Installatie-handleiding
en
Practicum-opgaven TICT-V1ADC-16
Algorithms and Datastructures in C

Cursuseigenaar en auteur: Joop Kaldeway
Versie: 1

Inhoudsopgave

Installatie van CodeLite.

Kenmerken:

- Een cross-platform IDE.
- Bij 'windows' moet de mingw-omgeving ook geïnstalleerd worden.

Sites:

- http://tdm-gcc.tdragon.net/
- http://codelite.org/
- http://codelite.org/LiteEditor/Documentation
- http://codelite.org/AddNewCompiler/AddNewCompiler

Installatie op Windows:

- Download en run 'tdm-gcc-5.1.0-3.exe' van de site http://tdm-gcc.tdragon.net/download
  Het volgende wordt uitgevoerd:
  - De map 'C:\TDM-GCC-32' wordt gemaakt. In de submap 'bin' staat o.a. 'gcc.exe'
  - Aan de omgevingsvariabele 'path' wordt 'C:\TDM-GCC-32\bin' toegevoegd.
- Download 'codelite-amd64-x.y.z.7z' van de site http://codelite.org/
- Pak 'codelite-amd64-x.y.z.7z' uit en installeer CodeLite
- Start CodeLite: de setup-wizard wordt opgestart.
  - Selecteer twee keer 'Next >' .
  - Selecteer 'Scan'. Selecteer de compiler MinGW(TDM-GCC-32)

Een C-programma uitvoeren:

- Maak een workspace: File → New → New Workspace
- Maak in workpace een project: Selecteer workspace → rechtermuisknop → New → New Project → Kies Simple executable (gcc) : in de map 'src' staat nu de file 'main.c'
- Kies menu Build → Run

Compiler-opties.

Standaard worden de volgende opties meegegeven:

- -g : produceer debug informatie
- -O0 : optimize for debugging
- -Wall : enable all warning messages
- -std=gnu11 : enable c11 features + gnu-extensions

Het is mogelijk dit aan te passen:

- -Werror : behandel alle warnings als fouten
- -std=c11 : sta geen gnu-extensies toe
- -pedantic : sta geen extra extensies toe

De aanpassing kun je doen via Menu Workspace → 'Open Active Project Settings…' → C compiler options.

Handig:

- Voeg toolbar toe: men View → Toggle Minimal View

Debuggen.

Om debuggen mogelijk te maken moet het volgende ingesteld worden:

Menu Settings → GDB Settings … → GNU gdb debugger → General →
Debugger path: C:\TDM-GCC-32\gdb32\bin\gdb32.exe

Practicum-richtlijnen.

- In de student-instructie staan 25 practicum-opdrachten. Alle opdrachten moeten worden gemaakt.
- Het practicum wordt individueel gedaan.
- Er mag onderling overlegd worden. Aantoonbaar jat-gedrag wordt gemeld aan de examencommissie
- De ontwikkelomgeving CodeLite wordt gebruikt.
- Alle practicum-opdrachten worden opgeslagen in een Codelite-workspace.
- De docent kan tijdens de werkcolleges vragen de code te tonen en toe te lichten.
- De docent kan vragen de workspace in zip-vorm op te sturen. De email heeft als onderwerp: C-practicum <student-naam> <studentnr> .
- De c-files moeten extensie '.c' hebben.
- C-warnings zijn niet toegestaan.

Week 1

Practicum-opdracht 1.1

Leerdoelen:
- Werken met 'printf' en 'scanf'.
- Bouwen van een robuust programma

Maak een programma dat twee integer variabelen (m.b.v. 'scanf') inleest en vervolgens de getallen bij elkaar optelt. Druk met 'printf' een tekst af die er als volgt uitziet:

x + y = z

Hier staat op 'x' de waarde van de eerste variabele, op 'y' staat de waarde van de tweede variabele en op 'z' staat de uitkomst van de optelling.
Beveilig het programma tegen incorrecte invoer.

