

Pemrograman Lanjut



Hanya dipergunakan di lingkungan Jurusan Teknik Informatika UB

LABORATORIUM PEMBELAJARAN

DAFTAR PENYUSUN

- Indriati, S.T., M.Kom
- Tri Afirianto, S.T.,M.T
- Nyoman Wira Prasetya
- Mochammad Hannast Hanafi Ichsan, S.ST., M.T
- Nurudin Santoso, S.T., M.T
- Budi darma Setiawan, S.Kom., M.Cs

Modul 1 CLASS dan OBJECT

1.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut (misalkan):

- 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- 60 menit untuk penyampaian materi
- 45 menit untuk pengerjaan jurnal, tes akhir atau tugas
- 50 menit

1.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

- memahami konsep OOP (Object Oriented Programming) dari bahasa C++
- membuat class dan object sebagai bentuk implementasi dari bab ini

1.3 Alat & Bahan

- IDE Bahasa C++

1.4 Dasar Teori

A. Pengenalan Object Oriented Programming (OOP)

OOP adalah sebuah konsep/cara pemrograman dengan menggunakan objek sebagai elemen dasar dari program. Jika kita memperhatikan dunia nyata, kita dapat menemukan beragam objek disekitar kita seperti mobil, singa, manusia dan seterusnya. Objek yang dimaksud di sini, dikarakterisasi oleh atribut dan tingkah lakunya.

Contohnya, objek sebuah mobil mempunyai atribut no plat, warna, manufaktur, . Objek Mobil juga mempunyai tingkah laku berbelok, mengerem dan berakselerasi. Dengan cara yang sama pula kita dapat mendefinisikan perbedaan sifat dan tingkah laku dari objek singa. Coba perhatikan tabel dibawah ini sebagai contoh perbandingan :

Objek	Atribut	Tingkah Laku
Mobil	No plat Warna Manufaktur Kecepatan	Berbelok Mengerem Mempercepat
Singa	Mata Kaki Taring	Tidur Berlari Memangsa

Dengan deskripsi ini, objek pada dunia nyata dapat secara mudah diasumsikan sebagai objek perangkat lunak menggunakan atribut sebagai data dan tingkah laku sebagai method. Data dan method dapat digunakan dalam pemrograman game atau perangkat lunak interaktif untuk membuat simulasi objek pada dunia nyata. Contohnya adalah perangkat lunak objek mobil dalam permainan balap mobil ataupun singa dalam perangkat lunak pendidikan interaktif untuk anak-anak.

B. Class

Class adalah struktur dasar dari OOP. Class inilah yang nantinya digunakan sebagai *template* atau cetakan dari sebuah objek. Pembentukan objek dilakukan dengan menggunakan class. Class terdiri dari 2 dua komponen yang disebut dengan field (menggambarkan atribut/properti) dan method (menggambarkan tingkah laku). Field merupakan tipe data yang didefinisikan oleh class, sementara method merupakan operasi. Sedangkan objek adalah

sebuah instance dari class. Untuk dapat membedakan class dan objek, dapat dilihat contoh pada tabel dibawah ini :

Class Mobil		Objek Mobil A	Objek Mobil B
Variabel instance	Nomor plat	N 7221 WZ	N 2010 KT
	Warna	Biru	Merah
	Manufaktur	Mitsubishi	Toyota
	Kecepatan	50 km/h	100 km/h

Ketika diinisialisasi, setiap objek mendapat satu set variabel yang baru. Bagaimanapun, implementasi dari method dibagi diantara objek pada class yang sama. Class menyediakan keuntungan dari *reusability* artinya programmer perangkat lunak dapat menggunakan sebuah class beberapa kali untuk membuat objek.

Untuk mendefinisikan class dapat dituliskan sebagai berikut

```
class <name> {
    <attribut declaration>
    <constructor declaration>
    <method declaration>
};
```

Contoh 1:

Method didefinisikan langsung didalam class

```
class Mobil{
    private:
        string nomorPlat;
        string warna;
        string manufaktur;
        int kecepatan;
    public:
        void berbelok();
        void mengerem();
        void mempercepat();
};
```

Instansiasi

Instansiasi adalah proses untuk membuat objek dari sebuah class. Membuat instan Objek dari sebuah class dilakukan dengan menggunakan 2 cara. Contohnya pada suatu kasus kita memiliki Class bernama mobil dan kita ingin menginstan objek dari class Mobil pada class mainMobil dan kita beri nama xenia dan avanza.

```
void main() {
    Mobil xenia; //cara yang pertama
    Mobil *avanza; //cara yang kedua
}
}
```

1.5 Prosedur Praktikum

A. Class

Ketikkan program di bawah ini

Mobil.cpp

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  class Mobil{
5      private:
6          string nomorPlat;
7          string warna;
8          string manufaktur;
9          int kecepatan;
10     public:
11         void setNomorPlat(string n){
12             this->nomorPlat=n;
13         }
14         void setWarna(string w){
15             this->warna=w;
16         }
17         void setManufaktur(string m){
18             this->manufaktur=m;
19         }
20         void setKecepatan(int k){
21             this->kecepatan=k;
22         }
23         void display(){
24             cout<<"Mobil anda berplat nomor " <<nomorPlat<< "
25 dan berwarna "<< warna<<endl;
26             cout<<"Diproduksi oleh perusahaan
27 "<<manufaktur<<endl;
28             cout<<"Mobil anda mampu menempuh kecepatan
29 "<<kecepatan<<"km/h"<<endl;
30         }
31 };
32 int main(){
33     cout<<"===== "<<endl;
34     Mobil xenia;//cara 1 menginstansiasi objek
35     xenia.setNomorPlat("N 1610 BG");
36     xenia.setManufaktur("Daihatsu");
37     xenia.setWarna("putih");
38     xenia.setKecepatan(100);
39     xenia.display();
40     cout<<"===== "<<endl;
41     Mobil *avanza=new Mobil();
42     avanza->setNomorPlat("N 1620 GG");
43     avanza->setManufaktur("Toyota");
44     avanza->setWarna("hitam");
45     avanza->setKecepatan(100);
46     avanza->display();
47     cout<<"===== "<<endl;
48     cout<<"mengubah warna mobil xenia"<<endl;
49     xenia.setWarna("merah");
50     xenia.display();
}
```

1.6 Hasil Percobaan

Tulislah hasil percobaan dari program Mobil.cpp

1.7 Analisis Hasil

Pertanyaan

1. Apakah yang disebut dengan variabel instance dan lokal variabel? Jelaskan perbedaanya!
.....
.....
2. Lakukan percobaan diatas dan benahi jika menemukan kesalahan!
.....
.....
3. Rubah kode pada Mobil diatas menjadi proses meminta masukan dari user dan buat menjadi interaktif!
.....
.....
4. Tambahkan method pada class mobil bernama setWaktu yang berparameter double, yang kemudian disimpan pada variabel waktu! (Ketentuannya adalah user harus menginputkan dalam satuan jam)
.....
.....
5. Tambahkan method bernama ubahSekon mempunyai parameter bertipe double dan hanya dapat dipanggil pada class mobil. Method ini memiliki fungsi untuk merubah masukan user yaitu jam menjadi sekon. Method tersebut di panggil pada method setWaktu dengan nilai parameter adalah nilai dari variabel parameter method setWaktu!
.....

-
6. Tambahkan method pada class mobil dan hanya dapat dipanggil pada class mobil bernama `ubahKecepatan` yang mempunyai fungsi untuk merubah format kecepatan yang awalnya km/h menjadi m/s. Dipanggil di method `setKecepatan`!

-
7. Tambahkan method pada class mobil bernama `hitungJarak` yang mempunyai aksi untuk menghitung jarak yang dapat di tempuh oleh mobil dengan rumus $\text{jarak} = \text{kecepatan} * \text{waktu}$!

-
8. Tambahkan informasi jarak yang dapat ditempuh pada method `displayMessage` kemudian rubah satuannya yang awalnya m (meter) menjadi km (kilometer)!
-

1.8 Kesimpulan

1.9 Latihan

1. Mahasiswa A ingin menulis pada sebuah buku tulis yang ingin dia miliki, isi lembar buku tersebut adalah 50 lembar. Setiap harinya ia menulis sebanyak 100 kata perhari yang cukup untuk 1/2 halaman buku. Buatlah rumus untuk menghitung berapa lama ia menghabiskan 1 buku tersebut serta identifikasilah objek, dan karakteristiknya kemudian implementasikan dalam bentuk class.
-

1.10 Tugas

Sebelum mengerjakan soal di bawah ini tentukan dahulu objek, attribut, behaviour dan class. Buatlah sebuah sistem sederhana yang menyerupai Sistem Informasi Akademik Mahasiswa (SIAM), dengan ketentuan user menginputkan Nama, Nim, IP serta jurusan. Selain itu mahasiswa juga dapat memasukkan kode Mata kuliah, Nama Mata kuliah dan jumlah sks mata kuliah tersebut. Jumlah sks yang di ambil harus sesuai dengan IP yang di dapat pada semester lalu. Buat skenario dengan banyak mahasiswa minimal 3 orang.

Modul 2 : Static Method dan Overloading

2.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut (misalkan):

- e. 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- f. 60 menit untuk penyampaian materi
- g. 45 menit untuk pengerjaan jurnal, tes akhir atau tugas
- h. 50 menit **Pengayaan**

(ket. Pengayaan wajib ada, waktu harus dilakukan selama 170 menit, kombinasi dari 170 menit bisa berbeda tiap bab-nya namun komponennya tidak boleh dikurangi atau ke-empat komponen tersebut wajib ada)

2.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mampu memahami konsep static method yang ada di C++
2. Mampu membedakan perbedaan method yang menggunakan kata kunci static atau tidak
3. Mampu memahami dan mengimplementasikan method overloading

2.3 Alat & Bahan

2. Dev C++
3. Atau dapat menggunakan compiler C++ lain missal Codeblock, Notepad++ dsb.

2.4 Dasar Teori

A. Static Method

Pada saat tertentu, terkadang dibutuhkan variabel yang di-share oleh semua *instance* kelas. Sebagai contoh, variabel yang digunakan sebagai basis untuk komunikasi antar *instance* atau menghitung jumlah *instance* yang telah dibuat. Untuk melakukan hal ini, tandai sebuah variabel dengan *keyword static*. Variabel tersebut seringkali disebut sebagai variabel kelas untuk membedakannya dari member variable atau variabel *instance* yang tidak di-share oleh semua instance. Perhatikan contoh kode program berikut.

Counter.cpp	
1	class Count1
2	{
3	private:
4	int serialNumber ;
5	static int counter ;
6	public:
7	Count1()
8	{
9	counter++;
10	serialNumber = counter;
11	}
12	};

Dalam contoh di atas, diberikan *serial number* yang unik untuk setiap obyek yang dibuat, dimulai dengan angka 1 untuk obyek pertama, 2 untuk obyek kedua dan seterusnya. Variable *counter* di-share oleh semua instance. Ketika membuat sebuah obyek *Count*, konstruktor akan dipanggil. Pemanggilan konstruktor ini akan meng-increment nilai dari variabel *counter* dan nilai ini diberikan kepada variabel *serialNumber*.

Variabel *static* memiliki konsep yang serupa dengan variabel global di dalam bahasa pemrograman lainnya. Bahasa Java tidak mengenal variabel global, tetapi variabel *static* adalah sebuah variabel yang dapat diakses oleh setiap instance dari sebuah kelas. Jika variabel *static* di atas (*counter*) tidak dideklarasikan *private*, maka variabel tersebut dapat diakses dari luar kelas. Untuk mengkasusnya tidak diperlukan instance dari kelasnya, tetapi menggunakan nama kelasnya sebagaimana ditunjukkan pada kode program berikut ini.

Counter.cpp	
1	class OtherClass
2	{
3	public:
4	virtual void incrementNumber()
5	{
6	Count1::counter++;
7	}
8	};

Terkadang perlu juga mengakses kode program dimana *instance* dari sebuah kelas tidak dimiliki. Hal tersebut dapat dilakukan dengan menggunakan method yang dideklarasikan *static*. method demikian seringkali disebut sebagai method kelas sebagaimana ditunjukkan pada kode program berikut.

Counter.cpp	
1	class Count2
2	{
3	private:
4	int serialNumber ;
5	static int counter;
6	public:
7	static int getTotalCount()
8	{
9	return counter;
10	}
11	Count2()
12	{
13	counter++;
14	serialNumber = counter;
15	}
16	};

Untuk menggunakan method yang dideklarasikan *static*, tidak digunakan referensi obyeknya melainkan nama kelas seperti ditunjukkan pada kode program berikut ini :

Counter.cpp	
1	class TestCounter
2	{
3	static void main(std::vector<std::wstring> &args)
4	{
5	std::wcout << L"number of counter is " <<
6	Count2::getTotalCount() << std::endl;
7	Count2 *count1 = new Count2();
8	cout << L"number of counter is " + Count2::getTotalCount();
9	}

```
10 };
11
```

Keluaran dari program TestCounter di atas adalah :

Number of counter is 0

Number of counter is 1

Karena method *static* dipanggil tanpa menggunakan variabel *instance* dari kelas yang bersangkutan, maka tidak dapat diterapkan variabel referensi *this*. Akibatnya, method *static* tidak dapat mengakses variabel selain daripada variabel lokal, atribut *static*, dan parameternya. Apabila berusaha mengakses atribut non-*static* dari method *static*, hal itu akan menyebabkan compiler error.

Atribut non-*static* terkait dengan satu instance tertentu dan ia hanya bisa diakses melalui referensi instance.

