Kuis 1, Edunex
Kamis, 20 Maret 2025
Multiple choices and answers

1. Manakah dari pernyataan berikut yang benar mengenai kelas abstrak (abstract class) di C++?
    a. **Kelas abstrak sebaiknya digunakan ketika kita ingin memaksa kelas turunan untuk mengimplementasikan perilaku spesifik**
    b. **Kelas abstrak memungkinkan polimorfisme melalui pointer atau referensi ke kelas dasar abstrak**
    c. **Kelas abstrak harus memiliki setidaknya satu fungsi virtual murni (pure virtual function)**
    d. Fungsi virtual murni di kelas abstrak harus memiliki implementasi di kelas abstrak itu sendiri

Penjelasan
Abstract base class adalah kelas dasar untuk objek abstrak. Ia merupakan kelas yang tidak dapat diinstansiasikan. Metode tidak dapat diimplementasikan di abstract class, tetapi dideklarasikan sebagai metode pure virtual. Sehingga setidaknya abstract class harus mempunyai satu pure virtual function. Opsi a benar karena tujuan fungsi virtual adalah untuk memaksa kelas turunan melakukan implementasi metode. Opsi b benar karena konsep polimorfisme dengan pointe/reference bisa dilakukan pada kelas dasar (termasuk kelas abstrak).

2. Pernyataan yang tepat jika mobil digunakan untuk menggambarkan konsep dalam paradigma objek.
    a. **Mobil sebagai objek terdiri dari objek-objek (mesin, roda, dll) yang memiliki tanggung jawabnya masing-masing**
    b. Pengemudi mobil diharuskan mengakses dan mengontrol mesin secara langsung untuk mengoperasikan mobil dengan benar
    c. **Antarmuka mobil (setir, tombol, tuas, pedal) berperan sebagai mediator antara pengemudi dan komponen internal, sehingga menyederhanakan interaksi**
    d. Pengemudi harus mengetahui bagaimana mesin pada mobil bekerja untuk mendapat mengoperasikan mobil

Penjelasan
Opsi a benar karena semua hal adalah objek. Setiap objek mempunyai tanggung jawabnya masing-masing. Opsi b dan d salah karena pernyataan ini bertentangan dengan prinsip enkapsulasi pada paradigma objek. Enkapsulasi artinya implementasi internal disembunyikan, sehingga pengguna hanya cukup memakai interface saja.

3. Berdasarkan potongan kode di bawah ini, pilih semua pernyataan yang benar mengenai penggunaan friend class!

```
class Secret {
     private:
          int secretValue;
     public:
          Secret(int v) : secretValue(v) {}
          friend class Revealer;
);

class Revealer {
     public:
          void reveal(const Secret &s) {
               cout << "Secret value: " << s.secretValue << endl;
          }
};

int main() {
     Secret s(42);
     Revealer r;
     r.reveal(s);
     return 0;
)
```

   a. **Kelas revealer dapat mengakses semua anggota private dan protected dari class Secret**
   b. **Friendship bersifat satu arah; hanya Revealer yang mendapat akses ke Secret, bukan sebaliknya.**
   c. Deklarasi friend dalam kelas Secret secara otomatis membuat Secret juga menjadi friend class dari Revealer
   d. Friendship diwariskan kepada subclass dari Secret

Penjelasan

Friend adalah pemberian hak pada fungsi atau kelas untuk mengakses anggota non-public suatu kelas. Ia bersifat satu arah. Misal,
`class B {`
`  friend class A;`
`}`
Ini berarti bahwa seluruh member kelas A dapat mengakses anggota non-public dari kelas B.
 	
Opsi d salah karena friendship tidak diwariskan ke subclass. Sehingga turunan class B bukan merupakan friend dari class A.

4. Pilih semua pernyataan yang benar mengenai peran kelas dan objek!
   a. **Kelas adalah blueprint yang mendeskripsikan objek-objek**

b. **Objek adalah instance dari kelas yang memiliki siklus hidup (creation, manipulation, destruction) selama eksekusi program (runtime)**
c. Kelas dapat diubah secara dinamis atau dimanipulasi selama runtime layaknya objek
d. Hanya satu objek yang dapat diciptakan dari sebuah kelas saat runtime

Penjelasan
Opsi c salah karena kelas tidak dapat diubah secara dinamis saat runtime. Yang bisa dimanipulasi hanyalah objek yang dibuat dari kelas tersebut. Opsi d jelas salah karena kita dapat membuat banyak objek selama runtime.

