ALGO2 - časté chyby v sadě 1

2021/2022

1 Opakování

- Pro práci se stringy používejte string.h a v ní obsaženou funkcionalitu - tj nepište například porovnávání stringů ručně.
- Řešení increase_debt a decrease_debt většinou byla CTRL-C, CTRL-V, to je špatně. Nekopírujte kusy kódu v rámci programu. Raději navrhněte proceduru, která dělá odpovídající úkol. V tomto případě stačilo pro decrease_debt(x,y) volat increase_debt(x, -y) anebo, ještě lépe, napsat change_debt(x,y) upravující dlužnou částku o x a poté volat change_debt v obou zadaných úkolech.

  Opakování kódu vede k náročné údržbě, nepřehlednému kódu a bývá zdrojem chyb v programech.
- U stringů je rozdíl pro porovnání používat == a strcmp.
- Pozor na předávání odkazem vs předávání hodnotou (zejména u struktur).

2 Lineární datové struktury - pole

Zde většinou problémy nebyly.

3 Lineární datové struktury - seznam

- Vždy kontrolujte, jestli ukazatel, který chcete použít, není NULL. U tohoto úkolu programy často padaly pro list→head == NULL na list→head→next a podobně pro node→next == NULL na node→next→next.
- Vícekrát se v řešeních opakoval kód pro procházení seznamu až na konec - nová procedura nebo makro?

4 Binární vyhledávací stromy

- Platí vše, co u seznamů.
- Výpis do šířky pomocí rekurzivního vyhledávání uzlů v jednotlivých patrech je zbytečně neefektivní - pro každý uzel se udělá extra sestup od kořene. Mnohem lepší je varianta s frontou - každým uzlem se projde právě jednou.
- Je zvykem mít menší prvky v levém podstromu a větší v pravém.

5 Obecné

- Return ukončuje proceduru. Nemá tedy smysl psát za něj další příkazy.
- Volte rozumné a popisné názvy proměnných, procedur atd.
- Dodržujte jednotné konvence jazyka, ve kterém píšete (vyberte si jedny a těch se držte).
- Každou alokovanou paměť musíte při rušení poslední reference uvolnit. Aneb používejte free. Je dobré pro kontrolu práce s pamětí použít valgrind.
- Nepoužívejte realloc(x,0) místo free(x). Není to totéž.
- malloc alokuje místo v paměti. Pokud už máte nějaký uzel v paměti vytvořený a potřebujete si na něj jen udělat další referenci, tak malloc znovu nevolejte. Jen deklarujte nový pointer na tentýž uzel. Tj následující kód je špatně:

```
node* x;
x = (node*)malloc(sizeof(node));
x = t->root;
```

- Pokud kompilátor vypisuje varování, tak je něco špatně. Neignorujte zprávy kompilátoru.
- Odevzdávejte kódy splňující požadavky zadání (pojmenování procedur, funkcionalita).
- Odevzdávejte pouze kódy a ne celé projekty z použitého IDE.
- U odevzdaných kódů je vhodné nechat ukázku použití (v mainu, v komentářích, ...). Zejména pokud používáte jiné hlavičky procedur než bylo zadáno nebo pokud nebyly zadány konkrétní hlavičky.
- Vždy kontrolujte, jestli vstupní argumenty procedur dávají smysl. Tj. když někdo chce například 15. prvek z 5prvkového pole, tak by to program měl ošetřit a ne spadnout.

- Snažte se, aby kód dával smysl. Následující je špatně:

```
if(t1->root->right_child == NULL){}
else{
 t1->root = t1->root->right_child;
 print_in_order(t1);
 t1->root = t2->root;
}
```

- Takovýto příkaz nedává smysl:

```
malloc(length(*seznam) + ( 1 * sizeof(uzel)));
```

  Pouze alokuje místo, ke kterému pak nemáte jak přistoupit, protože si nikam neukládáte jeho adresu. Navíc je otázka, co přesně znamená výraz v argumentu.