Examen SoftwareOntwikkeling 1
2e Bachelor Informatica
Academiejaar 2006-2007
23 Januari, 2007

BELANGRIJK

1. Schrijf je naam in commentaar bovenaan elk bestand !!
2. Maak je project in de folder C:\SOI\ aan !
3. Bewaar je werk regelmatig in C:\SOI\ !
4. Deze opgave handelt over C, schrijf dus ook enkel C code en stel je compiler daarop in (zie appendix)
5. Aanpassen van de bijgeleverde header files is niet toegelaten, de main mag je wel aanpassen maar dat is in principe niet nodig
6. Warnings van MS Visual C++ Express Edition over het gebruik van `strcpy` mag je verwaarlozen

Deel III. Open Boek op computer (10u00-12u00)

Implementeer **in C** een adresboek (bijvoorbeeld voor een gsm) dat contacten bijhoudt met hun nummers. Maak hiertoe gebruik van de bijgeleverde header files en het testprogramma.

1. Contact record (6 / 20 punten):

Gegeven de volgende header file, contact_record.h:

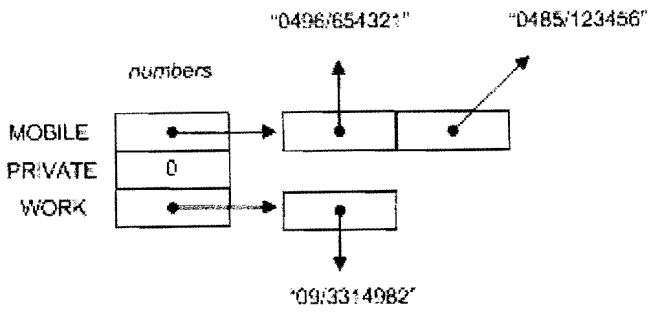
```
enum number_type {MOBILE = 0, PRIVATE = 1, WORK = 2};

struct contact_record {
      char* contact_name;
      char*** numbers;
      int* numbers_size;
};

typedef struct contact_record contact;

contact init_record(char* contact_name);
void add_number(contact* c, enum number_type numbertype,
                  char* number);
void delete_number(contact* c, char* number);
void delete_contact(contact* c);
void print_contact(contact* c);
```

Een contact heeft dus een naam en een array `numbers` die de mobiele, privé- en werknummers bijhoudt.
De interne structuur van de `numbers` array kan als volgt schematisch worden weergegeven:

```
#ifndef _VOERTUIG_H_
#define _VOERTUIG_H_

#include <string>
using std::string;

class Voertuig {
public:
      Voertuig();
      Voertuig(const string& inschrijvingsNummer,
               const string& merk,
               const string& fabricant,
               float pk);
      virtual ~Voertuig();

      virtual void printEigenschappen() = 0;
      virtual float berekenFiscaleIndex() = 0;

      string getInschrijvingsNummer();
      string getMerk();
      string getFabricant();
      float getPk();

      void setInschrijvingsNummer(const string& inschrijvingsNr);
      void setMerk(const string& merk);
      void setFabricant(const string& fabricant);
      void setPk(float pk);

protected:
      string inschrijvingsNummer;
      string merk;
      string fabricant;
      float pk;
};
```

```
class Auto: public Voertuig {
public:
      Auto();
      Auto(const string& inschrijvingsNummer,
            const string& merk,
            const string& fabricant,
            float pk,
            int aantalPassagiers);
      virtual ~Auto();

      virtual void printEigenschappen();
      virtual float berekenFiscaleIndex();

      int getAantalPassagiers();
      void setAantalPassagiers(int aantalPassagiers);
```

```
protected:
      int aantalPassagiers;
};

class BestelWagen: public Voertuig {
public:
      BestelWagen();
      BestelWagen(const string& inschrijvingsNummer,
                  const string& merk,
                  const string& fabricant,
                  float pk,
                  float hoogteLaadRuimte,
                  float breedteLaadRuimte,
                  float diepteLaadRuimte);
      ~BestelWagen();

      virtual void printEigenschappen();
      virtual float berekenFiscaleIndex();

      string getHoogteLaadRuimte();  //in cm
      string getBreedteLaadRuimte(); //in cm
      string getDiepteLaadRuimte();  //in cm

      void setHoogteLaadRuimte(float hoogteLaadRuimte);   //in cm
      void setBreedteLaadRuimte(float breedteLaadRuimte); //in cm
      void setDiepteLaadRuimte(float diepteLaadRuimte);   //in cm

protected:
      float hoogteLaadRuimte;  //in cm
      float breedteLaadRuimte; //in cm
      float diepteLaadRuimte;  //in cm
};

class VoertuigPark {
public:
      VoertuigPark(int maximaalAantalVoertuigen);
      virtual ~VoertuigPark();

      void printEigenschappenAlleVoertuigen();
      void voegVoertuigToe(const Voertuig* voertuig);
      void verwijderVoertuig(const string& inschrijvingsNummer);

      int getMaximaalAantalVoertuigen();
      int getAantalVoertuigen();

private:
      int maximaalAantalVoertuigen;
      int aantalVoertuigen;
      Voertuig** VoertuigArray;
};

#endif
```



De `numbers_size` array houdt bij hoeveel nummers van elke soort aanwezig zijn in `numbers`.

