Inleiding programmeren (C++)

Inleiding Programmeren (C++)	
Richting	Informatica
Jaar	1BINF

Bespreking

Voor 2011 werd dit vak in de programmeertaal Oberon gegeven.

Dit vak wordt door de speelse professor Goethals gegeven. Zijn lessen zijn dus heel ludiek en over het algemeen een plezier om te volgen maar hier mag je je niet door laten afleiden van de moeilijkheidsgraad van het vak. Er zijn mensen die al wat konden programmeren en die zal het in het begin iets gemakkelijker afgaan. Echter zelfs zij moeten opletten, want programmeren wordt hier op een andere, formelere manier bekeken als de doorsnee hobbyist dit doet.

Zorg zeker dat je dit vaak slaagt of je geraakt in de problemen voor de rest van je Bachelor aangezien je nog heel wat zult moeten programmeren waarbij ze verwachten dat je dit al perfect kan. Maak er voor jezelf een uitdaging van en je zult merken dat het des te interessanter en leuker wordt.

Theorie

Zoals hierboven gezegd leuk gegeven maar belangrijke theorie. Je examen voor dit deel zal uit een 5-tal vragen bestaan die nakijken of je wel weet hoe een programma ineen zit en wat de filosofie erachter een beetje is.

Praktijk

Op het examen zal je al je vaardigheden moeten combineren in één groter programma dat verschillende dingen zal moeten kunnen. Leer vlotjes met functies omgaan en de verschillende datatypes (de subtiliteiten van lijsten etc). Ook is het mogelijk dat je een bestand moet kunnen inlezen dus zorg dat je hiermee geen tijd verspilt en bereid je grondig voor.

Naast je examen zijn er ook de opdrachtjes doorheen het jaar. Deze staan op de helft van je praktijkpunten en zijn je hoofdmethode om het vak in te werken/studeren. Zeker de moeite waard om hiermee serieus om te gaan zodat je des te minder werk hebt tegen het examen.

Puntenverdeling

De punten van het vak Inleiding programmeren worden de punten van het vak bepaald door:

```
   Het resultaat voor Python (op 10)
       een deelexamen voor Python (3 punten);
       een praktijkexamen voor Python (7 punten);
   het resultaat voor C++ (op 10)
       een project voor C++ (3 punten);
       een praktijkexamen voor C++ (7 punten)
   Indien je slaagt voor beide onderdelen (>=5) worden beide delen bij elkaar
geteld, in het andere geval kan je maximaal 7/20 behalen.
```

studieprogramma bachelor informatica

Examenvragen

Academiejaar 2019 - 2020 - 2de zittijd

C++ en theorie: Media:IP_augustus_1920.pdf

website.cpp:

```
class user;

class message {
    string content;
    message* to;
    user* sender;

public:
    message(string c, user* sender, message* response_to = nullptr): content(c),
sender(sender), to(response_to) {}

    const string &getContent() const {
        return content;
    }

    string to_string();

    void setContent(const string &content) {
        message::content = content;
    }
};

class user {
    string name;

private:
    vector<user*> friends;
    vector<message*> messages;

public:
    user(string n): name(n) {}

    const string &getName() const {
        return name;
    }

    void add_friend(user p) {
        friends.push_back(&p);
        p.friends.push_back(this);
    }

    vector<user*> get_friends() {
        return friends;
    }

    message* post(string m,message* to=nullptr) {
        message* mess=new message(m,this,to);
        messages.push_back(mess);
        return mess;
    }

    message* mostRecent() {
        if (messages.size()==0) return nullptr;
        else return messages[messages.size()-1];
    }
```

```
    void print_wall() {
        cout << "Wall of " << getName() << endl;
        for (auto s:messages) {
            cout << s->to_string() << endl;
        }
        cout << "==================================================" << endl;
    }
};

string message::to_string() {
    string s=sender->getName()+" : "+this->content;
    if (to!=nullptr) {
        s+=" (responding to "+to->sender->getName()+")";
    }
    return s;
}


int main() {
    user len("Len");
    user toon("Toon");
    user stephen("Stephen");
    user daphne("Daphne");

    len.add_friend(toon);
    len.add_friend(stephen);
    toon.add_friend(daphne);

    cout << "Friends of Len: ";
    for (auto x:len.get_friends()) {
        cout << x->getName() << " ";
    }

    cout << endl << "Friends of Toon: ";
    for (auto x:toon.get_friends()) {
        cout << x->getName() << " ";
    }

    toon.post("Veel succes bij het examen morgen!");
    len.post("Wie gaat surveilleren?",toon.mostRecent());
    cout << endl << len.mostRecent()->to_string()<<endl;

    toon.print_wall();

    return 0;
}
```