```
Counter.cpp
1 class Count3
2 {
3 private:
4     int serialNumber;
5     static int counter;
6 public:
7     static int getSerialNumber()
8     {
9         return serialNumber; //COMPILER ERROR
10    }
11};
```

Hal yang harus diperhatikan ketika menggunakan method static antara lain :

1. Method *static* tidak dapat di-override, tetapi dapat disembunyikan.
2. Method main() merupakan sebuah method static karena JVM tidak perlu membuat sebuah instance dari kelas yang bersangkutan ketika method main() dieksekusi. Jika mempunyai data member, hendaknya dibuatkan sebuah obyek untuk mengaksesnya.

B. Overloading Method

Overloading adalah pemakaian beberapa methods ataupun properties dengan nama yang sama, tetapi memiliki daftar parameter/argument yang berbeda. Perbedaan yang dimaksud adalah beda jumlah parameter, beda tipe data, atau beda keduanya (jumlah parameter dan tipe data). Methods ataupun properties yang hanya beda return value (nilai balik) tidak bisa dikatakan sebagai overloading.

Keyword yang dipakai adalah *Overloads*. Jika dalam satu class, keyword ini bersifat optional artinya kita tidak harus menambahkan keyword *Overloads* ketika mendefinisikan beberapa methods ataupun properties yang menerapkan konsep overloading. Apabila kita tetap ingin memakainya, maka keyword *Overloads* harus digunakan di semua methods ataupun properties tadi. Apabila beda class, keyword *Overloads* ini harus digunakan.

Method overloading digunakan untuk membuat beberapa fungsi (method) dengan nama yang SAMA dan mengerjakan tugas yang mirip. Tujuannya, agar programmer tidak kesulitan dalam mengingat sebuah fungsi yang tugasnya mirip. Misalnya untuk membuat fungsi perkalian. Ada perkalian yang membutuhkan dua argumen dan ada perkalian yang membutuhkan tiga argumen (tugasnya mirip, yaitu sama-sama mengalikan argumen, hanya jumlah argumen yang berbeda). Kalau dibuat fungsi dengan nama yang berbeda,

kemungkinan programmer akan repot. Contoh, untuk fungsi perkalian dengan dua argumen, namanya adalah kali1(bil1, bil2). Sedangkan untuk perkalian dengan tiga argumen namanya adalah kali2(bil1, bil2, bil3). Repot kan? Gimana kalo ada sepuluh perkalian? Bandingkan bila namanya kali(bil1, bil2) dan kali(bil1, bil2, bil3). Kita cuma perlu mengingat jumlah argumen dari fungsi kali.

Sehingga, untuk membuat method overloading diperlukan setidaknya satu dari tiga syarat di bawah ini:

mempunyai jumlah argumen berbeda.

mempunyai tipe data argumen yang berbeda.

mempunyai urutan argumen yang berbeda.

Contoh #1 (jumlah argumen berbeda):

Method perkalian dengan argumen	
1	// method kali dengan dua argumen (bil1 dan bil2)
2	int kali(int bil1, int bil2) {
3	return bil1*bil2
4	}
5	
6	// method kali dengan tiga argumen (bil1, bil2 dan bil3)
7	int kali(int bil1, int bil2, int bil3) {
8	
9	return bil1*bil2*bil3;
10	}

Method kali pada contoh di atas VALID karena mempunyai jumlah argumen yang berbeda.

Contoh #2 (mempunyai tipe data argumen yang berbeda):

Method perkalian dengan argumen data yang berbeda	
1	// method tampilkanNilai mempunyai argumen bertipe char
2	void tampilkanNilai(char nilai) {
3	cout << "Nilai (dalam huruf): " << nilai;
4	}
5	
6	// method tampilkanNilai mempunyai argumen bertipe int
7	void tampilkanNilai(int nilai) {
8	cout << "Nilai (dalam angka): " << nilai;
9	}

Contoh method tampilkanNilai diatas VALID karena mempunyai tipe data argumen yang berbeda walaupun memiliki jumlah argumen yang sama.

Contoh #3 (urutan argumen berbeda):

Method perkalian dengan argumen	
1	// method kali dengan tipe data bil1 adalah int
2	// dan bil2 adalah double
3	double kali(int bil1, double bil2) {
4	return bil1*bil2;
5	}
6	
7	// method kali dengan tipe data bil1 adalah double
8	// dan bil2 adalah int

9	double kali(double bil1, int bil2) {
10	return bil1*bil2;
11	}

Method kali di atas VALID karena urutan argumen berbeda. Pada method kali yang pertama bil1 bertipe data int kemudian diikuti oleh bil2 dengan tipe data double. Sedangkan pada method kali yang kedua, bil1 bertipe data double diikuti oleh bil2 yang bertipe data int.

Contoh yang salah:

Method perkalian dengan argumen data yang berbeda	
---	--

1	// return type: int
2	int hitungUmur(int umur)
3	{
4	return umur + 2;
5	}
6	
7	// return type: void
8	void hitungUmur(int umur)
9	{
10	cout<<"umur siswa setelah lulus kuliah= "<<umur+2;
11	}

SYNTAX ERROR. Method tidak dapat di-overload karena mempunyai jumlah argumen yang sama dengan tipe data yang sama.

2.5 Prosedur Praktikum

Jalankan kode berikut

A. Static

Ketikkan program dibawah ini

Aritmatika.cpp	
----------------	--

1	class Aritmatika
2	{public:
3	virtual void hitungPenjumlahan(int a, int b);
4	static void hitungPerkalian(int a, int b);
5	static void hitungPengurangan(int a, int b);
6	};
7	
8	void Aritmatika::hitungPenjumlahan(int a, int b){
9	int nilai = a + b;
10	cout << "nilai penjumlahan adalah : " << nilai << endl;
11	}
12	void Aritmatika::hitungPerkalian(int a, int b){
13	int nilai = a*b;
14	cout << "nilai perkalian adalah : " << nilai << endl;
15	}
16	void Aritmatika::hitungPengurangan(int a, int b){
17	int nilai = a - b;
18	cout << "nilai pengurangan adalah : " << nilai << endl;
19	}

Selanjutnya adalah membuat class main dari class Aritmatika bernama MainAritmatika dan di beri instan objek untuk memanggil method static dan non static.

MainAritmatika.cpp

1	class MainAritmatika{
2	static void main(vector<wstring> &args);
3	};
4	
5	void MainAritmatika(vector<wstring> &args){
6	Scanner *in_Renamed = new Scanner(System::in);
7	cout << "masukkan nilai 1 : ";
8	int nil1 = in_Renamed->nextInt();
9	cout << "masukkan nilai 2 : ";
10	int nil2 = in_Renamed->nextInt();
11	//memanggil method static
12	Aritmatika::hitungPengurangan(nil1, nil2);
13	cout << "masukkan nilai 1 : ";
14	nil1 = in_Renamed->nextInt();
15	cout << "masukkan nilai 2 : ";
16	nil2 = in_Renamed->nextInt();
17	//memanggil method static
18	Aritmatika::hitungPerkalian(nil1, nil2);
19	cout << "masukkan nilai 1 : ";
20	int value1 = in_Renamed->nextInt();
21	cout << "masukkan nilai 2 : ";
22	int value2 = in_Renamed->nextInt();
23	//memanggil method NONstatic harus melalui objek
24	Aritmatika *a = new Aritmatika();
25	a->hitungPenjumlahan(value1, value2);
26	}

B. Overloading

Asas

MainAritmatika.cpp	
1	void MainAritmatika(vector<wstring> &args)
2	{
3	Scanner *in_Renamed = new Scanner(System::in);
4	cout << "masukkan nilai 1 : ";
5	int nil1 = in_Renamed->nextInt();
6	cout << "masukkan nilai 2 : ";
7	int nil2 = in_Renamed->nextInt();
8	//memanggil method static
9	Aritmatika::hitungPengurangan(nil1, nil2);
10	cout << "masukkan nilai 1 : ";
11	nil1 = in_Renamed->nextInt();
12	cout << "masukkan nilai 2 : ";
13	nil2 = in_Renamed->nextInt();
14	//memanggil method static
15	Aritmatika::hitungPerkalian(nil1, nil2);
16	cout << "masukkan nilai 1 : ";
17	int value1 = in_Renamed->nextInt();
18	cout << "masukkan nilai 2 : ";
19	int value2 = in_Renamed->nextInt();
20	//memanggil method NONstatic harus melalui objek
21	Aritmatika *a = new Aritmatika();

22	a->hitungPenjumlahan(value1, value2);
23	}

2.6 Hasil Percobaan

1. Tuliskan hasil dari percobaan pertama di atas! Perbaiki jika ada kode yang salah!
2. Tuliskan hasil dari percobaan kedua di atas! Perbaiki jika ada kode yang salah!

2.7 Analisis Hasil

A. Static Method

Pertanyaan

1. Apakah yang disebut dengan static variabel? Dan apa fungsi dari static variabel serta kapan kita dapat menggunakan static variabel?

.....

2. Mengapa pada main method harus dituliskan static? Jelaskan jawaban anda beserta dengan alasan!

.....

B. Overloading Method

Pertanyaan

1. Apakah yang disebut dengan overloading method? Dan apa fungsi dari overloading method serta kapan kita dapat menggunakan overloading method?

.....

2. Jelaskan apa saja faktor pembeda pada overloading method!

.....

2.8 Kesimpulan

1. Tuliskan kesimpulan dari percobaan pertama di atas!
2. Tuliskan kesimpulan dari percobaan kedua di atas!

2.9 Latihan

A. Static Method

1. Jika pada tubuh method hitungPenjumlahan ditambahkan syntax hitungPerkalian(a,b) apa yang terjadi? Jelaskan?

-
-
2. Jika pada tubuh method `hitungPerkalian` ditambahkan syntax `hitungPenjumlahan(a,b)` apa yang terjadi? Jelaskan?

-
-
3. Tambahkan method non static dengan nilai balikan double untuk menghitung pembagian dengan parameter String `nil` dan String `nil2`, dan panggil method tersebut pada method `main`!

.....

.....

B. Overloading Method

1. Jika pada baris 7, pada parameter `double value` dan `double value2` di hapus dan di ganti menjadi `int a` dan `int b` apa yang terjadi? Jelaskan!

-
-
2. Ubah method pada baris ketujuh menjadi method bertipe `void`, dan lakukan juga perubahan `main method`.

-
-
3. Tambahkan method non static bertipe `void` bernama `HitungLuas` yang mempunyai parameter String bernama `value1` dan String bernama `value2`! Kemudian panggil method tersebut pada `main method`!

-
-
4. Tambahkan method non static bernama `cek Kondisi` yang mempunyai parameter bertipe `double` bernama `value`. Method ini di gunakan untuk mengecek nilai `value` yang merupakan nilai dari luas sebuah bidang. Jika nilai `value` bernilai lebih dari sama dengan 60 dan kurang dari sama dengan 100 maka akan mencetak "Nilai luas bidang termasuk kategori Besar", jika nilai `value` bernilai lebih dari sama dengan 101 maka akan mencetak "Nilai luas bidang termasuk kategori sangat besar", dan jika selain kondisi di atas akan mencetak "Nilai luas bidang termasuk kategori kecil". Kemudian panggil method ini di dalam method `main`!

2.10 Tugas

Tugas Pengayaan Praktikum

1. Soal 1

Susun program dengan menggunakan overloading function dengan ketentuan :

Terdapat method bernama `overloadingMeth` berparameter `String` dan `integer`, dimana method tersebut mempunyai fungsi untuk merubah input teks menjadi bilangan dan input bilangan menjadi teks

Misal :

Input : `overloadingMeth(71)`

Output : tujuh puluh satu

Input : `overloadingMeth(tiga puluh lima)`

Output : 35

Range untuk input parameter adalah 0-100

2. Soal 2

Buatlah program dengan menggunakan class untuk menghitung penjumlahan, pengurangan, perkalian dan pembagian. Method penjumlahan dan pengurangan menggunakan static method sedangkan sisanya menggunakan method non static. Tambahkan method bertipe non static bernama `Sederhana` untuk menyederhanakan sebuah pecahan.

..... *

Modul 3 : Constructor dan Instance Method

3.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut.

- i. 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- j. 30 menit untuk penyampaian materi
- k. 65 menit untuk pengerjaan tugas / Studi Kasus
- l. 60 menit **Pengayaan**

3.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

4. Mampu mendeklarasikan constructor, membuat default konstruktor, dan overloading constructor dari class yang sudah mereka buat
5. Mampu membuat instance method pada class yang telah dibuat

3.3 Alat & Bahan

4. Komputer
5. DevCpp

3.4 Dasar Teori

3.4.1 Constructor

Constructor sangatlah penting pada pembentukan sebuah object. Constructor adalah method yang mana seluruh inisialisasi object ditempatkan. Saat object melakukan instantiate terhadap main class atau class lain, sebenarnya langkah tersebut telah memanggil sebuah constructor pada sebuah class yang telah di-instance object-nya.

Berikut ini merupakan property dari constructor:

1. Constructor memiliki nama yang sama dengan class
2. Constructor tidak memiliki return value, meskipun void
3. Constructor tidak dapat dipanggil secara langsung, namun harus dipanggil dengan menggunakan operator new

Untuk mendeklarasikan sebuah constructor dapat menggunakan sintaks berikut:

```
class class_name{
    access_specifier:
        class_name(parameter){statement}    // constructor
};
```

Contoh: constructor dari class Siswa:

```
class Siswa{
    public:
        Siswa(){                // default constructor
            //statement
        }
};
```

3.4.1.1 Default Constructor

Setiap class memiliki default constructor. Sebuah default constructor adalah constructor yang tidak memiliki parameter apapun. Jika di dalam class tidak didefinisikan constructor apapun, maka sebuah default constructor akan dibentuk secara implisit oleh C++.

Sebagai contoh, pada class Siswa memiliki bentuk default constructor berikut.

```
class Siswa{
```

```

        // secara default memiliki
        //default constructor seperti di bawah
    };

    class Siswa{
    public:
        Siswa(){                // default constructor
            //statement
        }
    };

```

3.4.1.2 Overloading Constructor

Tidak hanya method saja yang memiliki sifat overloading, constructor juga memiliki sifat tersebut. Sama halnya dengan overloading method, overloading constructor adalah constructor yang memiliki nama sama namun memiliki jumlah atau tipe parameter yang berbeda. Contoh overloading constructor adalah sebagai berikut.

```

class Siswa{
    private:
        string name, address;
        double mathGrade, englishGrade;
    public:
        Siswa(){                // default constructor
            //statement
        }
        Siswa(string temp){      // overloading constructor I
            this->name = temp;
        }
        Siswa(string name, string address){ // II
            this->name = name;
            this->address = address;
        }
        Siswa(double mGrade, double eGrade){ // III
            mathGrade = mGrade;
            englishGrade = eGrade;
        }
};

```

3.4.1.3 Menggunakan Constructor

Untuk menggunakan constructor yang telah dibuat pada overloading constructor dapat dilakukan dengan menggunakan object biasa maupun object pointer seperti langkah-langkah berikut.

