Structuur van Computerprogrammas II: Week 10

Elisa Gonzalez Boix Maarten Vandercammen Robbe De Greef

Datastructuren

1. (**Dodona**) Implementeer een functie `Tnode *max_value_node(Tnode *tree)` die de node met de maximale waarde in een gegeven gesorteerde boom teruggeeft.

2. (**Dodona**) Pas de code voor de binaire boom aan zodat de boom *generisch* wordt, i.e., zodat die werkt met waarden van om het even welk type (i.e., `void*`), zoals ook gedaan werd in de `quicksort` functie in slide 12 uit hoofdstuk 3c. Op Dodona en in het bestand `oefeningen.c` vind je al de definitie van een struct `Tnode_Generic` die een node in zo'n generische boom voorstelt, alsook de implementatie van de `insert` functie die je in het hoorcollege gezien hebt.

   Pas de `insert`-functie (zie slide 41 van hoofdstuk 3b) aan zodat die werkt voor generische bomen. Daarvoor zal je `insert` moeten veranderen naar `Tnode_Generic* insert_generic(Tnode_Generic* root, void *value, int (*smaller_than)(void*,void*), int (*equal_to)(void*,void*))`. Die functie neemt vier argumenten:

   - Een pointer naar de boom waarin de waarde moet toegevoegd worden.
   - De waarde die moet toegevoegd worden (deze waarde heeft het type `void*`).
   - Een function-pointer `smaller_than` van het type `int (*smaller_than)(void*, void*)` die twee waarden met elkaar vergelijkt en `1` teruggeeft indien de eerste kleiner is dan de tweede, en `0` indien niet.
   - Een function-pointer `equal_to` van hetzelfde type als het vorige argument, die `1` teruggeeft indien beide waarden gelijk zijn en `0` indien niet.

3. (**Dodona**) Schrijf een functie `int check_balanced_brackets(char *string)` om te controleren of in de gegeven string de vier soorten haakjes (`'{'`, `'('`, `'['`, en `'<'`) correct gebalanceerd zijn met hun tegenhanger. Als dat zo is, geeft de functie `1` terug, anders `0`. Voorbeelden:

   - `"Deze (string { is } <[correct] gebalanceerd>)"`
     Hiervoor geeft de functie `1` terug.

- "{ Deze <string (is >) niet } correct"
  Hiervoor geeft de functie 0 terug.

4. (**Dodona**) Maak de queue implementatie die je in het hoorcollege gezien hebt generisch, op dezelfde manier als dat je gezien voor de derde stack variant in hoofdstuk 3c. Implementeer de volgende functies:

   - `Queue create_queue(size_t elmtsize)`
   - `void insert(Queue q, void* e )`
   - `void first(Queue q, void *e)`

   De definitie van de generische queue variant wordt al gegeven op Dodona.

5. (**Dodona**) Schrijf een functie `Tnode *delete_node(Tnode *tree, int value)` die de node met de gegeven waarde uit de boom verwijdert en de bijgewerkte boom teruggeeft. Om deze oefening te maken, kan je gebruik maken van de functie `max_value_node` die je in oefening 1 geïmplementeerd hebt.

Extra

6. (**Dodona**) Pas het gelinkte lijst ADT aan zodat die generisch wordt, net zoals je gedaan hebt in oefening 2. Op Dodona en in het bestand `oefeningen.c` vind je al de definitie van een `struct node_generic` die een generische node voorstelt. Implementeer de volgende functies:

   - Een functie `struct node_generic *insert_node(void *value, struct node_generic *node)` om een element toe te voegen aan een generische lijst en de nieuwe node terug te geven.
   - Een functie `void print_list(struct node_generic *node, void (*display)(void*))` om de inhoud van de lijst te printen. Implementeer ook de functies `void print_int(void *i)` en `void print_rat(void *rat)` die respectievelijk een pointer naar een int en een pointer naar een `Rat` uitprinten. Een `Rat` met als teller `10` en als noemer `5` moet uitgeprint worden als `10/5`. Laat `print_list` een `"[ "` printen voor dat de elementen worden uitgeprint, en een `" ]"` nadat ze uitgeprint zijn.