Academiejaar 2018 - 2019

Python herkansing, C++ en Theorie: Media:IP_januari_1819.pdf

sudoku.py:

```
def maak_sudoku():
    """
        Maak nieuwe sudoku aan met op alle posities None
    """
    sudoku = []
    rij = [None, None, None, None, None, None, None, None, None]
    for i in range(9):
        sudoku.append(rij)
    return sudoku


def dubbels(rij):
    """
        Geef True terug als een element twee keer voorkomt in 'rij'
    """
    for e in range(len(rij)):
        if rij[e] in rij:
            return True
    return False


def geldig(sudoku):
    """
        Kijkt na of de gegeven sudoku geldig is
    """
    for i in range(9):
        kolom = []
        for j in range(9):
            kolom.append(sudoku[j][i])
        rij = sudoku[i]
        if (dubbels(rij) and dubbels(kolom)):
            return False

    # TODO in vraag 3.2: Code om de 9 blokken na te kijken op duplicaten

    return True




def print_sudoku(s):
    """
        Help functie om een sudoku mooi uit te printen
    """
    print("+-----------------+-----------------+-----------------+")
    for x in range(9):
        print("|", end="")
        for y in range(9):
            if s[x][y] is not None:
                print(" ", s[x][y], " ", end="")
            else:
                print("  _  ", end="")
            if ((y+1)%3) != 0:
                print(" ", end="")
            else:
                print("|", end="")
```

```
        print()
        if ((x+1)%3 == 0):
            print("+-----------------+-----------------+-----------------+")
        else:
            print("|                 |                 |                 |")


if __name__ == "__main__":
    # Begin debugging
    s1 = maak_sudoku()
    print_sudoku(s1)

    s1[0][0] = 5
    s1[1][2] = 3
    s1[7][7] = 2
    s1[8][8] = 4

    print(geldig(s1))
    s1[0][1] = 5
    print(geldig(s1))

    print("Ben je tot hier geraakt en zijn de vorige lijnen true en false?")
    # EINDE debugging

    # Begin uitbreiden
    s2 = [[1, None, None, None, None, None, None, None, 6],
      [None, 1, 6, None, 2, None, 7, None, None],
      [7, 8, 9, 4, 5, None, 1, None, 3],
      [None, None, None, 8, None, 7, None, None, 4],
      [None, None, None, None, 3, None, None, None, None],
      [None, 9, None, None, None, 4, 2, None, 1],
      [3, None, 2, 9, 7, None, None, 4, None],
      [None, 4, None, None, 1, 2, None, 7, 8],
      [9, None, 8, None, None, None, None, None, None]]
    print_sudoku(s2)
    print(geldig(s2))
    # geldig(s2) geeft foutief "true" het blokje links bovenaan(vierkantje
opgespannen door[0][0] en[2][2]) bevat twee maal het cijfer 1
    # breid geldig uit zodat ook de vierkantjes nagekeken worden

    # EINDE code uitbreiden

    # Begin nieuwe functionaliteit
    s3 = [[1, None, None, None, None, None, None, None, 6],
      [None, None, 6, None, 2, None, 7, None, None],
      [7, 8, 9, 4, 5, None, 1, None, 3],
      [None, None, None, 8, None, 7, None, None, 4],
      [None, None, None, None, 3, None, None, None, None],
      [None, 9, None, None, None, 4, 2, None, 1],
      [3, 1, 2, 9, 7, None, None, 4, None],
      [None, 4, None, None, 1, 2, None, 7, 8],
      [9, None, 8, None, None, None, None, None, None]]

    # print(mogelijk(s3, 1, 1)) # Geeft: [3, 5]
    # print(mogelijk(s3, 4, 5)) # Geeft: [1, 5, 6, 9]
    # print(mogelijk(s3, 8, 8)) # Geeft: [2, 5]
```

```
    s4 = [[None, None, None, None, None, None, None, None, 6],
      [None, None, 6, None, 2, None, 7, None, None],
      [7, 8, 9, 4, 5, None, 1, None, 3],
      [None, None, None, 8, None, 7, None, None, 4],
      [None, None, None, None, 3, None, None, None, None],
      [None, 9, None, None, None, 4, 2, None, 1],
      [3, 1, 2, 9, 7, None, None, 4, None],
      [None, 4, None, None, 1, 2, None, 7, 8],
      [9, None, 8, None, None, None, None, None, None]]
    print_sudoku(s4)
    # vul_aan(s4)
    print_sudoku(s4)
```