```

int main(){
    // membuat 5 object
    Siswa s1;                // menggunakan default constructor
    Siswa s2();              // menggunakan default constructor
    Siswa s3("Anna");        // menggunakan overload constructor I

    // menggunakan overload constructor II, dengan object pointer
    Siswa *s4 = new Siswa("Beauty", "Malang");
    // menggunakan overload constructor III, dengan object pointer
    Siswa *s5 = new Siswa(80.5, 85.0);

    return 0;
}

```

3.4.2 Instance Method

Sebuah class juga memiliki method yang dikaitkan dengan instance tertentu. Sewaktu method instance dipanggil, method tersebut akan mengakses data yang terdapat pada instance yang

dikaitkannya. Untuk lebih jelasnya dapat dilihat pada pelaksanaan percobaan bagian instance method.

3.5 Prosedur Praktikum

A. Percobaan Pertama: Constructor

1. Buatlah projek di DevCpp dengan nama Constructor.
2. Buatlah class Siswa pada projek yang telah dibuat, dan tambahkan atribut beserta method seperti kode di bawah ini.

```
1  class Siswa {
2      private:
3          string nama;
4          string alamat;
5          int umur;
6          double nilaiMatematika;
7          double nilaiInggris;
8          double nilaiIPA;
9          double rataRata;
10     public:
11         siswa();
12         Siswa(string nama, string alamat, int umur);
13         void setNama(string nama);
14         void setAlamat(string alamat);
15         void setUmur(int umur);
16         void setMatematika(int matematika);
17         void setInggris(int inggris);
18         void setIPA(int ipa);
19         double getRataRata();
20         void cetak();
21     }
```

3. Tambahkan implementasi dari masing-masing method di bawah deklarasi class Siswa.

```
1  Siswa::Siswa() {
2      nama = "";
3      alamat = "";
4      umur = 0;
5  }
6  Siswa::Siswa(string nama, string alamat, int umur) {
7      nama = nama;
8      alamat = alamat;
9      umur = umur;
10 }
11 void Siswa::setNama(string nama) {
12     this->nama = nama;
13 }
14 void Siswa::setAlamat(string alamat) {
15     this->alamat = alamat;
16 }
17 void Siswa::setUmur(int umur) {
18     this->umur = umur;
19 }
20 void Siswa::setMatematika(int matematika) {
21     nilaiMatematika = matematika;
22 }
23 void Siswa::setInggris(int inggris) {
24     nilaiInggris = inggris;
25 }
26 void Siswa::setIPA(int ipa) {
```

27	nilaiIPA = ipa;
28	}
29	double Siswa::getRataRata() {
30	return (nilaiMatematika + nilaiInggris + nilaiIPA)/3;
31	}
32	void Siswa::cetak() {
33	cout << "Siswa dengan nama " << nama << endl;
34	cout << "beralamat di " << alamat << endl;
35	cout << "berumur " << umur << endl;
36	cout << "mempunyai nilai rata-rata " << getRataRata() << endl;
37	}

4. Tambahkan method main untuk dijalankan pertama kali.

1	int main() {
2	Siswa anna = new Siswa();
3	anna.setNama("Anna");
4	anna.setAlamat("Malang");
5	anna.setUmur(20);
6	anna.setMatematika(100);
7	anna.setInggris(80);
8	anna.setIPA(89);
9	anna.cetak();
10	
11	// menggunakan constructor lain
12	cout << "=====
13	Siswa *chris = new Siswa("Chris", "Kediri", 21);
14	chris->setMatematika(70);
15	chris->setInggris(60);
16	chris->setMatematika(70);
17	chris->cetak();
18	
19	// siswa atas nama Chris diubah informasinya melalui
20	// method
21	cout << "=====
22	chris->setAlamat("Surabaya");
23	chris->setUmur(22);
24	chris->cetak();
25	}

5. Lakukan kompilasi dan tuliskan hasil percobaan, analisis hasil, dan kesimpulannya.

B. Percobaan Kedua: Instance Method

1. Buatlah projek di DevCpp dengan nama Instance Method.
2. Buatlah class Rasional pada projek yang telah dibuat, dan tambahkan atribut beserta method seperti kode di bawah ini.

1	class Rasional {
2	private:
3	int pembilang, penyebut;
4	public:
5	Rasional();
6	Rasional(int pembilang, int penyebut);
7	boolean isRasional();
8	void sederhana();
9	double cast();
10	void lebihDari(Rasional r);
11	void negasi();
12	void unaryPlus(Rasional r);
13	void cetak();
14	}

3. Tambahkan implementasi dari masing-masing method di bawah deklarasi class **Rasional**.

```
1  Rational::Rasional() {
2      pembilang = 0;
3      penyebut = 0;
4  }
5  Rational::Rasional(int pembilang, int penyebut) {
6      this->pembilang = pembilang;
7      this->penyebut = penyebut;
8  }
9  // mengecek suatu bilangan merupakan rasional atau bukan
10 boolean Rational::isRasional() {
11     return (penyebut != 0);
12 }
13 // menyederhanakan bilangan rasional
14 void Rational::sederhana() {
15     int temp, a, b;
16     if (penyebut == 0) {
17         return;
18     }
19     a = (pembilang < penyebut)? penyebut : pembilang;
20     b = (pembilang < penyebut)? pembilang : penyebut;
21     while (b != 0) {
22         temp = a % b;
23         a = b;
24         b = temp;
25     }
26     pembilang /= a;
27     penyebut /= a;
28 }
29 double Rational::cast() {
30     return (penyebut == 0.0) ? 0.0 :
31         static_cast<double>(pembilang) / static_cast<double>(penyebut);
32 }
33 // operator >
34 void Rational::lebihDari(Rasional r) {
35     return (pembilang * r.penyebut) > (penyebut * r.pembilang);
36 }
37 // operator A = -A
38 void Rational::negasi() {
39     pembilang = - pembilang;
40 }
41 // operator unary +=
42 void Rational::unaryPlus(Rasional r) {
43     pembilang = (pembilang * r.penyebut) + (penyebut * r.pembilang);
44     penyebut *= r.penyebut;
45 }
46 void Rational::cetak() {
47     cout << pembilang << "/" << penyebut << endl;
48 }
```

4. Tambahkan method main untuk dijalankan pertama kali.

```
1  int main() {
2      Rational r1(2, 4);
3      Rational r2(5, 15);
4
5      cout << "r1.isRasional: " << r1.isRasional() << endl;
```

6	<code>cout << "r2.isRasional: " << r2.isRasional() << endl;</code>
7	<code>cout << endl;</code>
8	
9	<code>cout << "r1 > r2 : " << r1.lebihDari(r2) << endl;</code>
10	<code>cout << endl;</code>
11	
12	<code>cout << "r1: ";</code>
13	<code>r1.cetak();</code>
14	<code>cout << "r2: ";</code>
15	<code>r2.cetak();</code>
16	<code>cout << endl;</code>
17	
18	<code>r1.sederhana();</code>
19	<code>r2.sederhana();</code>
20	
21	<code>cout << "Hasil dari penyederhanaan adalah " << endl;</code>
22	<code>cout << "r1: ";</code>
23	<code>r1.cetak();</code>
24	<code>cout << "r2: ";</code>
25	<code>r2.cetak();</code>
26	<code>cout << endl;</code>
27	
28	<code>cout << "Setelah dilakukan Cast ke double menjadi " << endl;</code>
29	<code>cout << "r1: " << r1.cast() << endl;</code>
30	<code>cout << "r2: " << r2.cast() << endl;</code>
31	<code>cout << endl;</code>
32	
33	<code>r1.negasi();</code>
34	<code>cout << "Negasi dari r1: ";</code>
35	<code>r1.cetak();</code>
36	<code>cout << endl;</code>
37	
38	<code>r1.unaryPlus(r2);</code>
39	<code>cout << "Nilai dari 'r1 += r2': ";</code>
40	<code>r1.cetak();</code>
41	<code>}</code>

5. Lakukan kompilasi dan tuliskan hasil percobaan, analisis hasil, dan kesimpulannya.

3.6 Hasil Percobaan

1. Tuliskan hasil dari percobaan pertama di atas! Perbaiki jika ada kode yang salah!
2. Tuliskan hasil dari percobaan kedua di atas! Perbaiki jika ada kode yang salah!

3.7 Analisis Hasil

1. Percobaan pertama:
 - a. Jelaskan fungsi dari statement siswa() pada baris 11 di dalam class Siswa!
 - b. Jelaskan persamaan dan perbedaan dari statement pada baris 12 dibandingkan 13, 14, dan 15 di dalam class Siswa!
 - c. Jelaskan perbedaan dalam pembuatan object antara object anna dan chirs pada method main!
 - d. Jelaskan perbedaan statement pada 13 dibandingkan 22, 23, dan 24 di dalam method main!
2. Percobaan pertama:
 - a. Jelaskan fungsi loop while pada baris 21 di dalam method sederhana, dan mengapa kondisinya harus b != 0!

- b. Jelaskan apa fungsi dari method unaryPlus, dan mengapa variabel pembilang dan penyebut harus melakukan statement tersebut!
- c. Jelaskan apa fungsi dari `static_cast<double>` dan apa yang terjadi apabila pernyataan tersebut dihilangkan!

3.8 Kesimpulan

- 3. Tuliskan kesimpulan dari percobaan pertama di atas!
- 4. Tuliskan kesimpulan dari percobaan kedua di atas!

3.9 Latihan

- 1. Percobaan pertama:
 - a. Tambahkan constructor pada class Siswa dengan parameter yang mempunyai parameter masing-masing nilai dari mata pelajaran yang ada! Kemudian buatlah sebuah object pada method main!
 - b. Tambahkan method dengan nilai balikan berupa boolean pada class Siswa bernama statusAkhir untuk menentukan apakah siswa tersebut remidi atau tidak. Ketentuannya adalah jika nilai lebih dari atau sama dengan 61 adalah lolos sedangkan nilai kurang dari atau sama dengan 60 adalah remidi. Nilai yang dicari adalah nilai rata-rata untuk semua mata pelajaran. Kemudian nilai pada method statusAkhir tampilkan pada method cetak!
 - c. Bagaimana cara menghitung banyaknya object yang kita buat dari method main? Tuliskan kodenya kemudian tampilkan informasinya dengan memanggil method jumlahObjek() bertipe void!
- 2. Percobaan kedua:
 - a. Tambahkan method untuk operator `<`, `<=`, dan `>=`!
 - b. Ubah method sederhana pada baris 21-25 yang awalnya menggunakan while menjadi for!
 - c. Tambahkan method untuk operasi `-`, `*`, dan `/`!

3.10 Tugas

- 1. Buatlah implementasi sebuah mesin ATM. Dalam mesin ATM, password dapat dimasukkan oleh user. Selain itu, sistem memiliki pilihan menu melihat saldo, menarik uang, dan menstransfer ke rekening lain!

Modul 4 : Enkapsulasi

4.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = 170 menit, dengan rincian sebagai berikut (misalkan):
15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
60 menit untuk penyampaian materi
45 menit untuk pengerjaan jurnal, tes akhir atau tugas
50 menit Pengayaan

4.2 Tujuan

1. Praktikan mampu memahami konsep encapsulation (enkapsulasi) yang ada di C++
2. Mampu memahami dan mengimplementasikan encapsulation

4.3 Dasar Teori

Enkapsulasi adalah suatu cara untuk menyembunyikan informasi detail dari suatu class. Dalam enkapsulasi terdapat hak akses *public*, *protected*, dan *private*. Hak akses *public* memungkinkan semua kelas dapat mengakses meskipun berada pada paket yang berbeda, hak akses *protected* hanya diberikan kepada kelasnya sendiri dan turunannya, serta kelas-kelas dalam satu paket. Sedangkan *private* hanya boleh diakses oleh kelasnya sendiri.

Enkapsulasi bertujuan untuk menjaga suatu proses program agar tidak dapat diakses secara sembarangan atau diintervensi oleh program lain. Konsep enkapsulasi sangat penting dilakukan untuk menjaga kebutuhan program agar dapat diakses sewaktu-waktu, sekaligus menjaga program tersebut. Dua hal yang mendasar dalam enkapsulasi yakni :

A.1 Information Hiding

Sebelumnya, kita dapat mengakses anggota class baik berupa atribut maupun method secara langsung dengan menggunakan objek yang telah kita buat. Hal ini dikarenakan akses kontrol yang diberikan kepada atribut maupun method yang ada di dalam class tersebut adalah 'public'. Kita dapat menyembunyikan informasi dari suatu class sehingga anggota class tersebut tidak dapat diakses dari luar, caranya adalah hanya dengan memberikan akses kontrol 'private' ketika mendeklarasikan atribut atau method. Proses ini disebut dengan information hiding.

Informasi berupa variable atau method kadang kala perlu disembunyikan sebab bisa jadi variabel atau method tersebut memang hanya dibutuhkan di dalam class itu saja, dan tidak diperlukan diakses dari luar.

A.2 Interface to Access Data

Jika kita telah melakukan information hiding terhadap suatu atribut pada suatu class, lalu bagaimana melakukan perubahan terhadap atribut yang kita sembunyikan tersebut. Caranya adalah dengan membuat suatu interface berupa method untuk menginisialisasi atau merubah nilai dari suatu atribut tersebut. Manfaat utama teknik ini adalah kita dapat melakukan validasi terhadap nilai yang akan diberikan kepada variable. Dengan melalui method interface ini, programmer yang menggunakan class tersebut, dipaksa memasukkan nilai yang valid, yang sudah didefinisikan dalam method itu.

A. Accessor

Untuk mengimplementasikan enkapsulasi, kita tidak menginginkan sembarang object dapat mengakses data kapan saja. Untuk itu, kita deklarasikan atribut dari class sebagai private. Namun, ada kalanya dimana kita menginginkan object lain untuk dapat mengakses data private. Dalam hal ini kita gunakan accessor methods.

Accessor Methods digunakan untuk membaca nilai variabel pada class, baik berupa instance maupun static. Sebuah accessor method umumnya dimulai dengan penulisan *get<namaInstanceVariable>*. Method ini juga mempunyai sebuah return value. Sebagai contoh, kita ingin menggunakan accessor method untuk dapat membaca nama, alamat, nilai bahasa Inggris, Matematika, dan ilmu pasti dari siswa. Mari kita perhatikan salah satu contoh implementasi accessor method.