Practicum-opdracht 1.2

Leerdoelen:
- Werken met de keuze-opdracht.
- Werken met het char-type
- Oefenen met representaties van getallen

Maak een C-programma dat een teken inleest (met 'getchar') en de ASCII-waarde hexadecimaal afdrukt als het een hoofdletter is en decimaal als het een kleine letter is, en het teken zelf als het teken geen letter is.

Practicum-opdracht 1.3

Leerdoelen:

- Oefenen met een for-statement.
- Oefenen met een while-statement

Maak een C-programma dat een geheel getal n inleest, met 0 < n < 81.
Vervolgens wordt het volgende afgedrukt:

```
*
**
***
…
** .. * ( n sterretjes )
…
***
**
*
```

Zo wordt bij n = 5 afgedrukt:

```
*
**
***
****
*****
****
***
**
*
```

Doe dit op twee manieren:

- met een for-statement
- met een while-statemen

Practicum-opdracht 1.4

Leerdoel:

- Netjes weergeven van gegevens m.b.v. uitlijnen.

Maak een C-programma dat een geheel getal n inleest met $0 < n < 100$. Vervolgens wordt de tafel van n afgedrukt. De kolommen zijn daarbij rechts uitgelijnd.

Practicum-opdracht 1.5

Leerdoel:

- Oefenen met een eenvoudige parsing-techniek: het omzetten van tekst naar een getal.

Een geheel getal wordt op het toetsenbord ingetikt. Het getal kan voorafgegaan worden door '–' of '+'. Nadat het getal is ingevoerd wordt op de Enter-toets gedrukt.
Schrijf een programma, dat het getal inleest en de waarde in 'i' plaatst. Alleen de functie 'getchar' mag gebruikt worden.
De functie 'scanf' mag niet gebruikt worden.
Beveilig het programma tegen incorrecte invoer.

Het programma heeft de volgende opbouw:

```
int main(void) {

      int i;
      ...
      ...

      ...
      printf("het ingevoerde getal is: %d\n", i);
}
```

Week 2

Practicum-opdracht 2.1

Leerdoelen:
- Oefenen met het inlezen van gegevens voor een int-array.
- Oefenen met het doorlopen van een array

Schrijf een programma dat een int array met lengte 10 vult met nullen en enen. De nullen en enen worden ingelezen. Ga na of het aantal enen gelijk is aan het aantal nullen en druk het resultaat af.

Practicum-opdracht 2.2

Leerdoelen:
- Oefenen met fgets.
- Oefenen met het vergelijken van twee strings

Schrijf een programma dat twee teksten in twee char-arrays inleest. Gebruik hierbij 'fgets'.
De char-arrays bevatten maximaal 80 karakters.
Het programma zoekt de eerste letter waarbij deze twee strings verschillen en drukt de waarde van de array-index af.

De interactie verloopt als volgt:

tik tekst 1 in: In zee met C
tik tekst 2 in: In zee leven kwallen
het verschil in tekst begint bij index 7

Houd rekening met de volgende situatie: de ene regel kan in zijn geheel overeenkomen met het voorste gedeelte van de andere regel.

Bijvoorbeeld:

tik tekst 1 in: In zee met C
tik tekst 2 in: In zee met C en Python

Practicum-opdracht 2.3

Leerdoelen:
- Oefenen met het doorlopen van een array m.b.v. indices
- Oefenen met het doorlopen van een array m.b.v. pointers

Schrijf een programma een tekst in een char-array inleest. De char-array bevat maximaal 80 karakters. Draai vervolgens de inhoud van de char-array om. Gebruik hierbij geen extra array. Doe dit op twee manieren:
- met indices
- met pointers

De interactie verloopt als volgt:

tik tekst in: een voorbeeld
in omgekeerde volgorde: dleebroov nee

Practicum-opdracht 2.4

Leerdoelen:
- Oefenen met het inlezen van gegevens voor een struct-variabele.
- Oefenen met het afdrukken van gegevens van een struct-variabele.

