UAS T0044 PAGI 20xx

1. Pilihan (40 poin)

01. Ide dari *generic programming* adalah mengganti

a. static function b. inline function

**c. virtual function** d. friend function

dengan fungsi yang ada di *template parameter*.

02. *Overloading function* juga dikenal dengan nama

**a.ad hoc polymorphism** b. subtype polymorphism

c. true polymorphism d. parametric polymorphism

03. Opsi berikut yang *bukan* merupakan fitur utama dari OOP adalah

a. encapsulation b. inheritance c. polymrophism **d. template**

04. *Inheritance* kerap kali dihubungkan dengan *polymorphism*. Istilah yang paling

tepat untuk menyatakan inheritance sebagai polymorphism adalah

a.trivial polymorphism b. universal polymorphism

c. pure polymorphism **d. subtype polymorphism**

05. Meskipun mendukung penggunaan *multiple inheritance* tetapi penggunaan fitur

ini tidak dianjurkan di C++. Opsi berikut yang dianjurkan sebagai penggantinya

adalah

**a. composition**  b. template

c. polymorphism d. overloading function

06. Selain sebagai *tool* untuk membentuk *generic programming*, t*emplate* juga

merupakan *tool* untuk membentuk

a. universal polymorphism b. static polymorphism

**c. ad hoc polymorphism**  d. subtype polymorphism.

07. *Run-time polymorphism* dapat diimplementasikan menggunakan

**a. virtual function** . b. overloading function

c. friend function d. inline function

08. Paradigma-paradigma berikut yang digunakan mendukung pemrograman C++

adalah

a. procedural dan object oriented programming

b. object oriented programming dan generic programming

c. generic programming dan object oriented programming

**d. procedural, object oriented programming dan generic programming.**

09. Contoh dari deklarasi *pure virtual function* adalah

a. int fungsi (virtual int ) = 0; **b. virtual int fungsi (int) = 0;**

c. int virtual fungsi (int ) = 0; d. virtual int fungsi (int = 0);

10. Fungsi-fungsi berikut yang *binding time*-nya terjadi saat *compile time* adalah

**a. overloading function dan template function**

b. template function dan overriding virtual function

c, overriding virtual function dan overloading function

d. overloading function, template function dan overriding virtual function

11. Secara sederhana *polymorphism* dapat didefinisikan sebagai *multiple routine*

*with the same name.* Opsi berikut yang *tidak dapat digunakan* untuk

mendefinisikan polymorphism adalah

**a. satic binding** b. dynamic binding

c. generic programming d. function overriding

12. Opsi berikut yang *bukan* merupakan *karakteristik* dari fungsi virtual adalah

a. dapat di override di derived class

b. digunakan untuk mendukung polimorfisme

**c. deklarasi virtual di derived class**

d. deklarasi virtual di base class

13. Salah satu kerugian penggunaan *inheritance* di OOP adalah

a. independensi di antara class dalam hirarki

b. independensi di antara object dalam hirarki

**c. dependensi di antara class dalam hirarki**

d. dependensi di antara object dalam hirarki

14. Dua prototipe berikut : *add (const Base &)* dan *add (const Derived &)* berbeda

dalam hal:

a. scope b. protocol c. signatures **d. precedence**

15 *Tipe inheritance* yang dianjurkan untuk melakukan *derivation*  adalah

a. private b. protected

**c. public** d. private, protected dan public

16. Nilai yang dikirim ke *non type parameter* pada *class template* harus berbentuk

a. variable b. ekspresi **c. constanta** d pointer

17. Fitur OOP yang bertentangan dengan konsep *pemrograman modular* adalah

**a. encapsulation** b. inheritance

c. overloading function d. generic programming

18. *Iterator* tidak lain adalah

a. generalized reference **b. generalized pointer**

c. specialized reference d. specialized pointer.

19. Mekanisme di OOP yang memungkinkan suatu fungsi bukan anggota untuk

mengakses anggota klas adalah

a. fungsi overloading b. fungsi overriding

c. fungsi virtual **d. fungsi friend**

20. Polymorphism yang dapat diimplementasikan tanpa harus menggunakan

konsep *encapsulation* dan *inheritance*

a. virtual polymorphism b. trivial polymorphism

b. true polymorphism **d. ad hoc polymorphism**

1. Uraian (40 poin)

1.(10 poin) Selalu ada pengecualian. Fenomena ini juga berlalu di C++, karena itu

tidak semua data atau fungsi anggota diturunkan dari *base class* ke *derived class*. Uraikan mengapa *constructor* dan *destructor* tidak diwariskan di C++ meskipun mereka kita deklarasikan dengan *level akses public* atau *protected*. Uraikan juga mengapa *fungsi friend* tidak diwariskan di klas turunan. Selanjutnya, sebutkan pula sejumlah fitur lain di klas induk yang tidak diwariskan di klas turunan.

