UNIVERSITEIT VAN AMSTERDAM

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Tentamen

C++ programmeermethoden

Bachelor Kunstmatige Intelligentie

laatste (2e) Deeltentamen
Datum: 1 juni 2017
Tijd: 17.00-19.00

Aantal pagina's: 12 (inclusief voorblad)
Aantal vragen: 5
Maximaal aantal te behalen punten: 10

VOORDAT U BEGINT

- **Wacht** tot u de instructie krijgt het tentamen te openen.
- Controleer of uw versie van het tentamen compleet is.
- Schrijf **uw naam en studentnummer en indien van toepassing versienummer** op **elk vel papier** dat u inlevert en **nummer de pagina's**.
- U dient uw **mobiele telefoon** uit te schakelen en te bewaren in uw jas of tas.
  Uw **jas en tas** moeten onder uw tafel liggen.
- **Toegestane hulpmiddelen**: boek & notities & laptop (alleen voor lezen van ebook niet voor compileren!).

HUISHOUDELIJKE MEDEDELINGEN

- De eerste 30 minuten en de laatste 15 minuten mag u de zaal niet verlaten, ook niet voor het bezoeken van het toilet.
- Op verzoek van de examinator (of diens vertegenwoordiger) moet u zich kunnen legitimeren met een bewijs van inschrijving of een geldig legitimatiebewijs.
- Tijdens het tentamen is toiletbezoek niet toegestaan, tenzij de surveillant hier toestemming voor geeft.
- 15 minuten voor het eind wordt u gewaarschuwd dat het inlevertijdstip nadert.
- Vul indien van toepassing na afloop van het tentamen alstublieft het evaluatieformulier in.

Succes!

C++ programmeermethoden Deeltoets 2

Donderdag 1 Juni, 17:00-19:00, Science Park zaal C0.05

Bas Terwijn

Schakel de Wifi en eventuele andere communicatie-mogelijkheden (bluetooth, telefonie, mobiel internet, etc.) van uw mobiel, laptop, tablet, e-reader, etc. uit.

Gebruik uw laptop, tablet of e-reader alleen voor het lezen van uw ebook. Andere windows (bv. een command prompt of IDE) kunnen worden aangezien als fraude zoals beschreven in de 'UvA Fraude- en plagiaatsregeling'. Fraude kan leiden tot uitsluiting van deelname aan dit vak of in het uiterste geval tot beindiging van de inschijving bij opleidingen van de UvA. Kom dus niet in de verleiding en houd ook de schijn tegen.

Als een vraag onduidelijk mochten zijn, dan bent u vrij in het maken van een redelijke aanname. Vermeldt deze aanname bij uw antwoord.

Vraag 1 (2 punten)

Wat is de uitvoer van het onderstaande programma?

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

void swapA(int a, int b)
{ int temp=a; a=b; b=temp; }

void swapB(int& a, int& b)
{ int temp=a; a=b; b=temp; }

void swapC(int* a, int* b)
{ int* temp=a; a=b; b=temp; }

void swapD(int* a, int* b)
{ int temp=*a; *a=*b; *b=temp; }

void swapE(int*& a, int*& b)
{ int temp=*a; *a=*b; *b=temp; }

void swapF(int** a, int** b)
{ int* temp=*a; *a=*b; *b=temp; }

int main()
{
    int a,b;

    a=100; b=200;
    swapA( a, b );     cout<<"A: "<<a<<","<<b<<endl;

    a=100; b=200;
    swapB( a, b );     cout<<"B: "<<a<<","<<b<<endl;

    a=100; b=200;
    swapC( &a, &b );   cout<<"C: "<<a<<","<<b<<endl;

    a=100; b=200;
    swapD( &a, &b );   cout<<"D: "<<a<<","<<b<<endl;
```

```
    int *ap,*bp; // introducing pointer

    a=100; b=200; ap=&a; bp=&b;
    swapE( ap, bp );   cout<<"E: "<<a<<","<<b<<endl;

    a=100; b=200; ap=&a; bp=&b;
    swapF( &ap, &bp ); cout<<"F: "<<a<<","<<b<<endl;

