# Vraag 1

Welk van de antwoorden is waar voor de volgende regel C of C++ broncode: (2)

**x = 1 / x;**

1. Deze regel test of **x** gelijk is aan **1/x**
2. Deze regel vervangt de inhoud van variabele **x** door **1** gedeeld door de oude inhoud
3. Deze vergelijking geldt alleen als **x** gelijk is aan **1**
4. Deze regel code is fout

# Vraag 2

Welk van de volgende antwoorden is waar voor de volgende regel C of C++ broncode: (2)

**bool betweenOneAndTen = (0 > x && x > 10);**

1. Boolean variabele **betweenOneAndTen** krijgt altijd de waarde **true**
2. De benaming **betweenOneAndTen**klopt niet met het gebruik van deze variabele
3. De expressie rechts van de **=** moet zijn **(x < 0 && 10 < x)**
4. Het moet zijn **==** in plaats van **=**

# Vraag 3

De statement **show (x)** drukt **x** af op het scherm, gevolgd door een spatie. Wat wordt getoond door het volgende C of C++ broncode fragment: (3)

**for (int index = 2; index < 16; index \*= 2) {**

**show (index);**

**}**

Wat wordt afgedrukt: (3)

1. 2 4 6 8 10 12 14
2. 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
3. 2 4 8
4. 2 3 4 5 6 7 8 9 10 11 12 13 14 15

# Vraag 4

Wat is de waarde van **j**  na het volgende C of C++ broncode fragment: (1)

**int i = 3;**

**int j = i++;**

1. 3
2. 4
3. 5
4. 0

# Vraag 5

Functie **askQuestion ()** stelt een vraag en function **answerIsOk ()** geeft een boolean waarde terug die aangeeft of het antwoord op de laatst gestelde vraag goed is. Welk C of C++ broncode fragment leidt er toe dat de vraag gesteld wordt tot 't antwoord goed is: (3)

1. **while (!answerIsOk) {**

**askQuestion ();**

**}**

2. **while (anwerIsOk) {**

**askQuestion ();**

**}**

3. **do {**

**askQuestion ();**

**} while (!answerIsOk ());**

4. **do {**

**askQuestion ();**

**} while (answerIsOk ());**

# Vraag 6

Wat is de waarde van i na het volgende C++ broncode fragment: (4)

**int i = 3;**

**int &j = i;**

**j++;**

**i++;**

**j++;**

1. 3
2. 4
3. 5
4. 6

## Vraag 7

Wat is de waarde van **x** na het volgende code fragment: (1)

**int square (int aNumber) {**

**aNumber = aNumber \* aNumber;**

**return aNumber;**

**}**

**int x = 10;**

**square (x);**

**x = square (x);**

1. 100
2. 10
3. 0
4. 10000

# Vraag 8

Wat is er mis met de volgende regel C of C++ broncode: (2)

**extern int Int = 3;**

1. De naam **Int** mag niet, omdat het een keyword is
2. Deze statement mag **Int** geen waarde geven omdat het geen definitie van **Int** is maar alleen een declaratie
3. Het woord **extern** mag alleen in een header file voorkomen
4. Het moet zijn **EXTERN**, omdat dit een macro is

# Vraag 9

De aanroep **show (x)** toont **x** op het beeldscherm, gevolgd door een spatie. Wat is er mis met het volgende C of C++ broncode fragment: (1)

**switch (aFloat) {**

**case 3.141532653588: {**

**show ("This is the number pi");**

**break;**

**}**

**case 2.718281828459: {**

**show ("This is the number e");**

**break;**

**}**

**default: {**

**show ("This is just a number");**

**break;**

**}**

**}**

1. Case labels mogen geen floats zijn. Dit zou ook niet goed kunnen, immers door afrondingsverschillen is het checken van exacte gelijkheid van floating point getallen meestal zinloos.
2. De eerste regel moet zijn **switch (int float) {**
3. Je mag in de cases van een switch geen **{}** gebruiken
4. De **break** statement mag alleen worden gebruikt om uit een lus te springen

# Vraag 10

Als je in plaats van een globale variabele het adres van die globale variabele aan een functie doorgeeft: (2)

1. Kun je die globale variabele in de functie een ander adres geven
2. Kun je die globale variabele in de functie een andere waarde geven
3. Werk je in de functie met een lokale kopie van die globale variabele en kun je hem dus niet veranderen
4. Moet je niet vergeten **delete** aan te roepen, anders heb je een geheugenlek

# Vraag 11

Wat is er mis met de volgende C of C++ code fragment: (1)