Gegeven is de struct-definitie:

```
typedef struct voorwerp
{
   int nummer;
   char naam[20];
   double gewicht, lengte;
} VOORWERP;
```

Schrijf een programma, dat een voorwerp inleest en vervolgens afdrukt.

De interactie verloopt als volgt:

nummer: 423
naam: stoel
gewicht: 4.5
lengte: 90

stoel heeft nummer 423, weegt 4.500000 kg en is 90.000000 cm

Let op het volgende:
- nadat je een waarde gelezen hebt, moet je de rest van de regel 'skippen'.
- de format-specifier bij 'scanf' van een 'double' is: %lf

Practicum-opdracht 2.5.

Leerdoel:

- Oefenen met tweedimensionale array

Schrijf een C-programma, waarbij een matrix als volgt is gedeclareerd:

```
int matrix[][10] = {{-1,-1,-1,-1,-1,-1, 0, 1,-1, 1},
                    { 1,-1,-1,-1,-1,-1,-1,-1,-1, 1},
                    {-1,-1, 0,-1, 1,-1,-1,-1,-1,-1},
                    {-1,-1,-1,-1, 0,-1,-1,-1,-1,-1},
                    {-1,-1,-1,-1,-1,-1, 0,-1,-1, 1},
                    {-1,-1,-1,-1,-1, 1,-1,-1, 1,-1},
                    {-1,-1,-1,-1,-1,-1, 0,-1,-1, 0},
                    { 0,-1,-1,-1,-1,-1,-1,-1,-1,-1},
                    {-1,-1, 1, 1,-1,-1,-1,-1,-1, 1},
                    { 0,-1, 1,-1,-1,-1,-1,-1, 0,-1}};
```

Druk de matrix af.

De uitvoer is als volgt:

```
------01-1
1--------1
--0-1-----
----0-----
------0--1
-----1--1-
------0--0
0---------
--11-----1
0-1-----0-
```

Week 3

Laat bij de ontworpen functies zien hoe de functie getest is.

Practicum-opdracht 3.1

Leerdoelen:
- Oefenen met functies.
- Oefenen met het doorlopen van een rij

Schrijf een functie die berekent hoe vaak een geheel getal n in een int-rij a voorkomt.
De functie heeft de vorm:

```
int count(int a[], int size, int n) {// size is de lengte van a
   ...
   ...
}
```

Practicum-opdracht 3.2

Leerdoelen:
- Oefenen met functies.
- Oefenen met het doorlopen van twee rijen
- Oefenen met boolean-expressies

Schrijf een functie, die bepaalt of twee int-rijen aan elkaar gelijk zijn.
De functie heeft de vorm:

```
bool equal_rows(int a1[],int a2[], int size) {
                              // size is de lengte van a1 en van a2
   ...
   ...
}
```

Practicum-opdracht 3.3

Leerdoelen:
- Combineren van functies
- Oefenen met boolean-expressies

Schrijf een functie, die bepaalt of een gegeven int-rij aan de volgende eisen voldoet:
- de rij bevat alleen nullen en enen
- het aantal enen is gelijk aan het aantal nullen
- in de rij komt een getal niet drie keer achter elkaar voor

De functie heeft de vorm:

```
bool valid_row(int a[], int size){ // size is de lengte van a
    ...
    ...
}
```

Gebruik de functie van opgave 3.1.

Practicum-opdracht 3.4

Leerdoel:
- Oefenen met matrices

Schrijf een functie, die een gegeven matrix spiegelt t.o.v. de hoofddiagonaal:

De matrix $\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix}$ wordt omgezet in : $\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$

M.a.w m[i][j] wordt verwisseld met m[j][i] .

De functie heeft de vorm:

```
void transpose_matrix(int size, int matrix[][size]) {
                                // toegestaan vanaf C99
                                // 'size' moet voor matrix staan
    ...
    ...
}
```

Practicum-opdracht 3.5.