Gevraagd is de functies `init_record`, `add_number`, `delete_number`, `delete_contact` en `print_contact` te implementeren in een file contact_record.c. In het bijgeleverde bestand main.c staat reeds code om deze functies te testen.

2. Contact lijst (3 / 20 punten):

Gegeven de volgende header file, contact_list.h:

```
#include "contact_record.h"

struct listhdr {
      struct listhdr *prev, *next;
};

void list_init(struct listhdr *list);
void list_addfront(struct listhdr *list, struct listhdr *item);
void list_remove(struct listhdr *list, struct listhdr *item);

struct contactlist_record {
      struct listhdr list;
      contact c;
};
```

Maak een dubbelgelinkte lijst van `struct listhdr` elementen door bovenstaande functies te implementeren. Deze functies verwachten als eerste argument "het anker" van de gelinkte lijst. Dit is een element dat tezelfdertijd het begin als het einde van de lijst markeert. Het anker is wel geen "echt" element, het behoort niet tot de elementen van de lijst. De next pointer wijst naar het eerste element in de gelinkte lijst en de prev pointer wijst naar het laatste. Bij een lege lijst wijzen beide pointers naar "het anker" zelf.

3. Adresboek (2,5 / 20 punten)

In contact_list.h staat nog 1 extra functie, namelijk:

```
char** get_numbers(struct listhdr *list, char* contactname,
                           enum number_type, int* number_count);
```

Deze functie zoekt de nummers van een bepaald type van een bepaald contact op in de opgegeven lijst, en geeft deze terug in een array. Het aantal nummers dat in de array bevat zit, wordt in de functie ingesteld in de variabele `number_count`. Implementeer de functie `get_numbers`.

Hint:je kan een `struct contactlist_record` casten naar een `struct listhdr` (omdat het eerste element een `struct listhdr` is).
Voor het testen van deel 2 en 3 is in het bijgeleverde bestand main.c reeds code in commentaar voorzien om deze functies te testen.

Veel succes!

Appendix (voor de volledigheid):

Project aanmaken in MS Visual C++ EE:

- Kies **File – New – Project** en selecteer als Project Type **Win32**.
- Als Template selecteer je **Win32 Console Application**.
- Onderaan vul je dan een projectnaam, de opslaglocatie (C:\SOI\) en de naam van de (nieuwe) Solution in.
- Na bevestigen wordt een Win32 Application wizard opgestart. Druk op Next, check of Console Application als Application type wordt weergegeven en vink **Empty Project** aan. Vervolgens kan je deze wizard beëindigen. Er is nu een nieuw project en een nieuwe Solution aangemaakt.
- De header files toevoegen kan je door rechts te klikken op de '**Header files**' folder en **Add – Existing Item** aan te klikken

Compiler instellen op C code:

Properties – Configuration Properties – C/C++ - Advanced – Compile As…
=> compile as C

Examen Software Ontwikkeling I
2e Bachelor Informatica
Academiejaar 2006-2007
23 Januari, 2007

BELANGRIJK : Schrijf je naam onderaan dit blad

Deel I. Gesloten Boek deel (8u30-8u45)

Leg volgende 3 begrippen kort en bondig uit :

a. Concurrent Version System (CVS)

Dit is een server die op een centrale plaats staat en waar broncode kan worden opgeslagen. Zo is het mogelijk voor iedereen die aan een bep. bestand/project werkt, om die bestanden af te halen, aan te passen en terug op te laden.
Er wordt voor elk bestand de verschillende versies bijgehouden, evenals de verschillen tussen deze versies

b. Make-bestand

Een bestand dat de voorwaardelijke compilatie verzorgt. Hierin worden bepaalde bestanden "gelinkt" zodanig dat als de ene aangepast wordt, ook de andere wordt gecompileerd.

c. Cross-compilatie

Het compileren van een bestand op het ene systeem, zodanig dat het uitvoerbaar is op het andere systeem.

Examen Software Ontwikkeling I

2e Bachelor Informatica
Academiejaar 2006-2007
23 Januari, 2007

BELANGRIJK
1. Schrijf je naam onderaan op elk blad
2. Dit deel handelt enkel over C++ (deel III enkel over C)
3. Veel succes!