**struct EmployeeRecord {**

**char firstName [64];**

**char lastName [64];**

**char employeeNumber [16];**

**};**

**EmployeeRecord employeeRecord;**

1. Niets
2. Het attribuut employeeNumber moet een **long** of een **int** zijn
3. Het moet geen **char** maar **string** zijn
4. Het moet geen **struct** maar **class** zijn

# Vraag 12

Wat is het effect van het volgende code fragment: (1)

**class Vector {**

**public:**

**float x, y, z;**

**Vector add (Vector v1, Vector v2);**

**};**

1. Er wordt een object type gedefinieerd, maar nog geen objecten van dat type
2. Er wordt een object gecreëerd
3. Er worden twee vectoren opgeteld
4. Er wordt geheugenruimte gereserveerd voor drie floating point getallen

# Vraag 13

Class **House** en class **School** erven beide van class **Building**. Welk van de volgende statements is correct: (1)

1. **Building \*building = new School;**

2.  **School \*School = new Building;**

3. **House \*house= new School;**

4. **School \*school = &new School;**

# Vraag 14

Wat is in de eerste plaats het nut van stabiele dunne interfaces in broncode: (2)

1. Executie van de code wordt er sneller van
2. Veranderde lokale ontwerpbeslissingen hebben minder impact op de code als geheel
3. Dit spaart geheugenruimte
4. Software is gemakkelijker te beveiligen

# Vraag 15

De aanroep **show (x)** toont **x** op het beeldscherm, gevolgd door een spatie.

Wat wordt afgedrukt door het volgende programma: (4)

**class A {**

**public:**

**void f ();**

**virtual void g ();**

**};**

**class B: public A {**

**public:**

**void f ();**

**void g ();**

**};**

**void A::f () {show (3);}**

**void A::g () {show (4);}**

**void B::f () {show (5);}**

**void B::g () {show (6);}**

**A \*a = new A, \*b = new B;**

**a.f ();**

**a.g ();**

**b.f ();**

**b.g ();**

1. 3 4 5 6
2. 3 4 3 4
3. 5 6 5 6
4. 3 4 3 6

# Vraag 16

Welk van de volgende uitspraken is waar: (2)

1. Meervoudige overerving is dat twee classes van de zelfde class erven
2. Lokale variabelen staan op de stack
3. Een functie mag zichzelf niet aanroepen
4. Automatic variabelen bevinden zich op de heap

# Vraag 17

Welk van de volgene uitspraken is onwaar: (3)

1. Een class is abstract als hij minimaal 1 pure virtual function heeft
2. Destructors worden aangeroepen als het geheugen van het bijbehorende object wordt vrijgegeven
3. Elke class kan slechts 1 constructor hebben
4. Macros zijn soms handig

# Vraag 18

Welk van onderstaande uitspraken is waar, gegeven de volgende C++ broncode, waarin alle namen zinnig zijn (average betekent gemiddeld): (2)

**float numbers [] = {3, 5, 5, 1, 2, 8};**

**lowestNumber (numbers) = average (numbers);**

1. Dit is onzincode, een functieaanroep kan nooit links van het **=** teken staan
2. Functie **lowestNumber** geeft blijkbaar een reference terug, en de waarde van array **numbers** is na afloop **{3, 5, 5, 4, 2, 8}**, gezien de naam van de functie
3. Deze code werkt niet, er had een constante tussen de **[]** moeten staan, namelijk **6**
4. De **{}** geven aan dat het om een set (verzameling) gaat

# Vraag 19

Wat is het primaire nut van header files? (1)

1. Header files maken cross-module typechecking mogelijk en voorkomen daarmee linkage errors
2. In headers kun je met **#define** constanten definieren, dat is beter dan gebruik van het keyword **const**
3. Headers zijn bedoeld om alle variabelen op een centrale plek te definieren, zodat je ze in elke .cpp file kunt gebruiken
4. Headers zijn nodig om object georienteerd te kunnen programmeren, want classes mogen alleen in een header gedeclareerd worden

# Vraag 20

Aan correct werkende broncode wordt een constructor toegevoegd met als prototype:

**MyClass (int i);**

Hierna blijkt de volgende regel opeens een foutmelding te geven:

**MyClass myObject;**

Wat is hiervan een mogelijke oorzaak: (3)

1. Er had moeten staan **MyClass myObject = new MyClass;**

2. Er had moeten staan **MyClass \*myObject = new MyClass ();**

3. Er had tevens een parameterloze constructor toegevoegd moeten worden, omdat de default constructor niet meer beschikbaar is door toevoeging van de nieuwe constructor

4. Er had ook een bijpassende destructor toegevoegd moeten worden **~MyClass (int i);**