5. Berikut ini pernyataan yang benar terkait operator assignment (=) dan copy constructor?
   a. Operator assignment membuat objek baru, sedangkan copy constructor menyalin objek yang sudah ada
   b. **Operator assignment dipanggil untuk objek yang sudah dibangun, sedangkan copy constructor dipanggil selama inisialisasi objek baru**
   c. Copy constructor dapat digunakan untuk melakukan deep copy, sedangkan operator assignment hanya dapat melakukan shallow copy
   d. **Keduanya sama-sama memiliki implementasi default jika penulis kelas tidak mengimplementasikannya**

Penjelasan

Cctor adalah konstruktor yang menciptakan objek dengan cara menduplikasi objek lain yang sudah ada. Passing parameter aktual ke parameter formal secara pass by value. Jika cctor tidak dideklarasikan oleh perancang kelas, cctor akan dilakukan secara bitwise copy. Pemberian return value dari fungsi atau metode yang nilai kembaliannya bertipe kelas tersebut.
 	
Operator assignment berbeda dengan cctor. Pada assignment `a = b`, objek a dan b sudah tercipta sebelumnya. Fungsi anggota operator assignment harus mengembalikan nilai kembali berupa objek yang sudah mengalami operasi assignment (`return *this`).
 	
Opsi c salah karena keduanya bisa digunakan untuk deep atau shallow copy, tergantung bagaimana perancang kelas mengimplementasikannya. Deep copy menyalin semua data termasuk isi yang ditunjuk oleh pointer (isi dan alamat). Shallow copy atau bitwise copy tidak membuat salinan dari data yang ditunjuk oleh pointer, hanya menyalin nilai-nilai member secara langsung. Akibatnya, pada shallow copy, dua objek bisa menunjuk ke memori yang sama.
 	
`// Contoh deep copy`
`data = new int(*other.data);`
`// Contoh shallow copy`
`data = other.data;`

Opsi d benar karena compiler bisa secara otomatis membuat versi defaultnya dengan melakukan shallow atau bitwise copy.

6. Apa tujuan dari penggunaan specifier virtual pada base class saat melakukan inheritance?
    a. **Untuk menghindari ambiguitas dalam multiple inheritance (diamond problem)**
    b. Untuk meningkatkan kinerja pemanggilan fungsi virtual
    c. **Untuk memungkinakan kelas turunan mewarisi dari beberapa kelas dasar tanpa duplikasi anggota kelas dasar yang sama**
    d. Untuk memberlakukan static binding dari pemanggilan fungsi

Penjelasan

Penggunaan specifier virtual pada base class saat inheritance berbeda dari virtual function.

```
// Virtual function
virtual void func() ;
// Virtual inheritance
class Derived : virtual public Base {}
```

Tujuan dari virtual function adalah untuk memungkinkan polimorfisme atau dynamic binding. Sementara, virtual inheritance menghindari diamond problem dalam multiple inheritance. Diamond problem dapat terjadi saat dua kelas turunan mewarisi dari base class yang sama, dan satu kelas lagi mewarisi dari keduanya, sehingga terjadi ambiguitas.

Opsi b salah, bahkan virtual function tidak meningkatkan kinerja pemanggilan fungsi. Opsi d salah karena virtual inheritance tidak ada kaitannya dengan binding fungsi, justru virtual function digunakan untuk dynamic binding.

7. Apa tujuan dari fungsi virtual di C++?
    a. Untuk mengaktifkan static binding dari pemanggilan fungsi
    b. **Untuk memungkinkan dynamic binding dari pemanggilan fungsi pada waktu runtime**
    c. **Untuk memungkinkan sebuah fungsi di-override di kelas turunan**
    d. Untuk meningkatkan kinerja panggilan fungsi

8. Diberikan hirarki kelas sebagai berikut:

```
class A {public: int x;};
class B : virtual public A {};
class C : virtual public A {};
class D : public B, public C {};
```

Manakah pernyataan berikut yang benar?

a. D punya dua copy subobjek A
b. D hanya punya satu copy subobjek A
c. Ini contoh virtual inheritance yang mencegah terjadinya diamond problem
d. Akan terjadi ambiguity error saat pengaksesan x pada D

Penjelasan
B dan C sama-sama mewarisi A secara virtual. D mewarisi B dan C. Karena pewarisan A bersifat virtual maka hanya satu subobjek A akan dimiliki oleh D. Virtual inheritance digunakan untuk mencegah diamond problem atau ambiguitas inheritance.