```
//StudentRecord.h
class StudentRecord{
public:
    string getName();
};

//StudentRecord.cpp
string StudentRecord::getName() {
    return name;
}
```

B. Mutator

Method yang dapat memberi atau mengubah nilai variable dalam class, baik itu berupa instance maupun static. Method semacam ini disebut dengan mutator methods. Sebuah mutator method umumnya tertulis *set<namaInstanceVariabel>*. Mari kita perhatikan salah satu dari implementasi mutator method.

```
//StudentRecord.h
class StudentRecord{
public:
    string setName(string newName);
};

//StudentRecord.cpp
string StudentRecord::setName(string newName){
    name = newName;
}
```

4.4 Prosedur Praktikum

1. Kerjakan program berikut, dan benahi jika ada kesalahan
2. Tuliskan hasil percobaan pada seksi berikutnya
3. Lakukan analisis terhadap hasil yang diperoleh

Student.h	
1	#include <iostream>
2	#include <string>
3	
4	using namespace std;
5	
6	#pragma once
7	class Student
8	{
9	public:
10	Student(void);
11	void setName(string newName);
12	string getName();
13	void setMark(double newMark);

14	double getMark();
15	~Student(void);
16	
17	string name;
18	double mark;
19	};

Student.cpp	
1	#include "stdafx.h"
2	#include "Student.h"
3	
4	Student::Student(void)
5	{ }
6	
7	void Student::setName(string newName){
8	name = newName;
9	}
10	
11	string Student::getName(){
12	return name;
13	}
14	
15	void Student::setMark(double newMark){
16	mark = newMark;
17	}
18	
19	double Student::getMark(){
20	return mark;
21	}
22	
23	Student::~~Student(void)
24	{ }

Ketikkan juga program di bawah ini untuk menggunakan class Student

MainStudent.cpp	
1	#include "stdafx.h"
2	
3	#include <iostream>
4	#include <string>
5	using namespace std;
6	
7	#include "Student.h"
8	
9	
10	int main(){
11	Student siswa1;
12	siswa.name = "Rahmat";
13	siswa.mark = 98;
14	
15	cout<<"Siswa dengan nama "<<siswa.name;
16	cout<<" mendapat nilai "<<siswa.mark;
17	cin.get();
18	
19	}

4.5 Percobaan

- Ketikkan kode tersebut, perbaiki jika ada kesalahan, jelaskan apa yang dilakukan oleh kode pada baris 12, 13, 15, dan 16 pada mainStudent.cpp

.....

.....

5. Pada file Student.h, pada baris ke 16, tambahkan `private:` jelaskan apa yang terjadi
.....
.....
6. Jika pada setelah ditambahkan `private:` muncul error, perbaiki kode tersebut, dan jelaskan langkah perbaikan yang sudah dilakukan
.....
.....
7. Kembalikan kode ke bentuk semula (tanpa variable `private`, dan perbaiki jika ada error), Jika diketahui nilai seorang siswa (`mark`) memiliki rentang antara 0 hingga 100, bisakah anda memasukkan angka -20 atau 200 pada program tersebut?
.....
.....

4.6 Analisis Hasil

Tuliskan hasil analisis anda berdasarkan percobaan yang telah dilakukan

.....
.....

4.7 Kesimpulan

Kesimpulan dari percobaan enkapsulasi

.....
.....

4.8 Latihan

1. Jika rentang nilai `mark` pada class `Student` adalah 0 hingga 100, tambahkan kode untuk validasi pada method `setMark` seperti yang dijelaskan pada bab teori
.....
.....
2. Masukkan kode validasi pada latihan nomor 1 ke dalam sebuah `private method`, dan kemudian method itu dipanggil di method `setMark`
.....
.....
3. Buat class `Circle` yang sebuah lingkaran. Class ini memiliki jari-jari, dapat menghitung luas dan keliling. Lengkapi dengan constructor, destructor, variable dan/atau method `private`
.....
.....

Modul 5 : Inheritance

5.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut (misalkan):

- m. 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- n. 60 menit untuk penyampaian materi
- o. 45 menit untuk pengerjaan jurnal, tes akhir atau tugas
- p. 50 menit **Pengayaan**

5.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

- 1. Dapat memahami proses pada Inheritance
- 2. Memahami perbedaan yang dimiliki oleh tiap-tiap inheritance (tunggal, jamak dan virtual)
- 3. Memahami constructor dan destructor pada Inheritance

5.3 Alat & Bahan

- 6. IDE Bahasa C++

5.4 Dasar Teori

Pewarisan (inheritance) adalah penurunan sifat yang ada pada suatu kelas kepada kelas baru yang menjadi turunannya. Sifat-sifat suatu kelas ditentukan oleh data anggota dan metode, sehingga yang dimaksud penurunan sifat adalah penurunan data anggota atau metode. Kelas yang menurunkan sifat disebut kelas dasar (base class), sedangkan yang kelas baru yang mewarisi sifat kelas dasar disebut kelas turunan (derived class).

Dengan pewarisan dapat diciptakan suatu kelas baru yang mana kelas tersebut mewarisi seluruh sifat kelas dasar yang mempunyai akses public atau protected ditambah sifat khusus dari kelas yang bersangkutan. Ketentuan ini tidak berlaku sebaliknya, artinya sifat yang ada pada kelas turunan tidak diwariskan pada kelas dasar.

Ketentuan utama dengan adanya pewarisan yaitu memungkinkan suatu kode yang telah ditulis mudah sekali untuk digunakan kembali. Anda telah membuat suatu kelas yang telah diuji. Jika suatu nanti, anda ingin menerapkan kode tersebut pada sesuatu yang mempunyai sifat-sifat kelas tersebut, anda tinggal mewariskan kelas yang telah ada di kelas baru (kelas turunan). Anda tidak perlu mengotak-atik kelas yang teruji, sehingga efek samping yang tidak diharapkan tidak terjadi. Dengan cara ini pengembangan program menjadi lebih efisien dan menghemat waktu. Selain itu, anda dapat menambahkan sifat-sifat baru yang tidak ada pada kelas dasar atau bahkan dapat mengganti sifat-sifat pada kelas turunan, yang berbeda dengan sifat kelas dasar.

Dari mekanisme pewarisan yang sudah diuraikan diatas, dapat disimpulkan bahwa pewarisan ini dikelompokkan menjadi tiga, antara lain

1. Pewarisan Tunggal (Single Inheritance)

Adalah pewarisan yang mana jumlah kelas dasarnya tunggal. Pada pewarisan ini, kelas turunan dapat berjumlah lebih dari satu. Pewarisan tunggal dapat digambarkan dengan sintak program sebagai berikut :

Struktur Single Inheritance	
1	class A
2	{
3	. . .

4	};
5	class B : public A
6	{
7	. . .
8	}

Sintak di atas adalah mekanisme pewarisan secara public. Dengan implementasi di atas, kelas B merupakan kelas turunan dari kelas A. Selain pewarisan public, pewarisan juga dilakukan secara protected maupun private.

2. Pewarisan Jamak (Multiple Inheritance)

Adalah pewarisan dimana satu kelas diturunkan lebih dari satu kelas yang berbeda. Dalam pewarisan ini jumlah kelas dasarnya lebih dari satu, dan perlu dicatat bahwa kelas dasarnya bukan merupakan turunan dari satu kelas. Kelas turunan mewarisi seluruh sifat dari kelas dasarnya, sehingga sifat dari beberapa kelas dasar dan sifat khas dirinya. Perhatikan sintak dari pewarisan tunggal berikut ini :

Struktur Multiple Inheritance	
1	class A
2	{
3	. . .
4	};
5	class B
6	{
7	. . .
8	}
9	class C: public A, public B
10	{
11	. . .
12	}

Pada bentuk tersebut terdapat dua kelas dasar yaitu kelas A dan kelas B yang menurunkan kelas C. Kelas C akan mewarisi sifat dari kelas A maupun sifat dari kelas B, tetapi tidak berlaku sebaliknya.

3. Pewarisan Jamak Maya (Virtual Multiple Inheritance)

Adalah pewarisan yang mana kelas dasarnya lebih dari satu dan beberapa di antara kelas dasar tersebut merupakan kelas turunan dari kelas dasar yang sama. Mekanisme pewarisan sifat suatu kelas dasar kepada kelas turunan sama dengan pewarisan yang lain. Perhatikan sintak berikut ini :

Struktur Virtual Multiple Inheritance	
1	class A
2	{
3	. . .
4	};
5	class B: virtual public A
6	{
7	. . .
8	};
9	class C: virtual public A
10	{
11	. . .

```

12 };
13 class D: public B, public C
14 {
15     . . .
16 };

```

Kelas D merupakan turunan dari kelas B dan C, sedangkan kelas B dan C merupakan kelas turunan dari kelas dasar yang sama yaitu kelas A. Supaya berjalan pewarisan dari kelas A kepada kelas B maupun C harus secara virtual. Kelas virtual A pertama menurunkan kelas B, sedangkan kelas virtual A kedua menurunkan kelas C. Contoh program dalam pewarisan ini :

4. Konstruktor Dan Destruktor Pada Inheritance

Di dalam contoh tersebut diatas belum melibatkan konstruktor dan destruktur secara eksplisit. Di dalam pewarisan suatu konstruktor perlu diperhatikan terutama konstruktor penyalinan. Jika konstruktor kelas dasar hanya berisi pernyataan memberi nilai data anggota private, maka akses private dapat diganti dengan akses protected agar data anggota pada kelas dasar dapat diakses dari kelas turunan.

```

Inheritance.cpp
1  #include<iostream.h>
2  #include<string.h>
3  Class Kendaraan
4  {
5      private:
6      char nama[15];
7      public:
8      void berkembang(char *nama_kendaraan = "XXX")
9      {
10         Strcpy (nama, nama_kendaraan);
11         Cout<< "Hidupkan mesin kendaraan anda..."<< endl;
12     }
13     ~Kendaraan()
14     {
15         Cout << "Matikan mesin kendaraan anda..." << endl;
16     }
17     Void info_kendaraan()
18     {
19         Cout << nama << "sedang berjalan..." << endl;
20     }
21 };
22 class Truk : public Kendaraan
23 {
24     public:
25     Truk (char *nama_truk);
26     {
27         Strcpy(nama, nama_truk);
28         Cout << "Hidupkan mesin truk..."<<nama <<endl;
29     }
30     ~Truck ()
31     {
32         Cout << "Matikan mesin truk anda...";
33     }
34 };
35 main()

```

```

36 {
37     System("cls");
38     Truk fuso("TRUK FUSO");
39     Fuso.info_kendaraan();
40     Cout << "Akhir main ()...";
41     System("pause");
42     Return 0;
43 }

```

Hasilnya sebagai berikut:

Hidupkan mesin kendaraan anda...

Hidupkan mesin truk...TRUK FUSO

TRUK FUSO sedang berjalan...

Akhir main ()...

Matikan mesin truk anda...

Matikan mesin kendaraan anda...

Catatan:

Ø Konstruktorkelas dasar dengan sendirinya dijalankan

Ø Kemudian konstruktorkelas turunan baru dijalankan

Pada saat obyek berakhir :

Ø Destruktorkelas turunan dijalankan terlebih dahulu

Ø Kemudian destruktorkelas dasar dijalankan.

4. Overriding

Overriding merupakan suatu keadaan dimana kelas anak dapat mengubah atau bisa kita bilang memodifikasi atau memperluas data dan method pada kelas induk. Keuntungan Overriding yaitu dapat menambahkan sifat / atribut pada kelas induknya.

Overriding mempunyai ciri-ciri sebagai berikut :

1. Nama method harus sama
2. Daftar parameter harus sama
3. Return type harus sama

```

Override.cpp
1  #include <iostream>
2  using namespace std;
3
4  class Sepeda
5  {
6      public:
7          int JmlRoda()
8          {
9              return 2;
10         }
11 };
12
13 class SepedaBayi:public Sepeda
14 {
15     public:
16         int JmlRoda()
17         {
18             return 3;

```


19	}
20	};
21	
22	int main(void)
23	{
24	SepedaBayi sepedaku;
25	cout << "Jumlah roda : " << sepedaku.JmlRoda() << endl;
26	return 0;
26	}

5.5 Prosedur Praktikum

Jalankan masing-masing kode program berikut untuk masing-masing bentuk Inheritance

1. Pewarisan Tunggal (single inheritance)

SingleInheritance.cpp	
1	#include<iostream.h>
2	class makhluk
3	{
4	public:
5	void berkembang();
6	};
7	class hewan : public makhluk
8	{
9	public:
10	void bergerak();
11	};
12	class kuda : public hewan
13	{
14	public:
15	void berlari();
16	};
17	main()
18	{
19	makhluk mk; hewan hw; kuda kd;
20	cout<<endl<<" Sifat-sifat dari Makhluk adalah : "<<endl;
21	mk.berkembang();
22	cout<<endl<<" Sifat-sifat dari Hewan adalah : "<<endl;
23	hw.berkembang(); hw.bergerak();
24	cout<<endl<<" Sifat-sifat dari Kuda adalah : "<<endl;
25	mk.berkembang(); hw.bergerak(); kd.berlari();
26	}
26	void makhluk::berkembang()
28	{
29	cout<<" Berkembang biak"<<endl;
30	}
31	void hewan::bergerak()
32	{
33	cout<<" Bergerak berpindah tempat"<<endl;
34	}
35	void kuda::berlari()
36	{
37	cout<<" Berlari sangat kencang seperti angin"<<endl;

2. Pewarisan Jamak (multiple inheritance)

MultipleInheritance.cpp