frame.cpp:

```
#include <iostream>
#include <vector>

using namespace std;

typedef vector<vector<int>> Sudoku;     // Een Sudoku is een 9x9 matrix die we
voorstellen als een vector van vectoren
const int LEEG=0;                       // We gebruiken constante "LEEG" om een
cel aan te duiden waar nog geen waarde in staat


////////////////////////////////////////////////////
// Declaratie van enkele hulpfuncties
// Definitie volgt na main
////////////////////////////////////////////////////
ostream& operator<<(ostream& s, const vector<int> &v);    // Print een vector uit
via cout << v;
void print(Sudoku s);                                     // Nette weergave van
een sudoku


// Initialiseer een nieuwe Sudoku door hem helemaal te vullen met 'LEEG'
void init(Sudoku s) {
    vector<int> rij {LEEG,LEEG,LEEG,LEEG,LEEG,LEEG,LEEG,LEEG,LEEG};
    for (int x=0;x<9;x++) {
        s.push_back(rij);
    }
}

// Geeft true als vector v twee keer dezelfde waarde bevat.
// LET OP! v mag enkel getallen 1-9 bevatten.
bool dubbels(vector<int> v) {
    bool present[9] = {false, false, false, false, false, false, false, false,
false};
    for (int i=0;i<v.size();i++) {
        if (present[v[i]-1]) return true;
        present[v[i]-1]=true;
    }
    return false;
}

// Kijk na of s een geldige (deels ingevulde) sudoku is
bool geldig(Sudoku s) {
    for (int i = 0; i < 9; i++) { // kijk na of er dubbels zijn in rij i of kolom
i
        vector<int> kolom;
        for (int j=0;j<9;j++) {
            kolom.push_back(s[j][i]);
        }
        vector<int> &rij=s[i];
        if (dubbels(rij) || dubbels(kolom)) {
            return false;
        }
    }

    // TODO in vraag 3.2: Code om de 9 blokken na te kijken op duplicaten
```

```
    return false;
}


int main() {
    cout << boolalpha;
// Begin debugging
    Sudoku s1;
    init(s1);
    print(s1);

    s1[0][0]=5;
    s1[1][2]=3;
    s1[7][7]=2;
    s1[8][8]=4;

    cout << geldig(s1);
    s1[0][1]=5;
    cout << geldig(s1) << endl;
    cout << "Ben je tot hier geraakt en is de vorige lijn truefalse?" << endl;
// EINDE debugging
// Begin uitbreiden
    Sudoku s2={{1,LEEG,LEEG,LEEG,LEEG,LEEG,LEEG,LEEG,6},
               {LEEG,1,6,LEEG,2,LEEG,7,LEEG,LEEG},
               {7,8,9,4,5,LEEG,1,LEEG,3},
               {LEEG,LEEG,LEEG,8,LEEG,7,LEEG,LEEG,4},
               {LEEG,LEEG,LEEG,LEEG,3,LEEG,LEEG,LEEG,LEEG},
               {LEEG,9,LEEG,LEEG,LEEG,4,2,LEEG,1},
               {3,LEEG,2,9,7,LEEG,LEEG,4,LEEG},