9. Pilih semua pernyataan yang benar mengenai penggunaan reference dalam kode ini!

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

void increment(int &ref) {
     ref++;
}
int main() {
     int x = 5;
     int &rx = x;
     increment(rx);
     cout << x << endl;
     return 0;
}
```

a. Fungsi increment menerima parameter dengan reference sehingga perubahan tidak mempengaruhi nilai asli
b. rx adalah alias dari x sehingga pemanggilan increment(rx) menaikkan nilai x
c. rx bisa dideklarasikan tanpa inisialisasi
d. Penggunaan reference pada parameter ref fungsi increment dapat digantikan dengan pointer tanpa mengubah program utama

Penjelasan

Reference variable adalah nama alias terhadap variabel tersebut. Jika sudah digunakan untuk mengacu suatu objek/variabel, reference tidak dapat direset untuk mengacu ke objek/variabel lain. Setiap pendefinisian reference variable harus selalu diinisialisasikan dengan variabel lain. Reference berbeda dengan pointer, sehingga tidak bisa digantikan fungsinya dengan pointer.
 	
Opsi d salah, karena fungsi increment dapat diubah seperti ini,
`void increment(int* ref) {`
`     (*ref)++;`

```
}
```
Tetapi program utama harus diubah seperti ini,
```
increment(&rx);
```

10. Pernyataan yang benar mengenai paradigma struktural (prosedural) dan berorientasi objek (object-oriented) berikut:
    a. Paradigma struktural mengutamakan penggunaan kembali komponen yang sudah ada dengan standarisasi komponen dasar
    b. Paradigma berorientasi objek memandang bahwa detail implementasi internal sebuah objek perlu diketahui oleh objek lain saat kedua objek tersebut akan berkomunikasi
    c. Paradigma struktural terinspirasi dari proses bisnis yang memiliki alur yang jelas, menyebabkannya kurang fleksibel terhadap perubahan
    d. Paradigma berorientasi objek memulai analisis dengan mengidentifikasi objek-objek apa saja yang ada pada domain persoalan, serta perilaku dan interaksi mereka

Penjelasan
Opsi a salah karena paradigma struktural tidak terlalu fokus pada reusability. Sementara OOP mendukung modularitas, encapsulation, inheritance, dan lain-lain. Opsi b salah karena OOP mempunyai prinsip encapsulation.

11. Cara valid untuk mendeklarasikan atau menginisialisasi objek dari kelas Queue:
    a. Queue myQueue;
    b. Queue oprQueue[10];
    c. Queue *pQueue = new Queue;
    d. Queue nsQueue = &myQueue;

Penjelasan

Opsi d salah karena &myQueue adalah pointer, sementara nsQueue bukan pointer melainkan bertipe Queue, sehingga tidak cocok.
 	
Perbedaan antara:
`Queue *pQueue = new Queue; // 1`
`Queue* nsQueue = &myQueue; // 2`
`Queue& nsQueue = myQueue; // 3`
 	
Nomor 1 membuat objek Queue yang baru secara dinamis di heap. Deklarasi seperti ini dilakukan ketika ingin melakukan kontrol manual atas siklus hidup objek. Sementara nomor 2 dan 3 tidak melakukan alokasi objek baru. Nomor 2, nsQueue merupakan pointer ke objek yang sudah ada. Dan nomor 3, nsQueue adalah reference atau nama lain dari myQueue.

12. Misalkan O1 dan O2 adalah dua objek dari kelas Person. Pilih semua pernyataan yang benar terkait dengan perbandingan kedua objek tersebut!

```
class Person {
     private:
          string name;
          int age;
     public:
          Person(string n, int a) : name(n), age(a) {}
          bool operator==(const Person &other) const {
               return (name == other.name) && (age == other.age);
          }
};

int main() {
     Person O1("Alice", 30);
     Person O2("Alice", 30);
     cout << "O1 == O2: " << (O1 == O2 ? "true" : "false") << endl;
      cout << "&O1 == &O2: " << (&O1 == &O2 ? "true" : "false") <<
endl;
     return 0;
}
```

a. Perbandingan objek O1 == O2 akan menghasilkan false
b. Dua objek akan dianggap sama dalam perbandingan &O1 == &O2
c. Membutuhkan perbandingan rekursif untuk O1 dan O2
d. O1 dan O2 adalah objek berbeda tetapi mirip

Penjelasan
Opsi a salah karena O1==O2 menghasilkan true karena operator== membandingkan nilai name dan age. Sementara &O1==&O2 membandingkan alamat memori, dan O1 serta O2 adalah dua objek yang berbeda (opsi b salah).

13. Perhatikan kelas berikut:

```
class Complex {
     public:
          double real, imag;
          Complex(double r = 0, double i = 0) : real(r), imag(i) {}
};
```

Manakah dari cara berikut yang tepat untuk meng-overload operator + untuk objek Complex?

a. Complex operator+(const Complex& other);
b. Complex& operator+(const Complex& other);
c. friend Complex operator+(const Complex& c1, const Complex& c2);
d. friend void operator+(Complex& c1, Complex& c2);

Penjelasan

Opsi a valid jika didefinisikan sebagai member function. Tetapi sebenarnya dianggap kurang tepat karena tidak mendukung komutatif dengan operan non-Complex. Opsi b salah karena operator+ harus mengembalikan objek baru, bukan reference. Opsi d salah karena operator+ harus mengembalikan nilai, bukan bertipe void.
 	
Opsi c yang paling tepat karena lebih fleksibel dan komutatif.
 	
`// Contoh implementasi opsi c`
`Complex a(2, 3);`
`Complex b = 5.0 + a; // Operan kiri diubah menjadi Complex, dan sifat komutatif bisa berlaku`

14. Kapan Anda sebaiknya memilih fungsi non-anggota (friend) daripada fungsi anggota untuk meng-overload operator?
    a. Ketika operator adalah operator unary
    b. Ketika operan kiri bukan objek dari kelas tersebut
    c. Ketika operator memodifikasi operan kiri
    d. Ketika operator adalah operator biner dan ingin dibuat komutatif, di mana salah satu operannya adalah tipe primitive

15. Seorang perancang kelas telah menyediakan pustaka dalam bentuk file header (Library.h) dan file objek (Library.o). Seorang pemakai kelas ingin membuat program di file main.cc yang menggunakan kelas dari pustaka tersebut. Pilih semua pernyataan yang benar mengenai langkah-langkah yang harus dilakukan!
    a. Pemakai kelas wajib memiliki kode sumber (.cpp) dari pustaka tersebut agar bisa melakukan linking
    b. Pemakai kelas harus menyertakan file header (Library.h) dalam main.cc untuk mendapatkan deklarasi kelas dan fungsi
    c. Pemakai kelas harus mengompilasi main.cc menjadi kode objek (main.o) sebelum proses linking
    d. Pemakai kelas harus melakukan linking antara main.o dan Library.o untuk menghasilkan executable

Penjelasan
Source file harus menyertakan file header (.h) untuk mendapatkan deklarasi kelas dan fungsi. File implementasi (.cc atau .cpp) akan dikompilasi terlebih dahulu menjadi file objek. File objek berisi binary code. Satu file implementasi menghasilkan satu file objek. File objek inilah yang digunakan untuk proses linking. File .o akan dilink dengan file .o lainnya untuk menjadi file executable (.exe). Opsi a salah karena linking tidak memerlukan file sumber (.cpp), tetapi hanya perlu file objek (.o).

16. Berdasarkan potongan kode, pilih semua pernyataan yang benar mengenai anggota statik kelas!

```
class Counter {
     public:
          static int count;
          Counter() {
               count++;
          }
          static void showCount() {
               cout << "Count: " << count << endl;
          }
};

int Counter::count = 0;

int main() {
     Counter a;
     Counter b;
     Counter c;
     Counter::showCount();
     return 0;
}
```

   a. **Variabel statik count bersifat bersama sehingga setiap objek a, b, dan c dari kelas Counter mengakses satu nilai yang sama**
   b. **Atribut statik harus didefinisikan ulang di luar definisi kelas untuk mengalokasikan memori**
   c. **Fungsi statik showCount() hanya dapat mengakses anggota statik dan tidak dapat mengakses anggota non-statik**
   d. **Nilai variabel count setelah objek b tercipta adalah 2**

Penjelasan
Anggota statik adalah anggota yang “dimiliki” oleh kelas, bukan oleh objek dari kelas tersebut. Anggota statik dapat berupa atribut maupun method. Inisialisasi harus dilakukan di luar deklarasi kelas dan di luar method.

17. Pilih semua pernyataan yang benar dalam merancang kelas di C++
   a. **Perancang kelas bertanggung jawab menentukan antarmuka (fungsi public) dan representasi internal (atribut private) dari kelas**
   b. Fungsi anggota yang didefinisikan inline pada body kelas memiliki kinerja lebih baik saat runtime
   c. **Pengguna kelas tidak dapat mengakses atribut value secara langsung**
   d. **Nama konstruktor sama dengan nama kelas**

Penjelasan

Contoh implementasi fungsi inline,

