C++ programming methods

Final Exam, 26 May 2016, 13:00 – 16:00 hr
Number of questions: 7
Number of pages: 8
Total number of points: 9

The “Absolute C++” or similar books and notes are allowed during this exam. E-books and computers are not allowed. Questions may be answered in English or Dutch.

1) The name “C++” comes from the older programming language “C” and the new operator “++”. What is the output of the following program? *(1 point)*

```
#include <iostream>
using namespace std;
int main()
{
    int C=5;
    cout<<C++<<endl;
    cout<<++C<<endl;
    cout<<C<<endl;
}
```

2) In C++ there are various loops and other statements that affect the control flow. What is the output of the following program? *(1 point)*

```
#include <iostream>
using namespace std;
int main()
{
    int n=500;
    int i=0;
    while(i<1000)
    {
        for (int j=0;j<20;j++)
        {
            if (j==10) continue;
            if (j==11) break;
            for (int j=0;j<10;j++)
            {
                i+=1;
            }
        }
        cout<<i<<endl;
        if (i>=n)
            break;
    }
}
```

3) A function argument can either be call-by-value or call-by-reference. What is the output of the following program? *(1 point)*

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

int var=10;

int func3(int& a)
{
    int var=5;
    a+=var;
    var+=10;
    return a+50;
}

int func2(int a)
{
    a+=var;
    func3(a);
    return func3(a);
}

int func1(int& var)
{
    var=func2(var);
    return var+30;
}

int main()
{
    int x=0;
    cout<<func1(x)<<endl;
    cout<<x<<endl;
}
```

4) We have 10 students that each got their first 19 of all the 20 grades of the year. Implement function “findMinimalGrade()” to calculate what minimal grade each students needs to get for the 20th test to get an average grade of at least 5.5. Each of the 20 grades has a weight factor which determines its influence on the average grade in the usual way (weighted average).

The resulting minimal 20th grade can be larger than the highest possible grade of 10 (for students that can't reach the 5.5 grade average) or lower than the lowest possible grade of 1 and even negative (for students that have already reached the 5.5 grade average). *(2 points)*

```
#include <iostream>
#include <stdlib.h> // for rand()
using namespace std;

const int students=10; // the number of students
const int grades=20;   // the number of grades for each student

// initializes grades and weights
void init(double studentGrades[][grades],
          double weights[grades])
{
    for (int s=0;s<students;s++)     // for each student
        for (int g=0;g<grades-1;g++) // set the first 19 grades
            studentGrades[s][g]=1+9*(rand()/static_cast<double>(RAND_MAX));

    for (int s=0;s<students;s++)      // for each student
        studentGrades[s][grades-1]=0; // set the 20th grade to zero

    for (int g=0;g<grades;g++) // set grade weights between 1.0 and 5.0
        weights[g]=1+4*(rand()/static_cast<double>(RAND_MAX));
}

// computes what the 20th grade should be to get a final grade of 5.5
// stores the result for each student in minimalGrade
void findMinimalGrade(const double studentGrades[][grades],
                      const double weights[grades],
                      double minimalGrade[])
{
  // . . . . your implementation goes here . . . .
}

int main()
{
    double studentGrades[students][grades]; // holds the grades for each student
    double weights[grades]; // holds the weight associated with each grade
    init(studentGrades,weights); // initializes grades and weights

    double minimalGrade[students]; // will hold the minimalGrade result
    findMinimalGrade(studentGrades,weights,minimalGrade);

    for (int s=0;s<students;s++)
        cout<<"student "<<s<<" minimalGrade:"<<minimalGrade[s]<<endl;
}
```

5) When an object of a class is instantiated its constructor is called. List for each of the 8 numbered print statements in function “main()” what it prints or if and **why** it produces a compile-time error. If it produces an error assume it is commented out so that the program can be run anyway. *(1 point)*

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

class A
{
public:

    A()
    { count++;} // counts how many times the constructor of A is called

    int getCount1()
    { return count;}

    int getCount2() const
    { return count;}

private:
    static int count;
};
int A::count=0; // initialize count outside of class because it is static

class B
{
public:
    const A& getA1();
          A  getA2();

private:
    A a;
};

const A& B::getA1()
{ return a;}

A B::getA2()
{ return a;}

int main()
{
    A a;
    cout<<a.getCount1()<<endl;         // 1
    cout<<a.getCount2()<<endl;         // 2

    B b;
    cout<<b.a.getCount1()<<endl;       // 3
    cout<<b.a.getCount2()<<endl;       // 4

    cout<<b.getA1().getCount1()<<endl; // 5
    cout<<b.getA1().getCount2()<<endl; // 6

    cout<<b.getA2().getCount1()<<endl; // 7
    cout<<b.getA2().getCount2()<<endl; // 8
}
```

6) Pointers are an essential part of the C++ language. What is the output of the following program? *(1.5 points)*

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

void func1()
{
    int i1=100;
    int i2=200;
    int* p1=&i1;
    int* p2=&i2;
    p1=p2;
    *p1+=100;
    cout<<"i1:"<<i1<<endl;
    cout<<"i2:"<<i2<<endl;
}

void helper(int** pp1,int** pp2) // some helper function
{
    int* temp=*pp1;
    *pp1=*pp2;
    *pp2=temp;
}

void func2()
{
    int i1=100;
    int i2=200;
    int* p1=&i1;
    int* p2=&i2;
    helper(&p1,&p2);
    cout<<"p1:"<<*p1<<endl;
    cout<<"p2:"<<*p2<<endl;
}

// second part of 2 parts on next page
```

```
// first part of 2 parts on previous page

class A
{
public:
    A(int d)
    { this->d=d;}

    void add(A* a)
    { n=a;}

    A* next()
    { return n;}

    int data()
    { return d;}

private:
    int d;
    A* n;
};

void func3()
{
    A* s=NULL;
    for (int i=0;i<5;i++)
    {
        A* a=new A(i*100);
        a->add(s);
        s=a;
    }
    A* p=s;
    p=p->next();
    cout<<"p->data():"<<p->data()<<endl;
    p=p->next();
    cout<<"p->data():"<<p->data()<<endl;
    // skipping using 'delete' to free memory
}

int main()
{
    func1();
    func2();
    func3();
}
```

7) A beginner C++ programmer created the following program where the sound of each animal in a randomized array gets printed. To the programmers surprise the printAnimals() function currently only prints the "generic-animal-sound" string instead of the specific sound (“meow” for Cat, etc.) for each animal. Correct the program so that the specific sound for each animal is printed (hint: to accomplish this three different concepts need to be addressed).

You still are only allowed to use some kind of array with all animals for printing in the printAnimals() function. You don't have to copy all lines of the original program if you clearly indicate the parts that you have changed. *(1.5 points)*

```
#include <stdlib.h> // for srand() and rand()
#include <iostream>
using namespace std;

const int n=10;

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

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

class Mice : public Animal
{
public:
    string says()
    { return "squeak";}
};

class Duck : public Animal
{
public:
    string says()
    { return "quack";}
};

void printAnimals(Animal animals[])
{
    for(int i=0;i<n;i++)
        cout<<animals[i].says()<<endl;
}

int main()
{
    srand(time(NULL)); // seed the random generator using current time
    Animal animals[n];

    for(int i=0;i<n;i++)
    {
        int a=rand()%3;
        switch (a)
        {
        case 0:  animals[i]=Cat();break;
        case 1:  animals[i]=Mice();break;
        default: animals[i]=Duck();
        }
    }

    printAnimals(animals);
}
```