UNIVERSITEIT GENT

FACULTEIT WETENSCHAPPEN

Systeemprogrammeren

Academiejaar 2018–2019

sysprog@lists.UGent.be

Oefeningenset 2
Omgaan met ééndimensionale arrays

Generatie van Acroniemen

Situering

In deze oefening zullen we een wachtlijn met prioriteit (zie cursus sectie C.4 Hopen – Heaps) implementeren, die zal gebruik worden bij het bijhouden en ophalen van acroniemen met hun toegekende prioriteit. Een Acroniem is een afkorting die uitgesproken wordt als een woord. Enkele gekende voorbeelden van acroniemen zijn "FAQ" (*Frequently Asked Questions"),* "ASAP" *(As Soon As Possible)* en wysiwyg (*What you see is what you get*). In dit practicum zullen we afkortingen genereren die zelf bestaande woorden vormen. We zullen niet enkel de eerste letters van de woorden in rekening brengen maar naast de beginletter nog één andere letter. Zo vormt "Basic Arithmetics" onder andere de Engelse woorden "BAt", "BAr" en "BiAs".

Opgave

Bij deze opgave worden zes bestanden geleverd:

- main.c: de main functie wordt meegeleverd. Deze roept, naast enkele hulpfuncties om te testen op memory leaks, een functie op die een woordenboek zal inlezen, de gebruiker zal vragen om input, een wachtlijn zal aanmaken en acroniemen zal genereren op basis van de gegeven input. Tot slot wordt al het gereserveerde geheugen terug vrijgegeven.
- priority_queue.h: het header-bestand voor de wachtlijn met prioriteit.
- priority_queue.c: het codebestand voor de wachtlijn.
- acronym.h: het header-bestand voor het genereren van de acroniemen.
- acronym.c: dit codebestand bevat al enkele voorgeïmplementeerde functies.
- wordlist.txt: de woordenlijst die doorzocht dient te worden. De gegeven woordenlijst bevat alle Engelse woorden die beginnen met de letter B. Bij het ingeven van input moet het eerste woord dus beginnen met een 'B'. Het is natuurlijk mogelijk zelf een ander woordenboek te gebruiken. (Waarbij elk woord moet beginnen op een nieuwe regel.)

main.c

Het bestand main.c bevat de volgende drie IO functies waarvan enkel de laatste twee nog door jullie moeten geïmplementeerd worden. Voorzie hierbij de nodige functionaliteit zodat fouten opgevangen kunnen worden.

- `unsigned long get_file_length(FILE* ifp);`

Deze hulpfunctie is gegeven en zal de lengte van een in te lezen bestand bepalen zodat de juiste hoeveelheid geheugen kan worden voorzien.

- `char* read_file(const char* filename);`

Deze functie zal, gebruikmakend van de voorgaand gedefinieerde functie, een bestand inlezen in het geheugen en het resultaat teruggeven als een `array` van karakters. Deze `array` bevat de volledige inhoud van het bestand (inclusief de '\n' karakters).

- `char* read_input(int* text_length);`

In deze functie zullen we input uit de console lezen. Doe dit aan de hand van de `fgetc` functie en voorzie een leesbuffer die telkens stijgt met grootte 10 als de capaciteit van de leesbuffer overschreden wordt. De initiële leesbuffer is dus 10 karakters groot (definieer hiervoor een macro BUFFER_SIZE die de initiële grootte bepaalt). Merk op dat in de grootte van de buffer het karakter '\0' niet meegerekend is. Na het inlezen van de input bevat de variabele `text_length` de lengte van de ingelezen tekst.

priority_queue.h

De wachtlijn met prioriteit, ook wel priority queue genoemd, is een Heap datastructuur. Deze zal geïmplementeerd worden zoals voorgesteld in de cursus in sectie C.4 Hopen – Heaps. De elementen in deze wachtlijn zijn koppels van acroniemen en hun bijhorende prioriteit. Dit koppel wordt voorgesteld in de volgende struct:

```
typedef struct{
	char* acroniem;
	unsigned int priority;
} queue_item;
```

De priority queue zelf bestaat uit een `array` van `queue_item`-koppels, het aantal elementen in de queue en het maximaal aantal elementen dat de queue kan bevatten.

```
typedef struct{
	queue_item* items;
	unsigned int size;
	unsigned int max_size;
} priority_queue;
```

priority_queue.c

Implementeer de volgende functies, met signatuur zoals gedeclareerd in `priority_queue.h`:

- `priority_queue* pq_create();`

Deze functie zal een nieuwe prioriteitwachtlijn initialiseren. Zorg er voor dat de wachtlijn initieel 1 element kan bevatten. Gebruik hievoor de gedefinieerde macro QUEUE_SIZE.

- `int pq_add(priority_queue* queue, char* acroniem, int priority);`

Deze functie laat toe een nieuw element toe te voegen aan de wachtlijn en werkt zoals beschreven in de cursus. Hou er rekening mee dat er bij het toevoegen moet gekeken worden als er in de wachtlijn nog plaats vrij is om een nieuw element toe te voegen. Indien dit niet het geval is dient de grootte van de wachtlijn **verdubbeld** worden. Na het toevoegen wordt telkens het `fixup` algortime toegepast, zoals beschreven in de cursus. Deze functie geeft de waarde 1 terug als het toe te voegen element nog niet in de wachtlijn aanwezig is, anders 0. Hou er dus rekening mee dat er moet gecontroleerd worden als het nieuwe element reeds in de wachtlijn aanwezig is.

- `void pq_free(priority_queue* queue);`

Deze functie geeft al het geheugen ingenomen door de wachtlijn terug vrij.

