return a.exec();
}
```

2. Kemudian jalankan kode diatas dengan menekan tombol Ctrl+R, outputnya adalah sebagai berikut.



Analisa

- Pada kode diatas terdapat 4 comparison operator yang berbeda, walaupun masing-masing ada 2 inequality dan equality operator namun parameternya berbeda ini disebut sebagai overloading operator.
- Pada operator equality (==) yang pertama membandingkan dua objek **`Kalender`** dengan cara membandingkan member variabel **`hari`**, **`bulan`**, **`tahun`** pada masing-masing objek yang dibandingkan, sedangkan operator == yang kedua membandingkan objek **`Kalender`** yang terlebih dahulu sudah dikonversi menjadi int.
- Anda dapat melihat bahwa kedua operator comparison diatas sama-sama dapat membandingkan isi dari 2 objek **`Kalender`**, baik dengan cara membandingkan member variabel maupun membandingkan nilai int (hasil konversi dari objek **`Kalender`**).

Overloading Operator <, >, <=, >=

Seperti pada contoh sebelumnya anda juga dapat menggunakan operator <, >, <=, >= untuk membandingkan objek Kalender. Agar mudah untuk dibandingkan maka objek Kalender dikonversi terlebih dahulu menjadi tipe int.

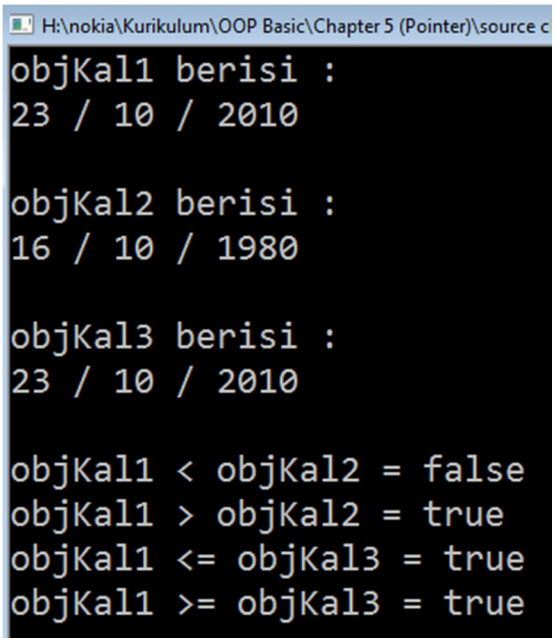
Labs.7 Menggunakan Operator <. >, <=, >=

1. Buka Qt Creator dan buat project Qt Console Application baru dengan nama **Labs7**, kemudian tulis kode berikut.

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
class Kalender
{
private:
    int _hari;
    int _bulan;
    int _tahun;
    void TambahHari(int hari);
    void TambahBulan(int bulan);
    void TambahTahun(int tahun);
public:
    Kalender(int hari,int bulan, int tahun) : _hari(hari), _bulan(bulan),
_tahun(tahun) {}
    void TampilTanggal()
    {
        cout << _hari << " / " << _bulan << " / " << _tahun << endl;
    }
    operator int() const;
    bool operator < (const Kalender &objKal) const;
    bool operator <= (const Kalender &objKal) const;
    bool operator > (const Kalender &objKal) const;
    bool operator >= (const Kalender &objKal) const;
};
Kalender::operator int() const
{
    return ((_tahun*10000)+(_bulan*100)+_hari);
}
bool Kalender::operator <(const Kalender &objKal) const
{
    return (this->operator int() < objKal.operator int());
}
bool Kalender::operator >(const Kalender &objKal) const
{
    return (this->operator int() > objKal.operator int());
}
bool Kalender::operator <=(const Kalender &objKal) const
{
    return (this->operator int() <= objKal.operator int());
}
bool Kalender::operator >=(const Kalender &objKal) const
{
```

```
    return (this->operator int() >= objKal.operator int());
}
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    Kalender objKal1(23,10,2010);
    Kalender objKal2(16,10,1980);
    Kalender objKal3(23,10,2010);

    cout << "objKal1 berisi : " << endl;
    objKal1.TampilTanggal();
    cout << endl;

    cout << "objKal2 berisi : " << endl;
    objKal2.TampilTanggal();
    cout << endl;

    cout << "objKal3 berisi : " << endl;
    objKal3.TampilTanggal();
    cout << endl;

    //menggunakan operator <
    cout << "objKal1 < objKal2 = ";
    cout << ((objKal1 < objKal2) ? "true" : "false") << endl;

    //menggunakan operator >
    cout << "objKal1 > objKal2 = ";
    cout << ((objKal1 > objKal2) ? "true" : "false") << endl;

    //menggunakan operator <=
    cout << "objKal1 <= objKal3 = ";
    cout << ((objKal1 <= objKal3) ? "true" : "false") << endl;

    //menggunakan operator >=
    cout << "objKal1 >= objKal3 = ";
    cout << ((objKal1 >= objKal3) ? "true" : "false") << endl;
    return a.exec();
}
```