    return 0;
}
```

Vraag 2 (2 punten)

- A. Wat is de uitvoer van het onderstaande programma?
- B. Verklaar de verschillen in de uitvoer aan de hand van de termen “deep copy” en “shallow copy”.
- C. Leg uit welke problemen je waar en wanneer in deze code verwacht als je aan “class A” de destructor toe voegt om het geheugen van pointer “data” weer vrij te geven?

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

class A
{
 public:
    A()
    { data=new int; *data=0;}

    // ~A() {delete data;} // destructor is commented out

    int get()
    { return *data; }

    void set(int i)
    { *data=i; }

 protected:
    int* data;
};
```

```
class B : public A
{
public:
    B() {}

    void operator=(const B& b) // assignment operator
    { data=new int; *data=*(b.data); }
};

class C : public A
{
public:
    C() {}

    void operator=(const C& c) // assignment operator
    { data=c.data; }
};

int main()
{
    B b1,b2;
    b2=b1; // calls the assignment operator
    b1.set(100);
    cout<<"B: "<<b2.get()<<endl;

    C c1,c2;
    c2=c1; // calls the assignment operator
    c1.set(100);
    cout<<"C: "<<c2.get()<<endl;

    return 0;
}
```

Vraag 3 (2 punten)

Wat is de uitvoer van het onderstaande programma?

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

class Animal
{
 public:
    virtual string says() const
    { return "generic-animal-sound";}
};

class Cat : public Animal
{
public:
    string says() const
    { return "meow";}
};
```

```
void petA(Animal animal)
{ cout<< animal.says() <<endl; }

void petB(Animal* animal)
{ cout<< animal->says() <<endl; }

void petC(const Animal* animal)
{ cout<< animal->says() <<endl; }

void petD(Animal animal)
{
    Animal* temp=&animal;
    cout<< temp->says() <<endl;
}

void petE(Animal* animal)
{
    Animal temp=*animal;
    cout<< temp.says() <<endl;
}

int main()
{
    Animal *animal=new Cat();
    cout<<"A: ";petA( *animal );
    cout<<"B: ";petB(  animal );
    cout<<"C: ";petC(  animal );
    cout<<"D: ";petD( *animal );
    cout<<"E: ";petE(  animal );
    delete animal;
    return 0;
}
```

Vraag 4 (2 punten)

Het onderstaande programma dient een 3-dimensionale array te implementeren. Alleen de methodes "get(...)" (welke een array element leest) en "set(...)" (welke een array element schrijft) zijn nog niet gedefinieerd.

- A. Geeft een correcte definitie voor deze methodes zodat de 3-dimensionale array goed werkt. Het is niet nodig om source-code over te schrijven.
- B. Verschillende definities zijn mogelijk bij vraag A. Geef 1 van de termen die in het boek worden gebruikt om te beschrijven dat een gebruiker niet op de hoogte hoeft te zijn van de precieze implementatie van een class om deze te kunnen gebruiken?

```
#include <iostream>
#include <iomanip> // for setw
using namespace std;

class My3DArray
{
public:
    My3DArray(int width,int height,int depth)
        : width(width), height(height), depth(depth)
    {
        data=new int[width*height*depth]; // allocate memory
        int i=0;
        for (int z=0;z<depth;z++)
            for (int y=0;y<height;y++)
                for (int x=0;x<width;x++)
                    set(x,y,z,i++); // set initial values
    }

    ~My3DArray()
    { delete[] data; } // free allocated memory
```

```
    void print()
    {
        for (int z=0;z<depth;z++)
        {
            for (int y=0;y<height;y++)
            {
                for (int x=0;x<width;x++)
                    cout<<setw(3)<<get(x,y,z); // get values
                cout<<endl;
            }
            cout<<"--------------------"<<endl;
        }
    }

    int get(int x,int y,int z)
    { /*** add your code here ***/ }

    void set(int x,int y,int z,int value)
    { /*** add your code here ***/ }

private:
    int width, height, depth;
    int *data;
};

int main()
{
    My3DArray my3DArray(2,3,4);
    my3DArray.print();
}
```

Vraag 5 (2 punten)

- A. Welke problemen kun je voorkomen door gebruik te maken van een “namespace”?
- B. Noem 2 belangrijke voordelen van een “vector” boven een dynamisch array.
- C. Wat stelt het keyword “friend” je in staat om te doen? en geef een voorbeeld waaruit blijkt dat dat ook echt nuttig is. Hier is niet noodzakelijk (werkende) source-code voor nodig.
- D. Wat stelt het keyword “const” je in staat om te doen met een methode en parameter? en geef een voorbeeld waaruit blijkt dat dat ook echt nuttig is. Hier is niet noodzakelijk (werkende) source-code voor nodig.

The End!