Deel II. Open Boek gedeelte op papier (8u45-9u45)

Voor het beheer van een voertuigpark in een bedrijf (alle bedrijfswagens en bestelwagens) gebruikt men C++ software. Bedrijfswagens worden gecategoriseerd volgens het aantal passagiers en bestelwagens volgens de dimensies van hun laadruimte. Bijgevoegd vind je het header-bestand `voertuig.h` met de declaraties van de klassen.

Vraag 1:
[op 2 punten van de 20]

Zoals aangegeven in het header-bestand, wordt in de klasse `VoertuigPark` een array van pointers naar Voertuigen bijgehouden, waarbij het aantal array-elementen in de constructor doorgegeven wordt en opgeslagen in het private attribuut `maximaalAantalVoertuigen`.
Schrijf de constructor en destructor van de klasse `VoertuigPark`.

a. Constructor:

```
VoertuigPark::VoertuigPark(int maximaalAantalVoertuigen)
Voertuigarray = new Voertuig* [maximaalAantalVoertuigen];
this.maximaalAantalVoertuigen = maximaalAantalVoertuigen;
```

b. Destructor:

```
VoertuigPark::~VoertuigPark()
delete [] Voertuigarray;
```

Geef ook de implementatie van een constructor en de destructor van de klasse Voertuig.

c. Constructor:

```
Voertuig::Voertuig (const string& inschrijvingsNummer,
                    const string& merk,
                    const string& fabricant,
                    float pk)
this.inschrijvingsnummer = & inschrijvingsNummer;
"      merk              = & merk
"      fabricant         = & fabricant;
"      pk                = pk;
```

d. Destructor:

```
Voertuig::~Voertuig()
```

Vraag 2:

[op 1 punt van de 20]

Leg uit waarom:

1. de methoden printEigenschappen() en berekenFiscaleIndex() in de klasse Voertuig als virtual gedefinieerd zijn en gelijk aan 0.
2. in de klasse VoertuigPark een array van pointers naar Voertuigen bijgehouden wordt (en niet een array van Voertuigen).

1. Dit is omdat Voertuig een abstracte bovenklasse is van Auto en Bestelwagen. Deze methodes zullen in de subklasse ~~te~~ geïmplementeerd en het heeft dus geen belang hoe de implementatie er uit ziet in de klasse Voertuig, want deze zal nooit uitgevoerd worde. Hiervoor stellen we deze gelijk aan 0.

2. Als je een nieuw voertuig aanmaakt (met new Voertuig(...)) dan geeft dat een pointer naar de Voertuigtemp
Omdat deze Voertuigen uit de array moeten kunnen gehaald worde en verwijderd worde. Dit zou niet kunnen mochte er gewoon waarde inzitten.

Vraag 3:

[op 2 punten van de 20]

Men wil in het publieke deel van de klasse `VoertuigPark` een operator + definiëren, zodat men objecten van dit type kan samennemen als bedrijven samensmelten.
Zorg ervoor dat volgende code ook mogelijk is:

```
vp_new = vp1+vp2+vp3;
/*vp_new, vp1, vp2, vp3 zijn geïnstantieerde objecten van de
VoertuigPark klasse */
```

a. Schrijf de + operator (signatuur + implementatie) van de klasse `VoertuigPark` :

```
VoertuigPark operator + ( VoertuigPark vp ) {
for ( int i = 0 ; i < maximaal_ ; i ++)
{
    voegVoertuigToe ( vp.VoertuigArray[i]);
    aantalVoertuigen ++;

}
delete vp;
return this;
}
```

b. Leg het belang van het return type van deze operator uit.

Het return type moet terug een VoertuigPark zijn, aangezien 2 voertuigparken samen terug een VoertuigPark is

Vraag 4:

[op 2 punten van de 20]

Men wil in de `VoertuigPark` klasse een STL container gebruiken in plaats van de array als privaat attribuut. Voor gemakkelijk opzoeken van voertuigen op basis van hun inschrijvingsnummer kiest men voor een associatieve container en meerbepaald een `map` (sleutel: inschrijvingsNummer en waarde: pointer naar Voertuig object).

a. Schrijf de declaratie van deze map als privaat attribuut van de `VoertuigPark` klasse:

```
private:
    map < string, Voertuig * > VoertuigMap;
```

b. Schrijf de methode `printEigenschappenAlleVoertuigen` in het geval van implementatie met de STL container `map`. Maak gebruik van een iterator.

```
void VoertuigPark:: printEigenschappenAlleVoertuigen()
{
    for (map<string, Voertuig*>::const_iterator iter = VoertuigMap.
        begin(); iter != VoertuigMap.end(); iter++)
    {
        ((*iter).second)->printEigenschappen();

    }
}
```