               {LEEG,4,LEEG,LEEG,1,2,LEEG,7,8},
               {9,LEEG,8,LEEG,LEEG,LEEG,LEEG,LEEG,LEEG}};
    print(s2);
    // geldig(s2) geeft foutief "true"; het blokje links bovenaan (vierkantje
opgespannen door [0][0] en [2][2]) bevat twee maal het cijfer 1
    // breid geldig uit zodat ook de vierkantjes nagekeken worden

// EINDE code uitbreiden

// Begin nieuwe functionaliteit
    Sudoku s3={{1,LEEG,LEEG,LEEG,LEEG,LEEG,LEEG,LEEG,6},
               {LEEG,LEEG,6,LEEG,2,LEEG,7,LEEG,LEEG},
               {7,8,9,4,5,LEEG,1,LEEG,3},
               {LEEG,LEEG,LEEG,8,LEEG,7,LEEG,LEEG,4},
               {LEEG,LEEG,LEEG,LEEG,3,LEEG,LEEG,LEEG,LEEG},
               {LEEG,9,LEEG,LEEG,LEEG,4,2,LEEG,1},
               {3,1,2,9,7,LEEG,LEEG,4,LEEG},
               {LEEG,4,LEEG,LEEG,1,2,LEEG,7,8},
               {9,LEEG,8,LEEG,LEEG,LEEG,LEEG,LEEG,LEEG}};

    // cout << mogelijk(s3,1,1) << endl;  // Geeft: [ 3 5 ]
    // cout << mogelijk(s3,4,5) << endl;  // Geeft: [ 1 5 6 9 ]
    // cout << mogelijk(s3,8,8) << endl;  // Geeft: [ 2 5 ]

    Sudoku s4={{LEEG,LEEG,LEEG,LEEG,LEEG,LEEG,LEEG,LEEG,6},
               {LEEG,LEEG,6,LEEG,2,LEEG,7,LEEG,LEEG},
```

```
                {7,8,9,4,5,LEEG,1,LEEG,3},
                {LEEG,LEEG,LEEG,8,LEEG,7,LEEG,LEEG,4},
                {LEEG,LEEG,LEEG,LEEG,3,LEEG,LEEG,LEEG,LEEG},
                {LEEG,9,LEEG,LEEG,LEEG,4,2,LEEG,1},
                {3,1,2,9,7,LEEG,LEEG,4,LEEG},
                {LEEG,4,LEEG,LEEG,1,2,LEEG,7,8},
                {9,LEEG,8,LEEG,LEEG,LEEG,LEEG,LEEG,LEEG}};
    print(s4);
    //vul_aan(s4);
    print(s4);
//EINDE nieuwe functionaliteit

    return 0;
}


// Definitie van de help functies

// Volgende functie zorgt ervoor dat je een vector<int>
// kan weergeven via cout. Dus: als v een vector<int> is,
// kan je gewoon schrijven cout << v << endl;
ostream& operator<<(ostream& s, const vector<int> &v) {
    s<< "[ ";
    for (int x:v) {
        s << x << " ";
    }
    s<<"]";
    return s;
}

void print(Sudoku s) {
    cout << "+-----------------+-----------------+-----------------+" << endl;
    for (int x=0;x<9;x++) {
        cout << "|";
        for (int y=0;y<9;y++) {
            if (s[x][y]!=LEEG) cout << "  " << s[x][y] << "  "; else cout << "  _
";
            if ((y+1)%3) cout << " "; else cout << "|";
        }
        cout << endl;
        if (!((x+1)%3)) cout << "+-----------------+-----------------+------------
-----+" << endl;
        else            cout << "|                 |                 |
|" << endl;
    }
}
```