```

1  #include<iostream.h>
2  class kuda
3  {
4      public :
5      void berlari()
6      {
7          cout<<" > Berlarinya sangat cepat"<<endl;
8      }
9  };
10 class burung
11 {
12     public:
13     void terbang()
14     {
15         cout<<" > Terbang menembus awan"<<endl;
16     }
17 };
18 class pegasus: public kuda, public burung
19 {
20     public:
21     void lariterbang()
22     {
23         cout<<" > Bersayap, lari dan dapat terbang ke
24         angkasa"<<endl;
25     }
26 };
27 main()
28 {
29     pegasus pg;
30     cout<<"Sifat dari PEGASUS yaitu : "<<endl;
31     pg.berlari();
32     pg.terbang();
33     pg.lariterbang();
34 }
```

3. Pewarisan Jamak Maya (virtual multiple inheritance)

VirtualMultiInheritance.cpp

```

1  #include"iostream.h"
2  class hewan
3  {
4      public:
5      void bergerak()
6      {
7          cout<<" # Bergerak berpindah tempat"<<endl;
8      }
9  };
10 class kuda: virtual public hewan
11 {
12     public :
```

13	void berlari()
14	{
15	cout<<" # Berlarinya sangat cepat"<<endl;
16	}
17	};
18	class burung: virtual public hewan
19	{
20	public:
21	void terbang()
22	{
23	cout<<" # Terbang menembus awan"<<endl;
24	}
25	};
26	class pegasus: public kuda, public burung
27	{
28	public:
29	void lariterbang()
30	{
31	cout<<"# Bersayap, lari dan dapat terbang ke angkasa"<<endl;
32	}
33	};
34	main()
35	{
36	pegasus pg;
37	cout<<">> Sifat dari PEGASUS << "<<endl;
38	cout<<"===== "<<endl;
39	pg.bergerak();
40	pg.berlari();
41	pg.terbang();
42	pg.lariterbang();
43	}
44	

4. Konstruktor dan Destructor pada Inheritance

ConstructorandDestructorInheritance.cpp	
1	#include"iostream.h"
2	#include"conio.h"
3	#include"string.h"
4	class Kendaraan
5	{
6	private:
7	char nama[15];
8	public:
9	Kendaraan(char *nama_kendaraan = "T1AS")
10	{
11	strcpy(nama, nama_kendaraan);
12	cout<<" Hidupkan mesin kendaraan anda ..."<<endl;
13	}
14	~Kendaraan()
15	{
16	cout<<" Matikan mesin kendaraan anda ..."<<endl;
17	}
18	void info_kendaraan()

```

19     {
20         cout<<nama<<" Sedang berjalan ..."<<endl;
21     }
22 };
23 class Mercy : public Kendaraan
24 {
25     public:
26     Mercy(char *nama_mercy) : Kendaraan(nama_mercy)
27     {
28         cout<<" Hidupkan mesin mobil merah ..."<<endl;
29     }
30     ~Mercy()
31     {
32         cout<<" Matikan mesin mobil merah itu ..."<<endl;
33     }
34 };
35 void main()
36 {
37     clrscr();
38     Mercy mewah(" Mobil Yang Mewah");
39     mewah.info_kendaraan();
40     cout<<" Akhir dari permulaaan()..."<<endl;
41 }

```

5. Overriding

```

Overriding.cpp
1
2 using namespace std;
3 class OvrRide {
4 private : int nilai1 , nilai2;
5 public : void Input(int a, int b) {
6     nilai1 = a;
7     nilai2 = b;
8 }
9 public : virtual void show(){
10 int hasil = nilai1 + nilai2;
11 cout << "Override 1" << endl;
12 cout << "Hasil penjumlahan " << nilai1 << " dan " << nilai2 << "
13 = " << hasil;
14 cout << endl;cout << endl;
15 }
16 };
17 class Ride : public OvrRide {
18 public : void show(){
19 int hasil = nilai1 + nilai2;
20 cout << "Override 2" << endl;
21 cout << "Hasil penjumlahan " << nilai1 << " dan " << nilai2 << "
22 = " << hasil;
23 cout << endl;cout << endl;
24 }
25 };
26 int main(int argc, char *argv[])
27 {

```

28	OvrRide over;
29	Ride ride;
30	over.Input(3, 4);
31	over.show();
32	ride.Input(5, 6);
33	ride.show();
34	system("PAUSE");
35	return EXIT_SUCCESS;
36	}

5.6 Hasil Percobaan

Data dan Analisis hasil percobaan

Pertanyaan

1. Jalankan code program diatas dan benahi jika menemukan kesalahan!

.....

2. Apakah hasil dari masing-masing nomor percobaan?

Percobaan ke-1
 Percobaan ke-2.....
 Percobaan ke-3.....
 Percobaan ke-4.....
 Percobaan ke-5.....

5.7 Analisis Hasil

Berikan analisis terhadap masing-masing percobaan

1. Apakah variabel yang berpengaruh untuk tiap-tiap percobaan?

Percobaan ke-1
 Percobaan ke-2.....
 Percobaan ke-3.....
 Percobaan ke-4.....
 Percobaan ke-5.....

2. Pada bab sebelumnya anda telah belajar mengenai konsep encapsulation, jelaskan mengapa pada super class menggunakan modifier protected? Apa yang terjadi jika modifier anda ubah menjadi private atau public? Jelaskan !

.....

.....

3. Ubahlah acces modifier method pada kelas employee menjadi :

- a. Private
- b. Protected

4. Amati perubahan apa yang terjadi? Jelaskan jawaban anda dengan detail!

.....

.....

5. Berdasarkan hasil eksekusi Percobaan ke-4 di atas terlihat bahwa, pada saat obyek berkelas Turunan diciptakan :

.....

.....

6. Berdasarkan hasil eksekusi Percobaan ke-5 di atas, bagaimanakah kelas turunan dapat meng-override kelas parent?

.....

.....

5.8 Kesimpulan

.....

5.9 Latihan

.....

5.10 Tugas

Tugas Pengayaan Praktikum

Buatlah class Employee yang d-inherit oleh class Manager, Pegawai tetap, Pegawai tidak tetap. Setelah itu program akan meminta user untuk menginputkan nama, idKerja, jabatan employee, mempunyai istri atau tidak, mempunyai anak atau tidak dan memasukkan tahun masuk kerja. Ketentuan untuk mengerjakan soal ini adalah sebagai berikut :

- a. Berlaku untuk semua jabatan (kecuali non PNS)
 - i. Jika masa kerja di bawah 5 tahun maka tidak mendapatkan bonus dan tidak mendapatkan tunjangan (sama dengan employee)
 - ii. Jika masa kerja di antara 6-10 tahun maka mendapatkan bonus 0.05 dari lama kerja dan tidak mendapatkan tunjangan
 - iii. Jika masa kerja di atas 10 tahun maka mendapatkan bonus 0.1 dari lama kerja dan mendapatkan tunjangan 0.1 dari lama kerja
 - iv. Jika mempunyai istri maka akan mendapatkan tunjangan 0.1 dari total gaji yang didapat (gaji+lama kerja+tunjangan) jika tidak maka tunjangan istri 0 rupiah

- v. Jika mempunyai anak maka mendapatkan tunjangan anak sebesar 0.15 dari jumlah gaji yang didapatkan (gaji+lama kerja+tunjangan), tunjangan anak hanya diberikan sampai anak ketiga saja.
- b. Berlaku untuk Manager saja
 - i. Untuk manager selain mendapatkan bonus kerja dan tunjangan, pada manager mendapatkan tunjangan jabatan sebesar 0.1 dari jumlah gaji (gaji+lama kerja+tunjangan).
- c. Berlaku untuk pegawai tidak tetap
 - i. Gaji yang didapatkan adalah sama dengan gaji employee, namun terdapat gaji lembur. Untuk mendapatkan gaji lembur, pegawai non PNS harus bekerja lebih dari 10 jam dengan mendapatkan gaji lembur sebesar 10.000 perjamnya. Namun jika di bawah 10 jam maka tidak akan mendapatkan gaji lembur.

Untuk mengerjakan soal ini gunakan test case sebanyak 1 Manager, 10 Pegawai tetap dan 5 pegawai tidak tetap yang mencakup seluruh ketentuan yang ada.

..... *

Modul 6 : Polimorfisme

6.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut.

- q. 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- r. 30 menit untuk penyampaian materi
- s. 65 menit untuk pengerjaan tugas / Studi Kasus
- t. 60 menit **Pengayaan**

6.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

- 6. Mampu memahami prinsip polimorfisme dan pemakaiannya dalam membentuk suatu class
- 7. Mampu membedakan antara polimorfisme dengan inheritance

6.3 Alat & Bahan

- 7. Komputer
- 8. DevCpp

6.4 Dasar Teori

6.4.1 Polimorfisme

Polimorfisme (memiliki banyak bentuk) menyatakan kemampuan untuk memperlakukan objek-objek dengan cara yang seragam meskipun objek-objek tersebut berbeda perilaku. Polimorfisme adalah suatu sifat yang merupakan efek langsung dari sifat inheritance pada OOP. Jika pada inheritance semua perilaku yang diturunkan kepada derived-class memiliki bentuk yang sama, namun pada polimorfisme ini subclass dapat mendefinisikan perilaku yang sama dengan cara yang berbeda, yang mana perilaku ini merupakan turunan dari base-class.

Sebagai contoh pada suatu game, polimorfisme digunakan untuk musuh yang pasti dapat menyerang. Misalkan musuh di dalam game tersebut dibagi menjadi dua, yaitu zombie dan monster. Zombie dapat menyerang dengan cara menggigit sedangkan monster dengan cara memakan pemain. Dari contoh tersebut terlihat bahwa kedua objek (zombie dan monster) sama-sama melakukan serangan namun dengan cara yang berbeda (memiliki fungsi sama dengan tugas berbeda).

Implementasi polimorfisme dari contoh di atas adalah sebagai berikut.

```
#include <cstdlib>
#include <iostream>

using namespace std;

class Musuh {
protected:
    int nilaiSerang;
public:
    void setSerang(int nSerang){
        nilaiSerang = nSerang;
    };
};

class Zombi: public Musuh {
public:
    void serang(){
        cout << "Zombi menggigit orang!! -" << nilaiSerang << endl;
    };
};
```



```

};

class Monster: public Musuh {
public:
    void serang(){
        cout << "Monster makan orang!! -" << nilaiSerang << endl;
    };
};

int main(){
    Zombi z;
    Monster m;
    Musuh *musuh1 = &z;
    Musuh *musuh2 = &m;
    musuh1->setSerang(10);
    musuh2->setSerang(50);
    z.serang();    // musuh1 tidak dapat mengakses fungsi serang
    m.serang();    // class Musuh tidak memiliki fungsi serang
}

```

6.4.2 Fungsi virtual

Polimorfisme yang sudah dibuat sebelumnya pointer objek musuh (musuh1 dan musuh2) tidak dapat mengakses fungsi setSerang secara langsung, hal ini dikarenakan pada class Musuh tidak memiliki fungsi serang. Apabila dilihat pada class Musuh dan Monster (derived-class) memiliki nama fungsi yang sama, yaitu serang. Oleh karena itu, fungsi tersebut dapat ditulis di bagian class Musuh (base-class). Agar fungsi serang dapat diakses dengan memberikan perilaku yang berbeda, maka perlu dibuat fungsi virtual. Fungsi virtual dibagi menjadi dua, yaitu fungsi virtual biasa (virtual function) dan fungsi virtual murni (pure virtual function).

6.4.2.1 Fungsi virtual

Fungsi virtual merupakan suatu fungsi yang dapat dideklarasikan ulang di class turunannya (override) dan pemanggilan body function-nya secara realtime (dynamic binding). Hal ini berbeda dengan fungsi tanpa virtual yang di-override. Pemanggilan body function-nya dijalankan sesuai apa yang telah didefinisikan (static binding). Adapun contoh fungsi virtual dari contoh sebelumnya seperti kode di bawah ini. Terlihat bahwa dengan menggunakan fungsi virtual, suatu objek dari base-class dapat mengakses body function dari fungsi yang ada di derived-class.

```

class Musuh {
public:
    virtual void serang(){
        cout << "Serangan musuh!" << endl;
    };
};

class Zombi: public Musuh {
public:
    void serang(){
        cout << "Zombi menggigit orang!!" << endl;
    };
};

class Monster: public Musuh {
public:
    void serang(){
        cout << "Monster makan orang!!" << endl;
    };
};

```

```

int main() {
    Zombi z;
    Monster m;
    Musuh *musuh1 = &z;
    Musuh *musuh2 = &m;
    musuh1->serang();
    musuh2->serang();
}

```

6.4.2.2 Fungsi virtual murni

Dari contoh fungsi virtual di atas, fungsi virtual `serang` memiliki body function padahal statement di dalam body function tidak pernah digunakan selama fungsi virtual tersebut di-override. Oleh karena itu, fungsi virtual tersebut harus diubah menjadi fungsi virtual murni. Fungsi virtual murni tidak diperbolehkan memiliki body function dan setiap derived-class-nya harus melakukan override terhadap fungsi tersebut. Adapun sintaksnya adalah dengan menambahkan “= 0” di akhir deklarasi fungsi. Sebagai contoh, kode fungsi virtual sebelumnya apabila diubah menjadi fungsi virtual seperti di bawah ini. Apabila suatu class memiliki setidaknya satu fungsi virtual murni maka class tersebut disebut sebagai abstract class. Suatu abstract class tidak dapat diinstansiasi secara langsung.