Leerdoelen:

- Oefenen met bit-patronen

a. Schrijf de functie

```
int get_bit(char ch, int n)
```

De functie bepaalt de bitwaarde van bit 'n' van byte 'ch'. De bits zijn van links naar rechts genummerd, te beginnen met 0.

b. Schrijf de functie

```
char verschuif_cyclisch(char ch, int n)
```

- Als n > 0 dan wordt n posities naar links geschoven. De weggevallen bits worden rechts teruggeplaatst.
- Als n < 0 dan wordt n posities naar rechts geschoven. De weggevallen bits worden links teruggeplaatst.

Week 4

Practicum-opdracht 4.1

Leerdoelen:
- Oefenen met binaire files
- Oefenen met 'stderr'

(zie boek blz. 109)

Maak een programma dat bij aanroep twee bestandsnamen meekrijgt en onderzoekt of de inhoud van de twee bestanden hetzelfde is. Als dat zo is moet de returnwaarde van het programma 0 zijn, en zo niet, dan is de returnwaarde 1 en geeft het programma op 'stderr' de positie van de eerste byte waarop de beide bestanden verschillen.

Practicum-opdracht 4.2

Leerdoelen:
- Oefenen met het inlezen van gegevens uit een ASCII-file
- Oefenen met matrices

De file 'binary_sudoku_puzzle.txt' bevat de tekst:

```
------01-1
1--------1
--0-1-----
----0-----
------0--1
-----1--1-
------0--0
0---------
--11-----1
0-1-----0-
```

Schrijf een functie die de file leest en de inhoud plaatst in een matrix.

De functie heeft de volgende opbouw:

```
int read_matrix(const char *filename, int size, int matrix[][size])
{
   ...
   ...
}
```

Voor het bepalen van 'size' moet de file vooraf één keer doorlopen worden.
De functie geeft de waarde 0 terug als de file correcte data bevat.
De functie geeft de waarde 1 terug als de file verkeerde data bevat.

Practicum-opdracht 4.3

Leerdoelen:

- Oefenen met het lezen uit en schrijven naar een ASCII-file
- Oefenen met het herkennen van speciale karakters in een ASCII-file

Schrijf een compress-programma, dat uit een gegeven C-file een nieuwe file maakt, waarbij van iedere regel alle spaties en tabs aan het begin van de regel zijn verwijderd. Verder zijn alle lege regels verwijderd. (een lege regel bevat '\n' , eventueel voorafgegaan door spaties en tabs) .

Het programma heeft de volgende opbouw:

```
void compress(char* src_filename, char* dest_filename){
      ...
      ...
}

int main(int argc, char* argv[])
{
    compress("opdracht_4_3.c","opdracht_4_3_compressed.c");
    return EXIT_SUCCESS;
}
```

Practicum-opdracht 4.4

Leerdoel:

- Oefenen met het filteren van getallen uit een ASCII-file

Schrijf een functie

```
int getIntegers(char* filename, int a, int size)
```

De functie plaatst alle gevonden gehele getallen uit de ASCII-file in array a. Het aantal ingelezen elementen wordt teruggeven. De file kan naast getallen ook tekst bevatten. Het bevat maximaal 100 gehele getallen.

Het programma heeft de volgende opbouw:

```
int getIntegers(char* filename, int a, int size){
      ...
      ...
}

int main(void) {
    int a[100];
    int n = getIntegers("getallen.txt",a, 100);

    if (n > 0) {
      puts("gevonden getallen:");
      for (int i = 0;i < n; i++){
            printf("%d ",a[i]);
      }
    putchar('\n');

    return EXIT_SUCCESS;
}
```

Voorbeeld:

'getallen.txt' bevat de volgende tekst:

12 -34 56 - 23423424
12voorbeeld-34en+56ge-tal345

Uitvoer:

gevonden getallen:
12 -34 56 23423424 12 -34 56 345

Practicum-opdracht 4.5.

Leerdoel:

- Filteren van woorden uit een ASCII-file

Schrijf een functie

```
int getWords(char* filename, int size, char a[][size])
```

De functie plaatst alle gevonden woorden uit de ASCII-file in een array.
Een woord is opgebouwd uit alphanumerieke karakters. Het is niet bekend hoe lang de woorden kunnen zijn. De file bevat maximaal 1000 woorden.
Gebruik de functie 'isalnum'. (zie http://www.cplusplus.com/reference/cctype/isalnum/)

Voor het bepalen van de lengte van het langste woord moet de file één keer doorlopen worden.

