Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Dátové štruktúry a algoritmy

Zadanie 1 – správa pamäti

LS 2020/2021

Samuel Kováč

Študijný program: B-INFO4

Vyučujúci: Ing. Lukáš Kohútka, PhD.

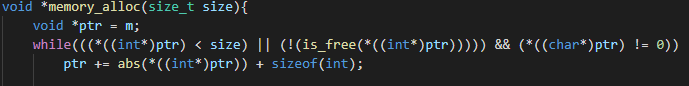
Cvičenia: Pondelok 9:00

Marec 2021

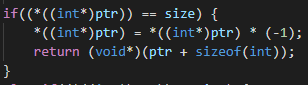
**Vypracovanie zadania**

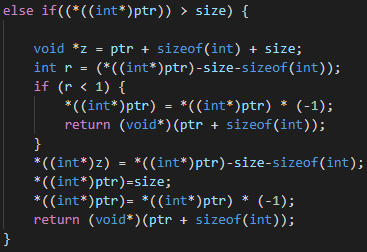
Hlavička v mojom riešení sa skladá z 4 bytov (integer), pričom voľnosť bloku pamäte určujem podľa znamienka čísla v hlavičke. Ak je dané číslo kladné, znamená to že je blok voľný a naopak. Pri inicializácií si tiež nastavím pätu pamäte na nulu (funguje ako ukončovací znak). Program využíva len jednu globálnu premennú ***m*** ktorá slúži ako ukazovateľ na začiatok pamäte. Algoritmus alokovania pamäte ktorý som použil, sa nazýva First fit. Jeho výhodou je rýchlosť, keďže alokuje hneď prvý voľný blok pamäte a nerobí ďalšie zložité porovnávania. Nevýhodou je ale zvýšená externá fragmentácia oproti algoritmu Best fit. First fit tiež obvykle zanecháva väčší voľný blok na konci pamäte. Na základe tohto algoritmu vieme určiť časovú [O(n)] a pamäťovú [O(n)] náročnosť programu.

**memory\_alloc**

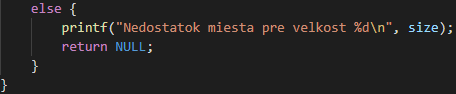
*Funkcia memory\_alloc má poskytovať služby analogické štandardnému malloc. Teda, vstupné parametre sú veľkosť požadovaného súvislého bloku pamäte a funkcia mu vráti: ukazovateľ na úspešne alokovaný kus voľnej pamäte, ktorý sa vyhradil, alebo NULL, keď nie je možné súvislú pamäť požadovanej veľkosť vyhradiť.*

Na začiatku funkcie pomocou while loopu nájdeme prvý voľný blok pamäte.

Prvá podmienka kontroluje či sa veľkosť nájdeného bloku rovná požadovanej veľkosti. Ak áno, nastaví znamienko hlavičky na (-) a funkcia vráti pointer na alokovaný blok, posunutý o veľkosť hlavičky.

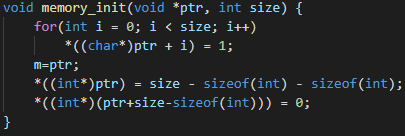


V prípade že veľkosť voľného bloku je väčšia ako požadovaná veľkosť, voľný blok je treba rozdeliť. Najprv skontroluje či po rozdelení bloku nezostane voľný blok veľmi malej veľkosti (5B – 4B hlavička) - v prípade že zvyšný blok je veľmi malý, voľný blok nebude rozdeľovať. Ak je zvyšok dostatočne veľký, funkcia voľný blok rozdelí a vráti pointer na alokovaný blok, posunutý o veľkosť hlavičky.

Nakoniec ak sa blok do teraz nepodarilo alokovať, znamená to že neexistuje voľný blok s dostatočnou veľkosťou a funkcia vráti NULL.

**memory\_init()**

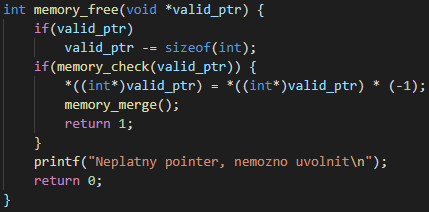
*Funkcia memory\_init slúži na inicializáciu spravovanej voľnej pamäte. Predpokladajte, že funkcia sa volá práve raz pred všetkými inými volaniami memory\_alloc, memory\_free a memory\_check. Viď testovanie nižšie. Ako vstupný parameter funkcie príde blok pamäte, ktorú môžete použiť pre organizovanie a aj pridelenie voľnej pamäte. Vaše funkcie nemôžu používať globálne premenné okrem jednej globálnej premennej na zapamätanie ukazovateľa na pamäť, ktorá vstupuje do funkcie memory\_init. Ukazovatele, ktoré prideľuje vaša funkcia memory\_alloc musia byť výhradne z bloku pamäte, ktorá bola pridelená funkcii memory\_init.*



Funkcia memory\_init najskôr nastaví celé pole pamäte na 1tky, potom nastaví globálny pointer (m) na začiatok pamäte. Nakoniec nastavíme do hlavičky veľkosť pamäte (odčítame veľkosť hlavičky a päty) a nastavíme pätu na 0 ako koncový znak poľa.