```

class Musuh {
public:
    virtual void serangan() = 0;
};

```

6.5 Prosedur Praktikum

C. Percobaan Pertama: Polimorfisme

6. Buatlah projek di DevCpp dengan nama Polimorfisme.
7. Buatlah class Poligon sebagai base class dan class Persegi serta Ssegitiga sebagai derived-class.

1	<code>class Poligon {</code>
2	<code>protected:</code>
3	<code>int lebar, tinggi;</code>
4	<code>public:</code>
5	<code>void setNilai (int a, int b)</code>
6	<code>{ lebar=a; tinggi=b; }</code>
7	<code>int luas ()</code>
8	<code>{ return 0; }</code>
9	<code>};</code>
10	
11	<code>class Persegi: public Poligon {</code>
12	<code>public:</code>
13	<code>int luas ()</code>
14	<code>{ return lebar * tinggi; }</code>
15	<code>};</code>
16	
17	<code>class Segitiga: public Poligon {</code>
18	<code>public:</code>
19	<code>int luas ()</code>
20	<code>{ return (lebar * tinggi / 2); }</code>
21	<code>};</code>

8. Tambahkan method main untuk dijalankan pertama kali.

1	<code>int main () {</code>
2	<code>Persegi psg;</code>
3	<code>Segitiga segi3;</code>

```

4   Poligon poli;
5
6   Poligon * pPoli1 = &psgi;
7   Poligon * pPoli2 = &segi3;
8   Poligon * pPoli3 = &poli;
9
10  pPoli1->setNilai (4,5);
11  pPoli2->setNilai (4,5);
12  pPoli3->setNilai (4,5);
13
14  cout << pPoli1->luas() << endl;
15  cout << pPoli2->luas() << endl;
16  cout << pPoli3->luas() << endl;
17
18  return 0;
19  }

```

9. Lakukan kompilasi dan tuliskan hasil percobaan, analisis hasil, dan kesimpulannya.

D. Percobaan Kedua: Fungsi Virtual

1. Buatlah projek di DevCpp dengan nama Fungsi Virtual.
2. Buatlah class Base dan Derived seperti kode di bawah ini.

```

1   class Base {
2   public:
3       void cetak ()
4       { cout << "cetak dari base class" << endl; }
5       virtual void cetakVirtual ()
6       { cout << "cetak virtual dari base class" << endl; }
7   };
8
9   class Derived: public Base {
10  public:
11      void cetak ()
12      { cout << "cetak dari derived class" << endl; }
13      void cetakVirtual ()
14      { cout << "cetak virtual dari derived class" << endl; }
15  };

```

3. Tambahkan method main untuk dijalankan pertama kali.

```

1   int main () {
2       Base* b = new Base();
3       Base* bd = new Derived();
4       Derived* d = new Derived();
5
6       b->cetak();
7       b->cetakVirtual();
8       bd->cetak();
9       bd->cetakVirtual();
10      d->cetak();
11      d->cetakVirtual();
12
13      return 0;
14  }

```

4. Lakukan kompilasi dan tuliskan hasil percobaan, analisis hasil, dan kesimpulannya.

6.6 Hasil Percobaan

1. Tuliskan hasil dari percobaan pertama di atas!

2. Tuliskan hasil dari percobaan kedua di atas!

6.7 Analisis Hasil

3. Percobaan pertama:
 - a. Jelaskan mengapa setelah dikompilasi menghasilkan keluaran seperti itu?
 - b. Coba hilangkan pernyataan baris ke-7 dan 8 pada class Poligon. Kompilasi dan tuliskan hasilnya! Jelaskan mengapa setelah diubah didapatkan keluaran seperti itu!
 - c. Setelah penghapusan pernyataan baris ke-7 dan 8, ubahlah kodenya sehingga menampilkan nilai luas yang sesuai dengan nilai lebar dan tingginya!
4. Percobaan kedua:
 - a. Jelaskan mengapa setelah dikompilasi menghasilkan keluaran seperti itu?
 - b. Ubahlah fungsi virtual pada class Base menjadi fungsi virtual murni. Kompilasi dan tuliskan hasilnya! Jelaskan mengapa setelah diubah didapatkan keluaran seperti itu dan berikan solusinya?
 - c. Setelah langkah di atas dijalankan, coba bandingkan hasilnya ketika pernyataan pada baris ke-2 diberi tanda komentar dengan pernyataan baris ke-3 diberi komentar! Jelaskan mengapa hasilnya seperti itu!

6.8 Kesimpulan

5. Tuliskan kesimpulan dari percobaan pertama di atas!
6. Tuliskan kesimpulan dari percobaan kedua di atas!

6.9 Latihan

3. Percobaan pertama:
 - a. Tambahkan constructor sebagai pengganti fungsi setNilai!
 - b. Tambahkan fungsi untuk menghitung keliling dengan menggunakan fungsi virtual! Untuk segitiga diasumsikan sebagai segitiga siku-siku!
4. Percobaan kedua:
 - a. Tambahkan destructor pada setiap class! Bagaimana keluarannya? Jelaskan mengapa menghasilkan keluaran seperti itu!
 - b. Buatlah kelas turunan dari class Derived dengan nama Derived2X dan berikan fungsi override dari class Derived, yang mana fungsi cetak diubah menjadi fungsi virtual sedangkan fungsi cetakVirtual diubah menjadi fungsi virtual murni! Buatlah objek dari kelas turunan tersebut dan jelaskan berdasarkan hasil yang diperoleh!

6.10 Tugas

1. Dalam sebuah perusahaan terdapat banyak pekerja, di antaranya adalah buruh, sales, dan proyek manajer. Adapun keterangan yang dimiliki oleh masing-masing pekerja adalah:
 - a. Setiap pekerja memiliki fungsi untuk menghitung gaji dan cetak informasi.
 - b. Buruh memiliki atribut nama, nip, upah mingguan.
 - c. Sales memiliki atribut nama, nip, gaji pokok, komisi, dan total penjualan.
 - d. Proyek manajer memiliki atribut nama, nip, gaji pokok, komisi, dan total hasil proyek.

Setiap kelas harus memiliki konstruktor, fungsi untuk menghitung gaji setiap pegawai, dan fungsi untuk menampilkan informasi yang dimiliki setiap jenis pekerja. Untuk perhitungan gajinya adalah sebagai berikut. 1.

- a. Buruh: $\text{gaji} = \text{upah mingguan}$.
- b. Sales: $\text{gaji} = \text{gaji pokok} + (\text{komisi} * \text{total penjualan})$.
- c. Proyek manajer: $\text{gaji} = \text{gaji pokok} + (\text{komisi} * \text{total hasil proyek}) - \text{pajak}$. Pajak dihitung 5% dari gaji pokok.

Implementasikan studi kasus di atas dengan menggunakan konsep polimorfisme.

Modul 7 : Graphical User Interface

7.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut (misalkan):

- u. 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- v. 60 menit untuk penyampaian materi
- w. 45 menit untuk pengerjaan jurnal, tes akhir atau tugas
- x. 50 menit **Pengayaan**

7.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Praktikan mengenal lingkungan dan prasarana untuk membuat program dengan GUI
2. Praktikan mampu membuat program sederhana dengan GUI

7.3 Alat & Bahan

9. WxDev C++
10. Update TangoImages.devpak

7.4 Prosedur Praktikum

Jalankan masing-masing kode program berikut untuk masing-masing bentuk Inheritance

1. Percobaan Pertama

- Buat wxFrame application baru.
- Atur nama sesuai keinginan atau gunakan 'Modul83'
- Simpan dalam folder baru 'Modul83'
- Atur properties sebagai berikut:

Class Name	Modul83Frm
File Name	Modul83Frm
Title	Modul83
Author	Nama Anda
Window Style	Use Caption, Resize Border, System Menu, No Parent, Min Button, Max Button, Close Button
Height	402
Width	553
Icon	Gunakan internet-web-browser16x16.png tango icons yang diinstall dari TangoImages devpak

- Membuat Toolbar
 - Ubah component selector menjadi Toolbar.
 - Pilih toolbar component dan drop kedalam form.
- Membuat tombol pada Toolbar
 - Masukkan items berikut dari Toolbar component palette ke toolbar.
 - Ubah properties sesuai dengan keterangan.
 - Untuk bitmaps gunakan Tango Icons 16x16 folder.

Type	Tool Button
Bitmap	document-new16x16.png
Tooltip	'Create a new document'

Type	Tool Button
Bitmap	document-open16x16.png
Tooltip	'Open an existing document'
Type	Tool Button
Bitmap	document-save16x16.png
Tooltip	'Save this document'
Type	Separator
Type	Tool Button
Bitmap	document-print16x16.png
Tooltip	'Print this document'
Type	Tool Button
Bitmap	document-print-preview16x16.png
Tooltip	'Preview this document'
Type	Separator
Type	Tool Button
Bitmap	edit-undo16x16.png
Tooltip	'Undo last change'
Type	Tool Button
Bitmap	edit-redo16x16.png
Tooltip	'Redo last change'
Type	Separator
Type	ComboBox
Items	'Heading 1' 'Heading 2' 'Heading 3' 'Heading 4' 'Heading 5' 'Heading 6'
Text	'Heading 1'
Tooltip	'Alter the heading type'
Type	Tool Button
Bitmap	format-justify-left16x16.png
Tooltip	'Use inside a tag to left align text'
Type	Tool Button
Bitmap	format-justify-center16x16.png
Tooltip	'Use inside a tag to center align text'
Type	Tool Button
Bitmap	format-justify-right16x16.png
Tooltip	'Use inside a tag to right align text'



- b. Membuat status bar
 - Status bar ada pada Controls palette
 - Pilih Status bar component dan drop kedalam form.
- c. Kemudian kita perlu sizer component yang bs ditemukan di Sizer palette.
 - Pilih BoxSizer component dan drop kedalam form.
 - Pastikan Orientation diatur pada wxHorizontal.

- Kemudian kita akan membuat Toolbar disebelah kiri
 - Pilih sizer yang baru saja kalian buat.
 - Pilih Panel dari palette, cari pada tab Containers.
 - Masukkan panel ke dalam sizer.
 - Atur alignment wxEXPAND menjadi true dan pastikan yang lainnya false.
 - Atur Border property menjadi 0.
 - i. Atur panel Height 278 dan Width 36.
- d. Kita akan membuat tombol sesuai kriteria berikut

Type	Tool Button
Font	Arial, Bold, size 10
Height	30
Label	B
Left	3
Tooltip	'Bold text style'
Top	0
Width	30
Type	Tool Button
Font	Times New Roman, Italic, size 11
Height	30
Label	I
Left	3
Tooltip	'Italic text style'
Top	29
Width	30
Type	Tool Button
Font	Arial, Regular, size 12, Underline
Height	30
Label	U
Left	3
Tooltip	'Underlined text style'
Top	58
Width	30
Type	Tool Button
Font	Times New Roman, Italic, size 12
Height	30
Label	F
Left	3
Tooltip	'Change font style'
Top	87
Width	30

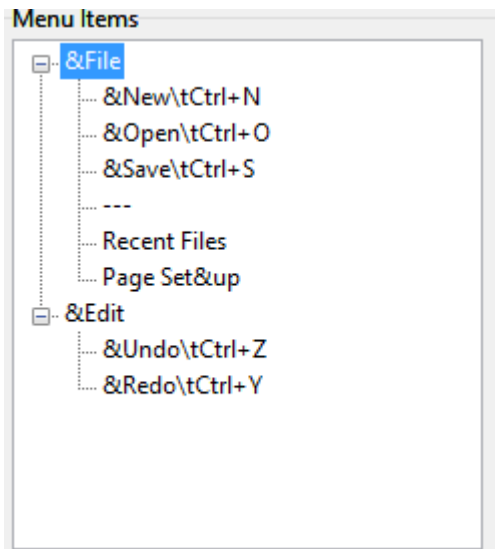
- e. Kemudian kita akan membuat Notebook component yang terdiri dari 2 halaman. Halaman pertama berfungsi sebagai preview dan edit pane, sedangkan halaman kedua sebagai preview saja
 - Pilih sizer Main pada form.
 - Pindah ke Window palette daan pilih NoteBook.
 - Masukkan kedalam sizer.
 - Ubah Alignment flag wxEXPAND menjadi true, pastikan yanglain adalah false.

- Atur Border property 0.
- Atur Height 275 dan Width 460.
- Atur Stretch Factor 1.
- f. Lalu kita akan membuat halaman
 - Pilih Notebook yang telah dibuat.
 - Pilih NotebookPage component dan masukkan ke dalam notebook.
 - Ubah Label property menjadi 'Combined HTML Preview'.
 - Pilih Notebook yang telah dibuat.
 - Pilih NotebookPage component lagi dan masukkan ke dalam notebook.
 - Pilih dengan click pada tab, kemudian click pada halamannya.
 - Ubah Label property menjadi 'Preview'.
- g. Setelah membuat halaman, kita perlu mengatur setting untuk mengontrolnya
 - Pilih halaman Combine HTML Preview dengan click pada tab kemudian click pada halaman
 - Pada Sizers palette Pilih BoxSizer.
 - Masukkan ke dalam halaman.
 - Pastikan sizer orientation adalah wxHORIZONTAL.
 - Sekaang pada Window palette pilih SplitterWindow dan masukkan pada sizer.
 - Pastikan splitter window's Alignment property adalah hanya wxEXPAND.
 - Atur splitter's Border property menjadi 0.
 - Atur Height 248 dan Width 195.
 - Pastikan Orientation property adalah wxVertical.
 - atur Stretch Factor menjadi 1.
 - Kemudian masukkan HTML Window dari palette yang sama.
 - Atur Height 89 dan Width 185.
 - Pada Controls palette pilih Memo control.
 - Masukkan control pada splitter window.
 - Atur Height 139 dan Width 185.
 - Gunakan Strings property untuk menghapus text yang ada.
 - Sekarang beralih pada halaman Preview dengan click tab Preview.
 - Kemudia click pada halamannya.
 - Masukkan BoxSizer pada tab ini.
 - Dari Window palette masukkan HTML Window pada sizer.
 - Atur HTML Windows Alignment hanya wxEXPAND.
 - Atur Border menjadi 0 dan Stretch Factor menjadi 1.
- h. Sekarang kita akan membuat MenuBar
 - Dari Menu palette pilih MenuBar control.
 - Masukkan kedalam area Main Form (bebas).
- i. Kemudian atur item properties pada Menu Item sebagai berikut

Type	Menu Item
Location	Root
Caption	&File
ID Name	Accept Default
ID Value	Accept Default
Type	Menu Item
Location	Child of File
Caption	&New\tCtrl+N
ID Name	Accept Default
ID Value	Accept Default
Hint	'Create a new document' document-new16x16.png

Bitmap	
Type	Menu Item
Location Caption ID Name ID Value Hint Bitmap	Child of File &Open\tCtrl+O Accept Default Accept Default 'Open an existing document' document-open16x16.png
Type	Menu Item
Location Caption ID Name ID Value Hint Bitmap	Child of File &Save\tCtrl+S Accept Default Accept Default 'Save current document' document-save16x16.png
Type	Separator
Location ID Name ID Value	Child of File Accept Default Accept Default
Type	File History
Location ID Name ID Value	Child of File Accept Default Accept Default
Type	Menu Item
Location Caption ID Name ID Value Hint Bitmap	Child of File Page Set&Up Accept Default Accept Default 'Set up the printer' document-properties16x16.png
Type	Menu Item
Location Caption ID Name ID Value	Root &Edit Accept Default Accept Default
Type	Menu Item
Location Caption ID Name ID Value Hint Bitmap	Child of Edit &Undo\tCtrl+Z Accept Default Accept Default 'Undo' edit-undo16x16.png
Type	Menu Item
Location Caption	Child of Edit &Redo\tCtrl+Y

ID Name	Accept Default
ID Value	Accept Default
Hint	Redo
Bitmap	edit-redo16x16.png



j.

k. Langkah terakhir

- Beralih pada Dialog palette.
- Pilih OpenFileDialog.
- Masukkan pada form.
- Ubah Extensions property menjadi '*.htm;*.html'.
- Ubah Message property menjadi 'Choose a file'.
- Kemudian pilih SaveFileDialog.
- Masukkan ke dalam form.
- Ubah Extensions menjadi '*.htm;*.html'.
- Ubah Message menjadi 'Choose a file'.
- Kemudian buat MatDialog.
- Ubah Caption menjadi 'Save Changes?'.
- Ubah Message menjadi 'The contents of this file has changed\n Do you want to save the changes?'.
- Atur Message Dialog Style menjadi wxICON_EXCLAMATION, wxYES_DEFAULT dan wxYES_NO.

2. Percobaan Kedua

- Setelah fungsi CreateGUIControls pada Modul83Frm.h, tambahkan sebagai berikut Modul83Frm.h