- `void pq_print(priority_queue* queue);`

Deze functie schrijft de volledige wachtlijn uit naar de console en werd reeds voorzien. Een voorbeeld van de finale output ziet er als volgt uit:

BAr : 2147480402
BAt : 2147480400
BiAs : 2147479296

- `char* pq_max(priority_queue* queue);`

Deze functie leest het element met de hoogste prioriteit uit(zonder het element te verwijderen). Als er geen elementen in de wachtlijn zitten moet de `NULL`-pointer teruggegeven worden.

Verder volgen nog enkele hulpfuncties:

- `void pq_fixup(priority_queue* queue, queue_item* item, int index);`

Deze hulpfunctie implementeert het `fixup` algoritme. We doen dit in een afzonderlijke functie zodat die recursief kan worden opgeroepen.

- `void pq_swap(queue_item* item1,queue_item* item2);`

Deze hulpfunctie wordt gebruikt in het `fixup` algoritme en zal twee elementen uit de wachtlijn van plaats wisselen.

acronym.h

Om het algoritme dat de acroniemen genereert wat te vereenvoudigen zullen we enkele voorbereidende berekeningen doen:

- `int analyse_num_words(char* words, int* total_length);`

Deze functie zal het aantal woorden uit de reeds ingelezen zin (omvat in de `words` variabele) teruggeven. De variabele `total_length` bevat na deze bewerking ook de totale lengte van de ingelezen zin.

- `void analyse_words(char* words,int total_length,int num_of_words,int* start_indices,int* lengths);`

In deze functie zullen we de startindices en de lengtes van de verschillende woorden bepalen. Waarbij de variabelen `start_indices` en `lengths` `arrays` van `int`s zijn. De `start_indices` array bevat op index *i,* de index van de eerste letter van het *i*^{de} woord in de ingelezen zin.

Het genereren van acroniemen zal op de volgende manier gebeuren: In plaats van gekozen letters bij te houden zullen we de posities van de letters in de oorspronkelijke zin bijhouden. Zo kunnen we op een eenvoudige manier de volgende letter selecteren en controleren indien het huidige woord nog extra letters bevat (m.b.v. de informatie verkregen uit de voorbereidende berekeningen). Merk op dat we slechts 1 extra letter in rekening zullen brengen. Als we bijvoorbeeld terug het initiële voorbeeld bekijken (“Basic Arithmetics”) dan verkrijgen we de volgende posities:

B	a	s	i	c		A	r	i	t	h	m	e	t	i	c	s
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

We kunnen dus de mogelijke acroniemen voorstellen als $B\{a,s,i,c\}^{+}A\{r,i,t,h,m,e,t,i,c,s\}^{+}$. Waarbij "B" en "A" letters zijn die in ieder geval moeten aanwezig zijn en kunnen gevolgd worden door een letter die zich bevindt op positie 1 tot 4 voor het eerste woord en een letter op positie 7 tot 16 voor het tweede woord. Merk op dat het ook moet mogelijk zijn geen extra letter te kiezen, vb. bij 'BAr' werd er uit het eerste woord geen letter gekozen. De `array` die het acroniem beschrijft zal dus een element bevatten voor elk woord uit de zin. In bovenstaande voorbeeld zal het eerste element uit de `array` de posities 1 tot 4 kunnen bevatten en het tweede element de posities 7 tot 16 als posities van mogelijks te kiezen letters.

Al deze functionaliteit komt samen in de `generate_acronym_recursive` functie die **reeds voor jullie geïmplementeerd werd**. De parameters van deze functie worden hier onder toegelicht:

- `int i`: index van het te verwerken woord in de recursie.
- `char** wordlist`: bevat het woordenboek. Deze parameter moet samen met het gegenereerde acroniem en de lengte van het woordenboek (`list_size`) meegegeven worden aan de functie `check_wordlist` om na te gaan als het gegenereerde woord een bestaand woord is.
- `priority_queue* queue`: de wachtlijn waaraan het gevonden acroniem dient toegevoegd te worden.
- `char* acronym`: het tussentijdse resultaat van het gegeneerde acroniem.
- `char* words`: de oorspronlijke input.
- `int num_of_words`: het aantal woorden waaruit de input bestaat.
- `int* start_positions`: de startposities van deze verschillende woorden
- `int* lengths`: de lengtes van de verschillende woorden

Hieronder volgt nog wat extra informatie omtrent enkele voorziene hulpfuncties die gebruikt worden bij het genereren van de acroniemen:

- `char** create_word_list(char* text, int* list_size);`

Deze functie zal een ingelezen tekstbestand omzetten in een woordenlijst. De lengte van de lijst komt in de variabele `list_size`. Merk op dat er een char** als resultaat wordt teruggegeven. Dit kan gezien worden als een pointer naar pointers en wordt hier gebruikt als 2-dimensionale array.

- `int check_wordlist(char** wordlist,char* acroniem,int list_size)`

Deze functie zal de woordenlijst overlopen en nagaan als het gegenereerde acroniem zich effectief in de woordenlijst bevindt. Deze functie geeft de waarde 1 terug als een match gevonden werd, anders 0.

- `int calculate_priority(char* acronym);`

Ook deze functie bepaalt de prioriteit van een gegenereerd acroniem. Aangezien we de voorkeur geven aan kortere acroniemen kennen we deze ook een hogere prioriteit toe.

- `char* convert_acronym(char* words,int* acronym,int num_of_words,int* start_positions);`

Deze functie zal de gekozen posities in de `acronym array` omvormen tot een volwaardig woord.