Jawab:

*Karena constructor bukan interface dari class, sementara yang diwariskan adalah interface dari class. Destructor tidak diwariskan karena satu class hanya boleh mempunyai satu destructor.*

*Friend function tidak diwariskan karena friend function membuat data yang ada di class itu dapat di akses oleh fungsi tertentu. Saat friend diwariskan, datanya akan berubah dan friend function itu harus berubah pula.*

*Fitur lain yang tidak diwariskan adalah assigment operator.*

2. (10 poin) Ilmu pengetahuan itu sifatnya kumulatif sehingga meskipun kita sudah mempunyai utilitas ADT yang tidak lain adalah enkapsulasi kita masih memerlukan *inheritance*. WHY? Untuk bisa menjawab kasus ini kamu harus membandingkan *inheritance* dengan ADT. Pertanyaan buat kamu adalah coba bandingkan *inheritance* dengan ADT dari berbagai sisi diantaranya dari sisi *komponen*, *struktur komponen*, *advantage* dan *disadvantage.*

Jawab:

|  |  |  |
| --- | --- | --- |
|  | Inheritance | ADT |
| Komponen | Constructor, copy constructor, destructor, data member and function, access modifier | Data member and function, access modifier |
| Struktur komponen | Turunan | Paralel |
| Advantage | Reusability, overriding, extendability | Independen |
| Disadvantage | Dependen | Independen |

3. (10 poin) Ada banyak tipe *polymorphism* yang kita kenal sehingga sukar bagi kita untuk membuat klasifikasi yang baku atau klasifikasi tunggal. Coba uraikan apa yang kamu ketahui tentang berbagai istilah berikut: *parametric polymorphism*, *ad hoc polymorphism*, *subtype polymorphism,* dan *override polymorphism* .

**Jawab:**

*parametric polymorphism*: *kode yang ditulis tanpa menulis secara spesifik tipenya sehingga dapat digunakan dalam berbagai tipe.*

*ad hoc polymorphism: fungsi dengan nama sama tapi parameter berbeda sehingga satu nama berarti banyak.*

*Subtype/override polymorphism: fungsi yang diturunkan tetapi isi diganti*

4. (10 poin) *Generic programming*, khususnya *template function*, yang di C++

diimplementasikan menggunakan *template* kerapkali dikontraskan dengan

*overloading function*. Melalui penggalan kode sebagai contoh, coba tunjukkan

keunggulan *template function* dibandingkan dengan *overloading function*.

Jawab:

Dengan overloading:

Void print(int a){cout<<a;}

Void print(char a){cout<<a;}

Void print(long a){cout<<a;}

Dengan template:

Template<class Type>

Void print(Type a){cout<<a;}

II. Pemrograman (20 poin)

Perguruan X adalah suatu institusi pendidikan dengan sejumlah pegawai atau karyawan. Dalam beberapa bulan terakhir ini, ada keinginan dari manajemen untuk mendesain ulang sistem informasi yang sedang berlaku. Sebagai pilot project, subsistem kepegawaian diambil untuk lebih dulu didesain ulang. Untuk mengembangkan sistem sederhana yang diinginkan, seorang program developer lulusan TI Binus di-*hire*. Pendekatan yang digunakan adalah OOD.

Berkat bantuan Prof. Djun sebagai konsultan dalam waktu yang relatif singkat, sesuai dengan kebutuhan sistem awal, developer menghasilkan tiga klas yaitu *person, employee*  dan *teacher* dimana klas *teacher* dirancang sebagai ***turunan*** dari klas *person* dan *employee*. Karena masih dalam taraf uji coba, data dan fungsi anggota untuk setiap klas dibuat sangat sederhana. Fungsi anggota untuk semua klas adalah *entryData()* dan *displayData().* Untuk data anggota hanya data dasar yang disimpan. Data anggota untuk klas person adalah *name, age* dan *sex*, klas employee dengan data anggota *department* dan *wage,* sedangkan klas *teacher*, sebagai turunan dari klas *person* dan *employee* data tambahanya adalah *teacherGrade*. Dengan demikian data angota dan fungsi anggota untuk setiap klas selengkapnya adalah sebagaui berikut:

**Klas** **Data anggota** **Fungsi anggota**

Person Name entryData()

Age displayData()

Sex

Employee Department entryData()

Wage displayData()

Teacher Name entryData()

Age displayData()

Sex

Deparement

Wage

TeacherGrade

Tugas kamu:

1. (10 poin) Gambar *class diagram* dari kasus di atas.
2. (5 poin) Tulislah kode untuk masing-masing klas. Di dalam klas, fungsi anggota hanya berbentuk ***prototype*** sedangkan definisinya ditulis di luar klas. Jangan lupa menggunakan *scope oprator ::* saat kamu menulis fungsi anggota.
3. (5 poin) Fungsi *main()* untuk menguji kode di bagian b.

Jawab:

1. Person Employee

^ ^

\ /

Teacher

1. class Person{

private:

char name[255];

int age;

char sex[7];

public:

void entryData(char[], int, char[]);

void displayData();

};

class Employee{

private:

char department[255];

unsigned int wage;

public:

void entryData(char[], unsigned int);

void displayData();

};

class Teacher: public Person, public Employee{

private:

long teacherGrade;

public:

void entryData(char[], int, char[], char[], unsigned int, long);

void displayData();

};

Person::entryData(char name[], int age, char sex[]){

strcpy(this->name, name);

this->age=age;

strcpy(this->sex, sex);

}

Person::displayData(){cout<<name<<endl<<age<<endl<<sex<<endl<<endl;}

Employee::entryData(char department[], unsigned int wage){

strcpy(this-> department, department);

this->wage=wage;

}

Employee::displayData(){cout<<department<<endl<<wage<<endl <<endl;}

Teacher::entryData(char name[], int age, char sex[], char department[], unsigned int wage, long teacherGrade){

strcpy(this->name, name);

this->age=age;

strcpy(this->sex, sex);

strcpy(this-> department, department);

this->wage=wage;

this->teacherGrade=teacherGrade;

}

Teacher::displayData(){cout<<name<<endl<<age<<endl<<sex<<endl <<department<<endl<<wage<<endl <<teacherGrade<<endl<<endl;}

1. int main(){

Teacher a;

a.entryData(“Thomas”, 25, ”Male”, “Medicine”, 5200000, 6.7);

a.displayData();

cin.get();

return 0;

}

-d1030-