2. Kemudian jalankan kode diatas dengan menekan tombol Ctrl+R, outputnya adalah sebagai berikut.



Analisa

- Pada program diatas penggunaan operator <, >, <=, >= digunakan untuk membandingkan dua objek **`Kalender`** yang berbeda.
- Untuk mempermudah membandingkan dua objek **`Kalender`** maka objek **`Kalender`** tersebut dikonversi terlebih dahulu menjadi **`int`**.

Subscript Operator

Subscript operator dapat digunakan jika anda ingin mengakses class seperti ketika anda mengakses array, anda dapat menambahkan operator **[ ]** pada objek yang anda buat untuk mengakses nilai dengan index tertentu dari objek. Contoh dibawah ini akan menjelaskan penggunaan subscript operator untuk membuat array yang dinamis.

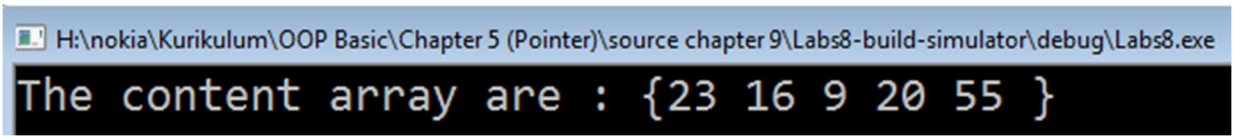
Labs.8 Subscript Operator untuk Dynamic Array

1. Buka Qt Creator dan buat project Qt Console Application baru dengan nama **Labs8**, kemudian tulis kode berikut.

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
class COpArray
{
private:
    int *_myArray;
    int _numElement;
public:
    COpArray(int numElement);
    ~COpArray();
    int& operator[](int index);
};
```

```
int& COpArray::operator [](int index)
{
    return _myArray[index];
}
COpArray::COpArray(int numElement)
{
    _myArray = new int[numElement];
    _numElement = numElement;
}
COpArray::~COpArray()
{
    delete [] _myArray;
}
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    COpArray arrOp(5);
    arrOp[0]=23;
    arrOp[1]=16;
    arrOp[2]=9;
    arrOp[3]=20;
    arrOp[4]=55;
    cout << "The content array are : " << "{";
    for(int i=0;i<5;++i)
    {
        cout << arrOp[i] << " ";
    }
    cout << "}" << endl;
    return a.exec();
}
```

2. Kemudian jalankan kode diatas dengan menekan tombol Ctrl+R, outputnya adalah sebagai berikut.



Analisa

- Dengan menggunakan subscript operator anda dapat mengakses member variable yang bertipe array pada class dengan menggunakan *array-like syntax* (nama objek diikuti dengan tanda []), misal: **`namaObjek[index]`**.

Function operator()

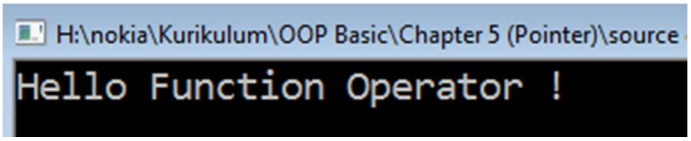
Function operator digunakan jika anda ingin membuat objek bekerja seperti function. Untuk lebih jelasnya penggunaan function operator() anda dapat mencoba program dibawah ini.

Labs.9 Menggunakan operator() untuk membuat function object

1. Buka Qt Creator dan buat project Qt Console Application baru dengan nama **Labs9**, kemudian tulis kode berikut.

```
#include <QtCore/QCoreApplication>
#include <iostream>
using namespace std;
class CTampil
{
public:
    void operator()(string msg) const
    {
        cout << msg << endl;
    }
};
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    CTampil objTampil;
    //penulisan ekuivalen dengan objTampil.operator ()("Hello Function Operator !");
    objTampil("Hello Function Operator !");
    return a.exec();
}
```

2. Kemudian jalankan kode diatas dengan menekan tombol Ctrl+R, outputnya adalah sebagai berikut.



Analisis

- Dengan membuat function operator maka anda dapat menggunakan objek seperti ketika anda menggunakan function.
- Pada program diatas objek **`objTampil`** dapat dipanggil seperti function.