```
public:
    void UpdateHTML();
    void OpenFile();
    void SaveFile();
    void NewFile();
    bool DoFileSaveCheck();
    void MnuNewClick(wxCommandEvent& event);
    void WrapTextInTag(const wxString& TagType);
    void InsertText(const wxString& Text);
    void MnuOpenClick(wxCommandEvent& event);
```

```
void MnuSaveClick(wxCommandEvent& event);
```

- Tambahkan code dibawah setelah `////GUI Control Declaration End`

```
wxString MRUFile;
```

- Tambahkan code dibawah pada Modul83Frm.cpp pada bagian paling bawah

```
void Modul83Frm::UpdateHTML()
{
    //Update and refresh the text on the HTML preview windows
    WxHtmlWindow1->SetPage(WxMemo1->GetValue());
    WxHtmlWindow2->SetPage(WxMemo1->GetValue());
    //Something has changed so mark the text as altered in the memo
    WxMemo1->MarkDirty();
}

void Modul83Frm::OpenFile()
{
    // Check if we need to save current file before opening new
    if(DoFileSaveCheck())
    {
        //Show open file dialog
        if(WxOpenFileDialog1->ShowModal() != wxID_CANCEL)
        {
            //If we have a file name add it to the most recently used menu
            if(!MRUFile.IsEmpty())
            {
                m_fileHistory->AddFileToHistory(MRUFile);
            }
            //Load in the file
            WxMemo1->LoadFile(WxOpenFileDialog1->GetPath());
            //Save the file path in the most recently used files
            MRUFile = WxOpenFileDialog1->GetPath();
            //Update HTML preview
            UpdateHTML();
            //And reset text editor
            WxMemo1->DiscardEdits();
        }
    }
}

void Modul83Frm::SaveFile()
{
    //Set default filename and show save dialog
    WxSaveFileDialog1->SetFilename(MRUFile);
    if(WxSaveFileDialog1->ShowModal() != wxID_CANCEL)
    {
        //Save file and add filename to most recently used files
        WxMemo1->SaveFile(WxSaveFileDialog1->GetPath());
        MRUFile = WxSaveFileDialog1->GetPath();
        m_fileHistory->AddFileToHistory(WxSaveFileDialog1->GetPath());
    }
}

void Modul83Frm::NewFile()
```

```

{
//Check if we need to save the file
if(DoFileSaveCheck())
{
//Clear reset text and HTML displays
WxMemo1->Clear();
UpdateHTML();
WxMemo1->DiscardEdits();
}
}
bool Modul83Frm::DoFileSaveCheck()
{
bool RetVal = true;
//If we have unsaved changes in the text editor
if(WxMemo1->IsModified())
{
//Check if we should save the changes
if(WxMessageDialog1->ShowModal() == wxID_YES)
{
//Set a default filename and save file
WxSaveFileDialog1->SetFilename(MRUFile);
if(WxSaveFileDialog1->ShowModal() != wxID_CANCEL)
{
WxMemo1->SaveFile(WxSaveFileDialog1->GetPath());
MRUFile = WxSaveFileDialog1->GetPath();
//Add to list of most recently used files
m_fileHistory->AddFileToHistory(MRUFile);
}
else
RetVal = false;
}
}
else
{
if(!MRUFile.IsEmpty())
{
//Add to list of most recently used files
m_fileHistory->AddFileToHistory(MRUFile);
}
}
return RetVal;
}
void Modul83Frm::WrapTextInTag(const wxString& TagType)
{
//Create variables to hold position of currently selected text
long PosFrom = 0, PosTo = 0;
//Get position of currently selected text
WxMemo1->GetSelection(&PosFrom,&PosTo);
wxString TempString;
//Add tag start to temp string
TempString << wxT("<") << TagType << wxT(">");

```

```

//Add currently selected text to temp string
TempString << WxMemo1->GetStringSelection();
//Add tag end to temp string
TempString << wxT("</") << TagType << wxT(">");
//Replace currently selected text with contents of temp string
WxMemo1->Replace(PosFrom,PosTo,TempString);
//And update controls
UpdateHTML();
}
void Modul83Frm::InsertText(const wxString& Text)
{
//Create variables to hold position of cursor
long PosFrom = 0, PosTo = 0;
//Get current selection
WxMemo1->GetSelection(&PosFrom,&PosTo);
//And replace with the text to insert
WxMemo1->Replace(PosFrom,PosTo,Text);
//Then update controls
UpdateHTML();
}

```

- Untuk mengatur EventOnClick lakukan sebagai berikut
 - Beralihlah ke Modul83Frm.wxform tab.
 - Pilih menuMainBar control
 - Pada Properties tab Pilih Edit MenuItems
 - Expand the &File option and select the &New
 - Click [Edit]
 - Pada OnMenu combobox click [Create].
 - Ubh nama fungsinya menjadi MnuNewClick.
 - Click [OK].
 - Click [Apply] kemudian [OK].
 - Kemudian beralihlah ke Modul83Frm.cpp dan scroll ke bagian paling bawah, disitu kita akan menemukan fungsi baru yang barusaaja kita buat.
 - Masukkan code berikut setelah `// insert your code here`

- MnuNewClick

```
NewFile();
```

- MnuOpenClick

```
OpenFile();
```

- MnuSaveClick

```
SaveFile();
```

7.5 Hasil Percobaan

Data dan Analisis hasil percobaan

Pertanyaan

3. Jalankan code program diatas dan benahi jika menemukan kesalahan!

.....

-
4. Apakah hasil dari masing-masing nomor percobaan?

Percobaan ke-1

Percobaan ke-2.....

7.6 Analisis Hasil

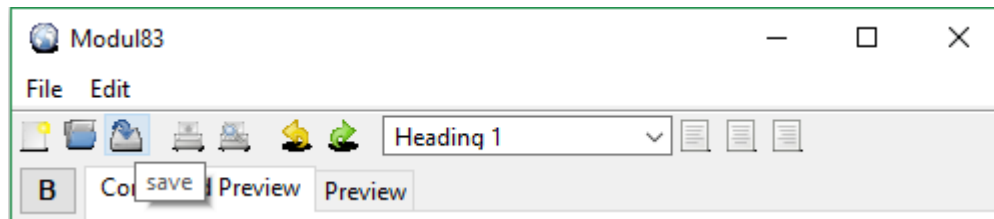
Berikan analisis terhadap masing-masing percobaan

7. Apakah fungsi dari tiap form yang ada pada percobaan?

.....

.....

8. Ketika cursor mengarah pada button, akan muncul keterangan. Jelaskan proses tersebut



.....

.....

9. Ketika cursor mengarah pada Menu, akan muncul keterangan. Jelaskan proses tersebut

.....

.....

10. Ketika kita click open pada menu, akan muncul jendela untuk memilih file yang akan kita buka. Jelaskan proses tersebut!

.....

.....

7.7 Kesimpulan

.....

7.8 Latihan

.....

7.9 Tugas (asisten)

Modul 8 : Graphical User Interface

8.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut (misalkan):

- y. 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- z. 60 menit untuk penyampaian materi
- aa. 45 menit untuk pengerjaan jurnal, tes akhir atau tugas
- bb. 50 menit **Pengayaan**

8.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

- 3. Praktikan mengenal lingkungan dan prasarana untuk membuat program dengan GUI
- 4. Praktikan mampu membuat program sederhana dengan GUI

8.3 Alat & Bahan

- 11. WxDev C++
- 12. Update TangoImages.devpak

8.4 Prosedur Praktikum

Jalankan masing-masing kode program berikut untuk masing-masing bentuk Inheritance

1. Percobaan Pertama

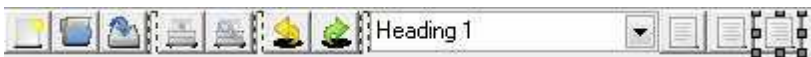
- Buat wxFrame application baru.
- Atur nama sesuai keinginan atau gunakan 'Modul83'
- Simpan dalam folder baru 'Modul83'
- Atur properties sebagai berikut:

Class Name	Modul83Frm
File Name	Modul83Frm
Title	Modul83
Author	Nama Anda
Window Style	Use Caption, Resize Border, System Menu, No Parent, Min Button, Max Button, Close Button
Height	402
Width	553
Icon	Gunakan internet-web-browser16x16.png tango icons yang diinstall dari TangoImages devpak

- Membuat Toolbar
 - Ubah component selector menjadi Toolbar.
 - Pilih toolbar component dan drop kedalam form.

- Membuat tombol pada Toolbar
 - Masukkan items berikut dari Toolbar component palette ke toolbar.
 - Ubah properties sesuai dengan keterangan.
 - Untuk bitmaps gunakan Tango Icons 16x16 folder.

Type	Tool Button
Bitmap	document-new16x16.png
Tooltip	'Create a new document'
Type	Tool Button
Bitmap	document-open16x16.png
Tooltip	'Open an existing document'
Type	Tool Button
Bitmap	document-save16x16.png
Tooltip	'Save this document'
Type	Separator
Type	Tool Button
Bitmap	document-print16x16.png
Tooltip	'Print this document'
Type	Tool Button
Bitmap	document-print-preview16x16.png
Tooltip	'Preview this document'
Type	Separator
Type	Tool Button
Bitmap	edit-undo16x16.png
Tooltip	'Undo last change'
Type	Tool Button
Bitmap	edit-redo16x16.png
Tooltip	'Redo last change'
Type	Separator
Type	ComboBox
Items	'Heading 1' 'Heading 2' 'Heading 3' 'Heading 4' 'Heading 5' 'Heading 6'
Text	'Heading 1'
Tooltip	'Alter the heading type'
Type	Tool Button
Bitmap	format-justify-left16x16.png
Tooltip	'Use inside a tag to left align text'
Type	Tool Button
Bitmap	format-justify-center16x16.png
Tooltip	'Use inside a tag to center align text'
Type	Tool Button
Bitmap	format-justify-right16x16.png
Tooltip	'Use inside a tag to right align text'

- a. 
- b. Membuat status bar
- Status bar ada pada Controls palette
 - Pilih Status bar component dan drop kedalam form.
- c. Kemudian kita perlu sizer component yang bs ditemukan di Sizer palette.
- Pilih BoxSizer component dan drop kedalam form.
 - Pastikan Orientation diatur pada wxHorizontal.
- Kemudian kita akan membuat Toolbar disebelah kiri
 - Pilih sizer yang baru saja kalian buat.
 - Pilih Panel dari palette, cari pada tab Containers.
 - Masukkan panel ke dalam sizer.
 - Atur alignment wxEXPAND menjadi true dan pastikan yang lainnya false.
 - Atur Border property menjadi 0.
 - i. Atur panel Height 278 dan Width 36.
- d. Kita akan membuat tombol sesuai kriteria berikut

Type	Tool Button
Font	Arial, Bold, size 10
Height	30
Label	B
Left	3
Tooltip	'Bold text style'
Top	0
Width	30
Type	Tool Button
Font	Times New Roman, Italic, size 11
Height	30
Label	I
Left	3
Tooltip	'Italic text style'
Top	29
Width	30
Type	Tool Button
Font	Arial, Regular, size 12, Underline
Height	30
Label	U
Left	3
Tooltip	'Underlined text style'
Top	58
Width	30
Type	Tool Button
Font	Times New Roman, Italic, size 12
Height	30
Label	F
Left	3
Tooltip	'Change font style'
Top	87

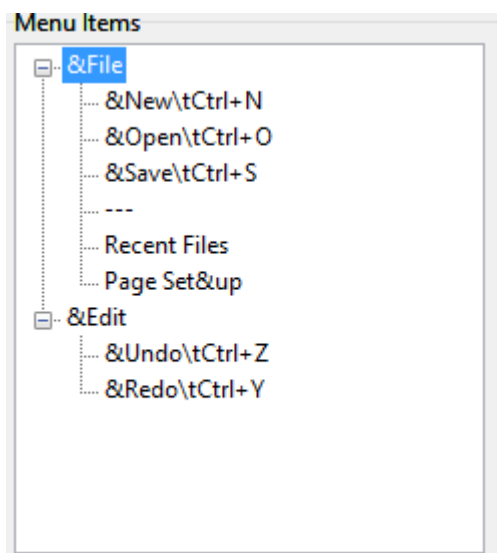
Width	30
--------------	----

- e. Kemudian kita akan membuat Notebook component yang terdiri dari 2 halaman. Halaman pertama berfungsi sebagai preview dan edit pane, sedangkan halaman kedua sebagai preview saja
 - o Pilih sizer Main pada form.
 - o Pindah ke Window palette dan pilih NoteBook.
 - o Masukkan kedalam sizer.
 - o Ubah Alignment flag wxEXPAND menjadi true, pastikan yanglain adalah false.
 - o Atur Border property 0.
 - o Atur Height 275 dan Width 460.
 - o Atur Stretch Factor 1.
- f. Lalu kita akan membuat halaman
 - o Pilih Notebook yang telah dibuat.
 - o Pilih NoteBookPage component dan masukkan ke dalam notebook.
 - o Ubah Label property menjadi 'Combined HTML Preview'.
 - o Pilih Notebook yang telah dibuat.
 - o Pilih NoteBookPage component lagi dan masukkan ke dalam notebook.
 - o Pilih dengan click pada tab, kemudian click pada halamannya.
 - o Ubah Label property menjadi 'Preview'.
- g. Setelah membuat halaman, kita perlu mengatur setting untuk mengontrolnya
 - o Pilih halaman Combine HTML Preview dengan click pada tab kemudian click pada halaman
 - o Pada Sizers palette Pilih BoxSizer.
 - o Masukkan ke dalam halaman.
 - o Pastikan sizer orientation adalah wxHORIZONTAL.
 - o Sekarang pada Window palette pilih SplitterWindow dan masukkan pada sizer.
 - o Pastikan splitter window's Alignment property adalah hanya wxEXPAND.
 - o Atur splitter's Border property menjadi 0.
 - o Atur Height 248 dan Width 195.
 - o Pastikan Orientation property adalah wxVertical.
 - o atur Stretch Factor menjadi 1.
 - o Kemudian masukkan HTML Window dari palette yang sama.
 - o Atur Height 89 dan Width 185.
 - o Pada Controls palette pilih Memo control.
 - o Masukkan control pada splitter window.
 - o Atur Height 139 dan Width 185.
 - o Gunakan Strings property untuk menghapus text yang ada.
 - o Sekarang beralih pada halaman Preview dengan click tab Preview.
 - o Kemudia click pada halamannya.
 - o Masukkan BoxSizer pada tab ini.
 - o Dari Window palette masukkan HTML Window pada sizer.
 - o Atur HTML Windows Alignment hanya wxEXPAND.
 - o Atur Border menjadi 0 dan Stretch Factor menjadi 1.
- h. Sekarang kita akan membuat MenuBar
 - o Dari Menu palette pilih MenuBar control.
 - o Masukkan kedalam area Main Form (bebas).
- i. Kemudian atur item properties pada Menu Item sebagai berikut

Type	Menu Item
Location	Root
Caption	&File
ID Name	Accept Default
	Accept Default

ID Value	
Type	Menu Item
Location Caption ID Name ID Value Hint Bitmap	Child of File &New\tCtrl+N Accept Default Accept Default 'Create a new document' document-new16x16.png
Type	Menu Item
Location Caption ID Name ID Value Hint Bitmap	Child of File &Open\tCtrl+O Accept Default Accept Default 'Open an existing document' document-open16x16.png
Type	Menu Item
Location Caption ID Name ID Value Hint Bitmap	Child of File &Save\tCtrl+S Accept Default Accept Default 'Save current document' document-save16x16.png
Type	Separator
Location ID Name ID Value	Child of File Accept Default Accept Default
Type	File History
Location ID Name ID Value	Child of File Accept Default Accept Default
Type	Menu Item
Location Caption ID Name ID Value Hint Bitmap	Child of File Page Set&Up Accept Default Accept Default 'Set up the printer' document-properties16x16.png
Type	Menu Item
Location Caption ID Name ID Value	Root &Edit Accept Default Accept Default
Type	Menu Item
Location Caption	Child of Edit &Undo\tCtrl+Z

ID Name	Accept Default
ID Value	Accept Default
Hint	'Undo
Bitmap	edit-undo16x16.png
Type	Menu Item
Location	Child of Edit
Caption	&Redo\tCtrl+Y
ID Name	Accept Default
ID Value	Accept Default
Hint	Redo
Bitmap	edit-redo16x16.png



j.

k. Langkah terakhir

- Beralih pada Dialog palette.
- Pilih OpenFileDialog.
- Masukkan pada form.
- Ubah Extensions property menjadi '*.htm;*.html'.
- Ubah Message property menjadi 'Choose a file'.
- Kemudian pilih SaveFileDialog.
- Masukkan ke dalam form.
- Ubah Extensions menjadi '*.htm;*.html'.
- Ubah Message menjadi 'Choose a file'.
- Kemudian buat MatDialog.
- Ubah Caption menjadi 'Save Changes?'.
- Ubah Message menjadi 'The contents of this file has changed\n Do you want to save the changes?'.
- Atur Message Dialog Style menjadi wxICON_EXCLAMATION, wxYES_DEFAULT dan wxYES_NO.

2. Percobaan Kedua

- Setelah fungsi CreateGUIControls pada Modul83Frm.h, tambahkan sebagai berikut Modul83Frm.h