```
inline int square(int x) {
     return x * x;
}

int y = 5 * 5;
```

Fungsi inline kadang bisa memberikan kinerja lebih baik karena lebih cepat untuk fungsi yang kecil dan sering dipanggil. Namun, compiler bisa saja mengabaikan inline jika fungsinya terlalu besar atau kompleks.

18. Manakah dari pernyataan berikut tentang operator overloading di C++ yang benar?
    a. Semua operator di C++ dapat di-overload
    b. Operator overloading memungkinkan Anda untuk mendefinisikan ulang perilaku operator bawaan untuk tipe data buatan pengguna
    c. Operator overloading dapat mengubah prioritas operator
    d. Operator overloading tidak dapat mengubah aritas (jumlah operan) operator

Penjelasan

Operator overloading adalah fitur yang dapat mendefinisikan ulang perilaku operator sepertti +, -, ==, !, dll agar bisa digunakan pada objek buatan sendiri. Opsi a salah karena ada beberapa operator yang tidak bisa di-overload di C++, seperti ::, ., sizeof, dll. Opsi c salah karena prioritas tidak dapat diubah. Misalnya jika a+b*c, b*c akan selalu dilakukan terlebih dahulu. Opsi d benar, karena jumlah operan tetap harus sesuai. Contohnya operator biner + tidak bisa diubah menjadi operator unary.

19. Berdasarkan kode C++ di atas, pilih semua pernyataan yang benar mengenai kelas MyClass!

```
class MyClass {
     private:
          int* data;
          int size;
     public:
          MyClass(int s) : size(s) {
               data = new int[s];
               for (int i = 0; i < s; i++) {
                    data[i] = 0;
               }
          }
          MyClass(const MyClass &other) : size(other.size) {
               data = other.data;
          }
          void setData(int index, int value) {
```

```
            if (index >= 0 && index < size) {
                data[index] = value;
            }
        }
        int getData(int index) const {
            if (index >= 0 && index < size) {
                return data[index];
            }
            return -1;
        }
};

int main() {
    MyClass a(10);
    a.setData(0, 42);
    MyClass b = a;
    cout << b.getData(0) << endl;
    return 0;
}
```

a. **Copy constructor hanya melakukan shallow copy**
b. Tidak ada alokasi memori yang perlu dibebaskan manual
c. Tidak perlu operator assignment overloading
d. Deklarasi objek bisa dilakukan dengan MyClass obj;

Penjelasan
Opsi b salah karena ada alokasi dinamis (data = new int[s]), sehingga harus dibebaskan manual dengan delete[] data. Opsi c salah karena salah satu atribut kelas adalah pointer data (*data). Jika operator assignment tidak dideklarasikan, maka compiler secara otomatis membuatnya dengan shallow atau bitwise copy. Sementara terdapat pointer data (*data), sehingga perlu deep copy (membuat salinan baru). Opsi d salah karena dibutuhkan parameter untuk konstruktor MyClass(int s). Jika tidak ada parameter, akan error.

20. Perhatikan skenario berikut:
    - Sebuah kelas Mobil memiliki atribut warna dan metode berjalan()
    - Sebuah kelas Mesin memiliki atribut kapasitas dan metode menghasilkanTenaga()
    - Sebuah kelas MobilSport memiliki tambahan metode ngebut() di samping atribut atribut warna dan metode berjalan()

    Manakah dari pernyataan berikut yang benar menggambarkan “is-a” dan “has-a” dalam skenario ini?
    a. MobilSport “has-a” Mobil dan Mobil “is-a” Mesin

b. **MobilSport “is-a” Mobil dan Mobil “has-a” Mesin**
c. **MobilSport “is-a” Mobil dan MobilSport “has-a” Mesin**
d. Mesin “has-a” Mobil dan Mobil “is-a” MobilSport

21. Perhatikan kode berikut:

```
#include <iostream>

class GrandBase {
     public:
          virtual void show() { std::cout << "grandbase::show()\n"; }
};

class Base1 : virtual public GrandBase {
     public:
          virtual void print() { std::cout << "base1::print()\n"; }
};

class Base2 : virtual public GrandBase {
     public:
          virtual void print() { std::cout << "base2::print()\n"; }
          void show() override { std::cout << "base2::show()\n"; }
};

class Derived : public Base1, public Base2 {
     public:
           void print() override { std::cout << "derived::print()\n";
}
          void show() override { std::cout << "derived::show()\n"; }
};

int main() {
     Derived* d = new Derived(); // Line 26
     Base1& b1 = *d; // Line 27
     Base2 b2 = *d; // Line 28
     GrandBase& gb = *d; // Line 29

     b1.print(); // Line 31
     b2.print(); // Line 32
     gb.show(); // Line 33
     d->show(); // Line 34
     delete d;
     // …
}
```

Manakah output yang benar dari kode tersebut?

a. **Baris 31 akan menampilkan “derived::print()”**
b. **Baris 32 akan menampilkan “base2::print()”**
c. Baris 33 akan menampilkan “grandbase::show()”
d. **Baris 34 akan menampilkan “derived::show()”**

Penjelasan
Apabila metode kelas base ditandai dengan keyword virtual, maka implementasi metode tersebut akan dilimpahkan ke kelas turunan. Ini merupakan contoh polimorfisme. Maka dari itu opsi a dan opsi d benar. Pada opsi b, terjadi slicing. *d adalah objek bertipe Derived, namun b2 bukan reference atau pointerm melainkan objek baru bertipe Base2. Maka saat objek turunan Derived disalin ke objek base Base2, semua bagian yang bukan milik Base2 akan dibuang (sliced).

22. Manakah dari pernyataan berikut tentang pewarisan (inheritance) di C++ yang benar?
    a. **Inheritance memungkinkan sebuah kelas untuk mewarisi semua properti/atribut dan method dari kelas lain**
    b. **Anggota protected dari kelas dasar dapat diakses di kelas turunan**
    c. **Inheritance juga berfungsi untuk meningkatkan code reusability**
    d. Multiple inheritance tidak didukung di C++

23. Berdasarkan potongan kode di atas, pilih semua pernyataan yang benar mengenai konstruksi objek!

```
class Rectangle {
     private:
          int width;
          int height;
     public:
          Rectangle() : width(0), height(0) {}
          Rectangle(int w, int h) : width(0), height(h) {}
          int area() { return width * height; }
};

int main() {
     Rectangle r1;
     Rectangle r2(3, 4);
     cout << “Area of r1: “ << r1.area() << endl;
     cout << “Area of r2: “ << r2.area() << endl;
     return 0;
}
```

    a. **r1 dibuat menggunakan default constructor yang menginisialisasi width dan height menjadi 0, sehingga r1.area() menghasilkan 0**
    b. **r2 dibuat dengan user-defined constructor, sehingga area r2 dihitung sebagai 3 x 4 = 12**
    c. Saat objek r2 diciptakan, kedua konstruktornya (default dan user-defined) dijalankan secara berurutan
    d. Deklarasi objek Rectangle r1; akan tetap berhasil walaupun hanya ada user-defined constructor dalam deklarasi kelas

24. Berdasarkan potongan kode C++ di atas, pilih semua pernyataan yang benar berikut:

```
class MyClass {
     private:
          const int x;
          int y;
     public:
          MyClass(int a, int b) {
               x = a;
               y = b;
          }
          void display() const {
               cout << "x: " << x << ", y: " << y << endl;
          }
};
```

a. Kode konstruktor kelas MyClass valid
b. Assignment y = b tidak bisa dilakukan di dalam body konstruktor
c. Dibutuhkan initialization list untuk menginisialisasi anggota kelas pada konstruktor
d. Member variable y harus diinisialisasi pada initialization list

Penjelasan
Const dapat diterapkan pada atribut maupun method. Pada atribut, nilai tersebut akan tetap sepanjang waktu hidup objeknya. Pengisian nilai awal harus dilakukan saat objek tersebut diciptakan, yaitu melalui constructor initialization list.

25. Manakah dari pernyataan berikut yang benar tentang program di atas?

```
class Complex {
     public:
          double real, imag;
          Complex(double r = 0, double i = 0) : real(r), imag(i) {}
          Complex operator+(const Complex& other) const {
               return Complex(real + other.real, imag + other.imag);
          }
};

int main() {
     Complex a(1, 2), b(3, 4);
     Complex c = a + b; // Line 12
     // ...
}
```

a. c.real akan bernilai 4 dan c.imag akan bernilai 6
b. Objek asli a dan b dimodifikasi di dalam operator
c. Akan terjadi penciptaan objek Complex sebanyak tiga kali saat baris 12 dieksekusi

d. Return type dari operator+ tersebut seharusnya adalah Complex&