Academiejaar 2014 - 2015

Theorie

Elk antwoord diende gegeven te worden in max. 200 woorden

1. Wat wordt verstaan onder *logical constness* in de context van mutabele variabelen?
2. Waarvoor wordt het keyword *"this"* gebruikt in C++?

3. Wat is het verschil tussen volgende vier functie-declaraties?
    - void functie1(const WeirdClass & a);
    - void functie2(WeirdClass & a);
    - void functie3(const WeirdClass a);
    - void functie4(WeirdClass a);
4. Welke output wordt er getoond bij het uitvoeren van volgend programma? Leg uit wat er gebeurt.

```
#include <iostream>

int answer = 42;

void question(){
    std::cout << answer << std::endl;
    answer += 1;
    std::cout << answer << std::endl;
}

int main(){
    question();
    std::cout << answer << std::endl;
    return 0;
}
```

1. Leg uit wat er in volgende functie5 gebeurt en verbeter ze indien nodig.

```
void functie5(){
    int **lijstje = new int*[2];
    *lijstje[0] = 13;
    *lijstje[1] = 42;
}
```

Praktijk

Opgave Bij een foutgelopen experiment van prof. dr. Bart Goethals werden zijn proefkonijnen blootgesteld aan een overdosis radio-actieve straling. Op het eerste zicht leek er niets mis met de konijnen, maar bij nader onderzoek blijkt hun reproductiesysteem grondig ontregeld: dat verloopt nu in fasen.

Stel dat er op een gegeven ogenblik n mutant-konijnen in het konijnenhok zitten:

- Indien n even is, dan gaan de konijnen in hongerstaking tot er nog maar de helft over blijft.
- Indien n oneven is, dan kweken ze bij tot ze zijn aangegroeid tot 3n+1 konijnen.

Dit proces herhaalt zich tot er maar 1 konijn over is.

Deel 1

- Voorzie een klasse *Konijnenhok*, die in de constructor een integer *aantal* binnenkrijgt.

- (Controleer dat aantal een positief getal is!) Dit getal wordt opgeslagen in een attribuut van de klasse.
- Voorzie dan een methode *void sequentie()*, die de sequentie van aantal konijnen op het scherm afdrukt, beginnende van het in de constructor opgegeven aantal.
- Voorzie een main functie die de volgende stappen onderneemt:
    - Vraag de gebruiker om een getal in te geven.
    - Lees het getal in
    - Instantieer een object van de klasse Konijnenhok met het ingelezen aantal.
    - Roep dan op dit object de methode sequentie() op om de sequentie van aantallen konijnen af te drukken.

Voorbeeld:
8 4 2 1

Deel 2

- Voorzie nu een methode *int lengte(int)*, die op een recursieve manier de lengte van de sequentie van het aantal konijnen berekent (het aantal generaties dus), beginnende van het aantal dat als parameter is opgegeven.
- Voorzie ook de methode *int lengte()*, die de voorgaande methode oproept met het aantal dat in het attribuut van je klasse is opgeslagen.
- Breidt vervolgens ook de main functie uit met een oproep naar deze nieuwe methode en print het resultaat af.

voorbeeld:
Als je start met 7 konijnen is het resultaat 17.

Deel 3

- Voorzie een methode *int maxLengte()*, die berekent wat de maximale lengte is van alle sequenties die beginnen bij een aantal kleiner dan of gelijk aan het in de constructor opgegeven aantal. Je mag hiervoor gebruik maken van de methode uit het vorige deel.
- Breidt vervolgens ook de main functie uit met een oproep naar deze nieuwe methode en print het resultaat af.

Voorbeeld:
Indien je object 8 konijnen heeft, zijn de lengte van de sequenties van 1->8: 1,2,8,3,6,9,17,4. maxLengte() zal dus 17 teruggeven.

Deel 4

- Voorzie een methode *int grootsteAantal()* die berekent wat het grootste aantal konijnen is dat kan voorkomen in een sequentie die start met een aantal kleiner dan of gelijk aan het in de constructor opgegeven aantal.
- Breidt vervolgens ook de *main* functie uit met een oproep naar deze nieuwe methode en print het resultaat af.

Voorbeeld:
Indien je object is geïnstantieerd met 8 konijnen, is het grootste aantal konijnen 52.

Deel 5 Voorzie tenslotte een manier om een instantie van de klasse Konijnenhok af te drukken door operator overloading. Die de informatie uit al de vorige delen afdrukt op het scherm.
Voorbeeld:

```
Konijnenhok* hok = new Konijnenhok(7);
cout << hok << endl;
```

krijgt de volgende output:
Dit konijnenhok krijgt 17 generaties met devolgende aantallen konijnen:
7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
De grootste generatie bedraagt 52 konijnen.
Het grootste aantal generaties met 7 konijnen of minder is 17.

Academiejaar 2013 - 2014

Theorie

1. Aanschouw volgend stukje 'code'.

Wat gebeurt er wanneer je het programma uitvoert? Verklaar!