**memory\_free()**

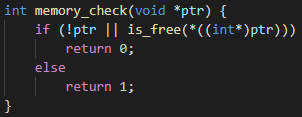
*Funkcia memory\_free slúži na uvoľnenie vyhradeného bloku pamäti, podobne ako funkcia free. Funkcia vráti 0, ak sa podarilo (funkcia zbehla úspešne) uvoľniť blok pamäti, inak vráti 1. Môžete predpokladať, že parameter bude vždy platný ukazovateľ, vrátený z predchádzajúcich volaní funkcie memory\_alloc, ktorý ešte nebol uvoľnený.*



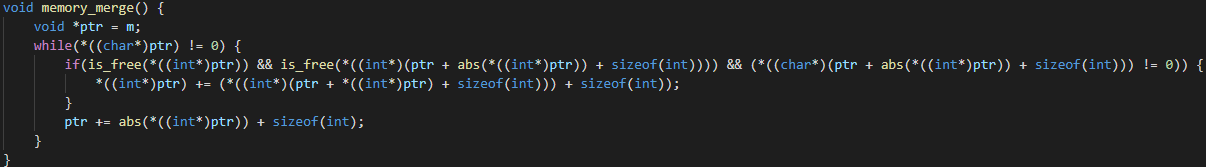
Funkcia memory\_free najprv odčíta veľkosť hlavičky od zadaného pointera, následne skontroluje pointer funkciou memory\_check a uvoľní ho pomocou otočenia znamienka hlavičky na kladnú hodnotu.

**memory\_check()**

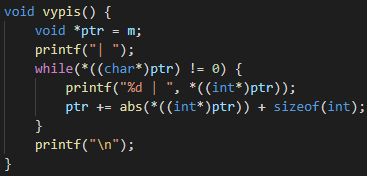
*Funkcia memory\_check slúži na skontrolovanie, či parameter (ukazovateľ) je platný ukazovateľ, ktorý bol v nejakom z predchádzajúcich volaní vrátený funkciou memory\_alloc a zatiaľ nebol uvoľnený funkciou memory\_free. Funkcia vráti 0, ak je ukazovateľ neplatný, inak vráti 1.*



Funkcia memory\_check skontroluje či sa pointer nerovná NULL alebo nie je voľný a podľa toho vráti 0/1

**memory\_merge()**

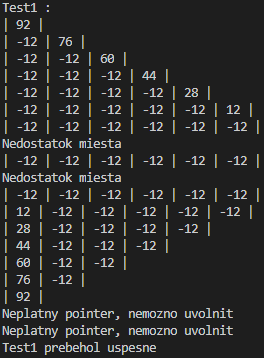
Pomocná funkcia ktorej úlohou je pospájať susediace voľné bloky. Ak sú za sebou 2 voľné bloky, k veľkosti prvého bloku pripočíta veľkosť druhého + veľkosť hlavičky druhého.

**vypis()**

Zabezpečuje vizuálne zobrazenie rozloženia pamäte. Zobrazí všetky bloky pamäte ako číslo predstavujúce veľkosť daného bloku (bez hlavičky). Ak je číslo kladné, daný blok je voľný a naopak.

**Testy**

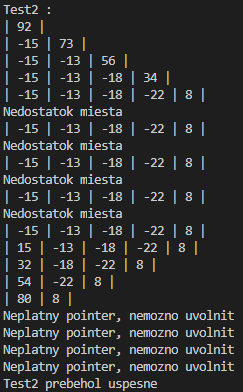
**1. test**

*Prideľovanie rovnakých blokov malej veľkosti (12 bytov) pri použití malých celkových blokov pre správcu pamäte (100bytov)*

Priebeh pamäte počas testu

*Úspešnosť alokovania oproti ideálnemu riešeniu :* ***62.5%***

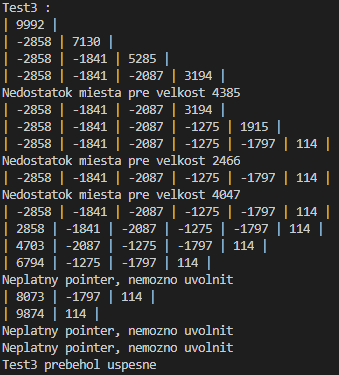
**2. test**

*Prideľovanie nerovnakých blokov malej veľkosti (náhodné veľkosti 8 až 24bytov)pri použití malých celkových blokov pre správcu pamäte (100 bytov)*

Priebeh pamäte počas testu

*Úspešnosť alokovania oproti ideálnemu riešeniu :* ***65 - 75%***

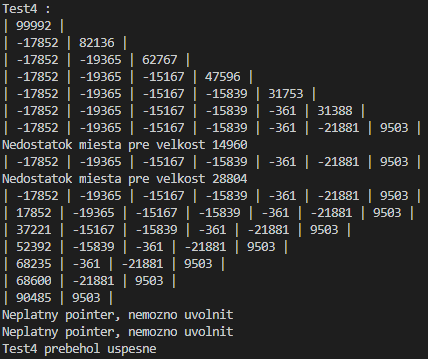
**3. test**

*Prideľovanie nerovnakých blokov väčšej veľkosti (veľkosti 500 až 5000 bytov) pri použití väčších celkových blokov pre správcu pamäte (10 000 bytov)*

Priebeh pamäte počas testu

*Úspešnosť alokovania oproti ideálnemu riešeniu :* ***100%***

**4. test**

*Prideľovanie nerovnakých blokov malých a veľkých veľkostí (veľkosti od 8 bytov do 50 000 bytov) pri použití väčších celkových blokov pre správcu pamäte (100 000 bytov).*

Priebeh pamäte počas testu

*Úspešnosť alokovania oproti ideálnemu riešeniu :* ***99.7 - 99.94%***

**Záver testovania**

Z testovania vyplýva že pamäť pridelená funkciou memory\_alloc nepresahuje pôvodný blok ani neprekrýva doteraz pridelenú pamäť, a dá sa úspešne uvoľniť funkciou memory\_free.