```
public:
    void UpdateHTML();
```

```

void OpenFile();
void SaveFile();
void NewFile();
bool DoFileSaveCheck();
void MnuNewClick(wxCommandEvent& event);
void WrapTextInTag(const wxString& TagType);
void InsertText(const wxString& Text);
void MnuOpenClick(wxCommandEvent& event);
void MnuSaveClick(wxCommandEvent& event);

```

- Tambahkan code dibawah setelah `////GUI Control Declaration End`

```

wxString MRUFile;

```

- Tambahkan code dibawah pada Modul83Frm.cpp pada bagian paling bawah

```

void Modul83Frm::UpdateHTML()
{
    //Update and refresh the text on the HTML preview windows
    WxHtmlWindow1->SetPage(WxMemo1->GetValue());
    WxHtmlWindow2->SetPage(WxMemo1->GetValue());
    //Something has changed so mark the text as altered in the memo
    WxMemo1->MarkDirty();
}

void Modul83Frm::OpenFile()
{
    // Check if we need to save current file before opening new
    if(DoFileSaveCheck())
    {
        //Show open file dialog
        if(WxOpenFileDialog1->ShowModal() != wxID_CANCEL)
        {
            //If we have a file name add it to the most recently used menu
            if(!MRUFile.IsEmpty())
            {
                m_fileHistory->AddFileToHistory(MRUFile);
            }
            //Load in the file
            WxMemo1->LoadFile(WxOpenFileDialog1->GetPath());
            //Save the file path in the most recently used files
            MRUFile = WxOpenFileDialog1->GetPath();
            //Update HTML preview
            UpdateHTML();
            //And reset text editor
            WxMemo1->DiscardEdits();
        }
    }
}

void Modul83Frm::SaveFile()
{
    //Set default filename and show save dialog
    WxSaveFileDialog1->SetFilename(MRUFile);
    if(WxSaveFileDialog1->ShowModal() != wxID_CANCEL)

```

```

{
//Save file and add filename to most recently used files
WxMemo1->SaveFile(WxSaveFileDialog1->GetPath());
MRUFile = WxSaveFileDialog1->GetPath();
m_fileHistory->AddFileToHistory(WxSaveFileDialog1->GetPath());
}
}
void Modul83Frm::NewFile()
{
//Check if we need to save the file
if(DoFileSaveCheck())
{
//Clear reset text and HTML displays
WxMemo1->Clear();
UpdateHTML();
WxMemo1->DiscardEdits();
}
}
bool Modul83Frm::DoFileSaveCheck()
{
bool RetVal = true;
//If we have unsaved changes in the text editor
if(WxMemo1->IsModified())
{
//Check if we should save the changes
if(WxMessageDialog1->ShowModal() == wxID_YES)
{
//Set a default filename and save file
WxSaveFileDialog1->SetFilename(MRUFile);
if(WxSaveFileDialog1->ShowModal() != wxID_CANCEL)
{
WxMemo1->SaveFile(WxSaveFileDialog1->GetPath());
MRUFile = WxSaveFileDialog1->GetPath();
//Add to list of most recently used files
m_fileHistory->AddFileToHistory(MRUFile);
}
else
RetVal = false;
}
}
else
{
if(!MRUFile.IsEmpty())
{
//Add to list of most recently used files
m_fileHistory->AddFileToHistory(MRUFile);
}
}
return RetVal;
}
void Modul83Frm::WrapTextInTag(const wxString& TagType)

```

```

{
//Create variables to hold position of currently selected text
long PosFrom = 0, PosTo = 0;
//Get position of currently selected text
WxMemo1->GetSelection(&PosFrom,&PosTo);
wxString TempString;
//Add tag start to temp string
TempString << wxT("<") << TagType << wxT(">");
//Add currently selected text to temp string
TempString << WxMemo1->GetStringSelection();
//Add tag end to temp string
TempString << wxT("</") << TagType << wxT(">");
//Replace currently selected text with contents of temp string
WxMemo1->Replace(PosFrom,PosTo,TempString);
//And update controls
UpdateHTML();
}
void Modul83Frm::InsertText(const wxString& Text)
{
//Create variables to hold position of cursor
long PosFrom = 0, PosTo = 0;
//Get current selection
WxMemo1->GetSelection(&PosFrom,&PosTo);
//And replace with the text to insert
WxMemo1->Replace(PosFrom,PosTo,Text);
//Then update controls
UpdateHTML();
}

```

- Untuk mengatur EventOnClick lakukan sebagai berikut
 - Beralihlah ke Modul83Frm.wxform tab.
 - Pilih menuMainBar control
 - Pada Properties tab Pilih Edit MenuItems
 - Expand the &File option and select the &New
 - Click [Edit]
 - Pada OnMenu combobox click [Create].
 - Ubh nama fungsinya menjadi MnuNewClick.
 - Click [OK].
 - Click [Apply] kemudian [OK].
 - Kemudian beralihlah ke Modul83Frm.cpp dan scroll ke bagian paling bawah, disitu kita akan menemukan fungsi baru yang barusaja kita buat.
 - Masukkan code berikut setelah `// insert your code here`

- **MnuNewClick**

```
NewFile();
```

- **MnuOpenClick**

```
OpenFile();
```

- **MnuSaveClick**

```
SaveFile();
```


8.5 Hasil Percobaan

Data dan Analisis hasil percobaan

Pertanyaan

5. Jalankan code program diatas dan benahi jika menemukan kesalahan!

.....
.....

6. Apakah hasil dari masing-masing nomor percobaan?

Percobaan ke-1

Percobaan ke-2.....

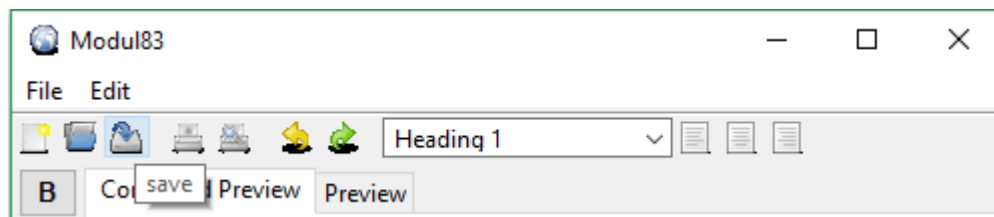
8.6 Analisis Hasil

Berikan analisis terhadap masing-masing percobaan

11. Apakah fungsi dari tiap form yang ada pada percobaan?

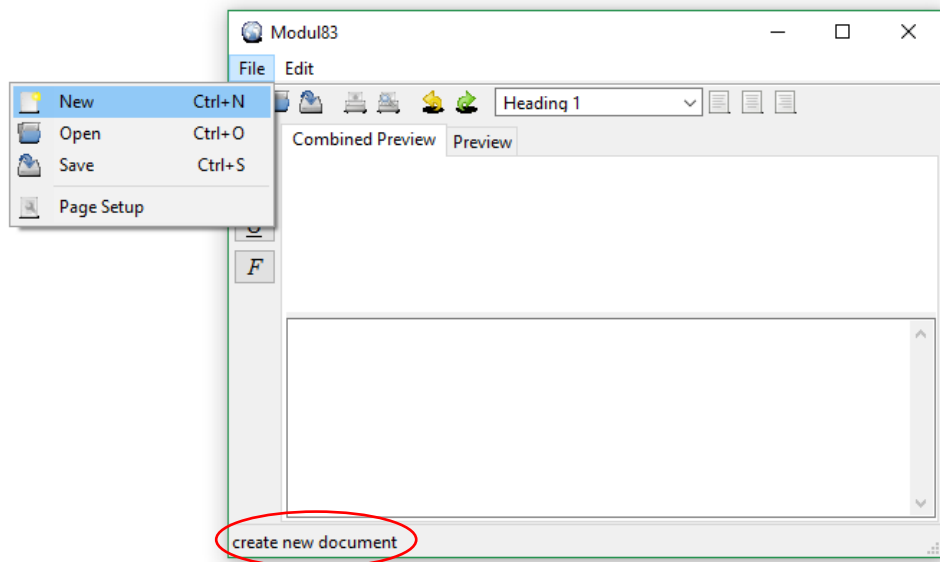
.....
.....

12. Ketika cursor mengarah pada button, akan muncul keterangan. Jelaskan proses tersebut



.....
.....

13. Ketika cursor mengarah pada Menu, akan muncul keterangan. Jelaskan proses tersebut



.....

.....

14. Ketika kita click open pada menu, akan muncul jendela untuk memilih file yang akan kita buka. Jelaskan proses tersebut!

.....

.....

8.7 Kesimpulan

.....

8.8 Latihan

.....

8.9 Tugas (asisten)

DAFTAR PUSTAKA

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, "Data Structures and Algorithms Using Java 6 edition", Wiley, USA, 2014.
- John R. Hubbard, "Scaum's Outline of Data Structures With Java second Edition", McGraw-Hill, New york, 2007.
- Robert Lafore, "Data Structures and Algorithm in Java second Edition", Sams Publishing, Indiana, 2003
- Y. Daniel Liang. 2013. Introduction to Programming with C++, 3rd Edition. Pearson Education