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

int main(){
    int * p;
    cout << *p << endl;
    return 0;
}
```

1. Wat is het verschil tussen 'call by value' en 'call by reference' in een user defined function? (met voorbeeld!)
2.
    1. Wanneer we gebruik maken van inheritance in C++, wat wordt er dan juist overgeerfd van de 'base class'?
    2. Wat is het verschil tussen de drie 'access types': public, private en protected?
    3. Waarvoor staat het keyword 'this'?
    4. Wat is een static class member?
3. Wat is het verschil tussen de variabelen b en c na het uitvoeren van onderstaande code?

```
myClass a, b;
myClass c = a;
b = a;
```

Praktijk

1. Belangrijke info:

- Maak deze opdracht individueel. Het gebruik van communicatiemiddelen zoals smartphone en internet is niet toegestaan.
- Je hebt 2 uur de tijd.
- Het gebruik van eigen lesmateriaal is niet toegestaan.
- Dien je oplossing in door ze te kopieren op de USB-stick van de docent.
- Zorg dat je code voldoende gedocumenteerd is.

1. Opgave:

De bedoeling van deze opdracht is dat je het spel Tic Tac Toe implementeert. Bovendien moet je een Artificieel intelligente component maken die het toelaten voor en "single player" om tegen de computer te spelen. Tic Tac Toe is een eenvoudig spel voor twee spelers. Boter-kaas-en-eieren is in het Nederlands verreweg de meest gebruikte naam voor het spel. Het spel wordt met een potlood of een pen op papier gespeeld. Het spel wordt gespeeld op 3 bij 3 velden. Bij het begin zijn alle velden leeg. De ene speler zet een 'kruis' en de andere speler een 'rondje'. Degene die drie van zijn eigen tekens op een rij heeft (diagonaal, verticaal of horizontaal), heeft gewonnen.

Voor iemand die het spel goed genoeg kent, is het eenvoudig om ieder spel in een gelijkspel te laten eindigen, ongeacht wat de tegenstander doet. Zeker voor beginners is het veld in het midden het belangrijkste veld.

De artificiele intelligentie gaat uit van de volgende regel: kies steeds de regel met de hoogste prioriteit uit deze tabel:

Prioriteit	Regel
1	**Win**: Als je zelf al twee op een rij hebt, speel je de derde plaats om drie op een rij te krijgen.
2	**Block:** Als de tegenstander twee op een rij heeft, speel je op die rij de derde plaats om de tegenstander te blokkeren.
3	**Center:** Speel de plaats in het midden van het bord.
4	**Opposite Corner:** Als de tegenstander een plaats in een hoek heeft gespeeld, speel dan de plaats in de tegenoverstaande hoek.
5	**Empty Corner:** Speel de plaats in een willekeurige hoek.
6	**Empty Side:** Speel de plaats in het midden van een willekeurige zijde van het bord

Academiejaar 2012 - 2013 - 1e zittijd

Theorie

1. Wat is het verschil tussen een struct en een class?
2. Wat is er mis met onderstaande functie?

```
int* functie (int a, int b)
{
        int resultaat = a % b;
        return & resultaat ;
}
```

1. Geef de output van onderstaand programma weer. Daar waar adressen worden uitgeprint mag je symbolische aanduidingen gebruiken zoals bijvoorbeeld <adres-van-var1>.

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

int main () {
        cout << " Examen Inleiding Programmeren" << endl ;
        int myData [2][3] = { { 10, 30, 50 }, { 20, 40, 60 } };
        int * myArray [6];
        for (int i = 0; i < 6; i++) {
                myArray [i] = & myData [i % 2][ i / 2];
        }
        int ** myPtr = myArray ;
        while ( myPtr <= & myArray [5]) {
                cout << ** myPtr << endl ;
                cout << myPtr ++ << endl ;
        }
        return 0;
}
```

1. Wat is het verschil tussen call by value en call by reference, en waarvoor worden ze gebruikt?
2. Compileert onderstaand programma? Indien niet, verbeter het met zo weinig mogelijk aanpassingen aan de class definitie van A. Indien wel, verklaar wat er in het programma gebeurt.

```
class A {
  public:
        A() { a = 42; }
        int a;
} ;

int main() {
        const A b;
        b.a = 13;
        return 0;
}
```

Praktijk