Het programma heeft de volgende opbouw:

```
int getWords(char* filename, int word_size, char a[][word_size]){
      ...
      ...
}

int main(void) {
    int word_size;
    // lees de hele file voor het bepalen van de lengte van het langste woord

    char a[1000][word_size];
    int n = getWords("opdracht_4_5.c", size, a);
    if (n > 0){
       puts("gevonden woorden:");
       for (int i = 0;i < n; i++){
          printf("%3d %s\n",i,a[i]);
       }
    }

    return 0;
}
```

Week 5

Practicum-opdracht 5.1

Leerdoel:
- Oefenen met recursie
- Oefenen met pointers

Schrijf een functie

```
bool is_sorted(int a[], int size) { // size is lengte van a
    ...
    ...
}
```

De functie gaat recursief na of de rij a gesorteerd is.

Onderscheid daarbij de volgende gevallen:

- **size <= 1**
- **size == 2**
- **size > 2**

Practicum-opdracht 5.2

Leerdoel:
- Oefenen met het op basis van efficiëntie vergelijken van functies

Genereer met de random-functie rand() (zie http://www.cplusplus.com/reference/cstdlib/rand/) een int-array met 10000 willekeurige elementen.
Pas de sorteer-methoden 'insertion sort' en 'merge sort' toe op de rijen en bepaal per methode de tijdsduur.
Maak eerst een kopie van de oorspronkelijke rij. (Waarom?)

Practicum-opdracht 5.3

Leerdoelen:
- Oefenen met een bibliotheek-functie

C bevat een ingebouwde 'qsort'-functie. (zie http://www.cplusplus.com/reference/cstdlib/qsort/)

Vergelijk deze functie met 'insertion sort' en 'merge sort'.
Doe dit op dezelfde manier als bij opdracht 5.2

Practicum-opdracht 5.4

Leerdoelen:
- Oefenen met de stack
- Oefenen met parsing-technieken

Een haakjes-expressie bevat de volgende karakters:

openingshaakjes: '(' , '[' , '{' , '<'
sluitingshaakjes: ')', ']', '}', '>'

De volgende haakjesexpressies zijn geldig: (())[]<{}> , <(())> , ()()()
Voorbeelden van niet geldige haakjes-expressie zijn: (())) ,)(, <((, (>

Schrijf een programma dat een haakjes-expressie inleest en nagaat of de expressie geldig is.
Dit is op te lossen m.b.v. een stack.

Pas de volgende strategie toe:
- als een ingelezen karakter een openingshaakje is, wordt het op de stack geplaatst.
- als een ingelezen haakje een sluitingshaakje is, wordt nagegaan wat het karakter op de top van de stack is.
  - is de stack leeg of is het karakter niet het corresponderende openingshaakje, dan is de haakjesexpressie niet geldig.
  - is het karakter het corresponderende openingshaakje, dan wordt dit karakter van de stack verwijderd.
- Nadat de hele haakjesexpressie is verwerkt, moet de stack leeg zijn.

(De stack-code moet aangepast worden; i.p.v. een int-array moet een char-array gebruikt worden)

Practicum-opdracht 5.5.

Leerdoelen:

- Oefenen met het implementeren van een datastructuur in C
- Oefenen met het werken met meerdere source-files (gescheiden compilatie)

Implementeer in C de queue.

Maak drie files:

- queue.h
- queue.c
- main.c

'queue.h' bevat de struct:

```
typedef struct {
            int a[QUEUESIZE];
            int head;
            int tail;
      } Queue;
```

'queue.c' bevat de implementatie van de functies:

```
void init_queue(Queue* pq);
void enqueue(Queue* pq, int data);
int dequeue(Queue* pq);
void show(Queue q);
```

In 'main.c' wordt de queue getest.