*Marcel Valent*

Slovenská technická univerzita v Bratislave  
Fakulta informatiky a informačných technológií  
Ilkovičova 2, 842 16 Bratislava 4  
[xvalentm@stuba.sk](mailto:xvalentm@stuba.sk)

8. marec 2021

ZADANIE 1 – SPRÁVCA PAMÄTI

Prednášajúci: Ing. Lukáš Kohútka, PhD.  
Cvičiaci: Mgr. Martin Sabo, PhD.  
Cvičenie: Utorok 16:00

Obsah

[**1.** **Zadanie** 3](#_Toc66132584)

[**2.** **Implementácia** 4](#_Toc66132585)

[**3.** **Funkcie** 5](#_Toc66132586)

[**3.1** **memory\_init** 5](#_Toc66132587)

[**3.2** **memory\_alloc** 5](#_Toc66132588)

[**3.3** **memory\_free** 5](#_Toc66132589)

[**3.4** **memory\_check** 6](#_Toc66132590)

[**4.** **Testovanie** 7](#_Toc66132591)

1. **Zadanie**

V štandardnej knižnici jazyka C sú pre alokáciu a uvoľnenie pamäti k dispozícii funkcie malloc a free. V tomto zadaní je úlohou implementovať vlastnú verziu alokácie pamäti.

Konkrétnejšie je vašou úlohou je implementovať v programovacom jazyku C nasledovné ŠTYRI funkcie:

**• void** \*memory\_alloc(unsigned int size);

**• int** memory\_free(void \*valid\_ptr);

**• int** memory\_check(void \*ptr);

**• void** memory\_init(void \*ptr, unsigned int size);

Vo vlastnej implementácii môžete definovať aj iné pomocné funkcie ako vyššie spomenuté, nesmiete však použiť existujúce funkcie malloc a free.

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.

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\_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\_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.

1. **Implementácia**

Moje riešenie využíva implicitný zoznam. Pri riešení zadania som využil najmä pointerovú aritmetiku a prehĺbil som si vedomosti práce s pointrami. Princípom implicitného zoznamu je, že všetky bloky sú pospájané do jedného zoznamu. Oproti explicitnému zoznamu sa líši v tom, že pri explicitnom zozname pointer prechádza len voľné bloky, ale v implicitnom zozname prechádza cez všetky(už aj alokované) bloky zoznamu. Na začiatku procesu máme veľkosť pamäte uloženú v hlavičke. Následne sa pri alokácii jednotlivé bloky pamäte alokujú do tohto veľkého bloku pamäte.

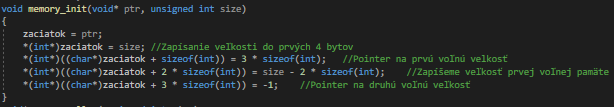
Diagram

Description automatically generated

Obr.1 Implicitný zoznam

1. **Funkcie**
   1. **memory\_init**

Funckia **memory\_init** je funkcia, ktorá sa zavolá práve raz pred ostatnými funkciami, pričom jej úlohou je inicializovať pamäť. Pointer globálnej premennej priradím na začiatok pamäte.

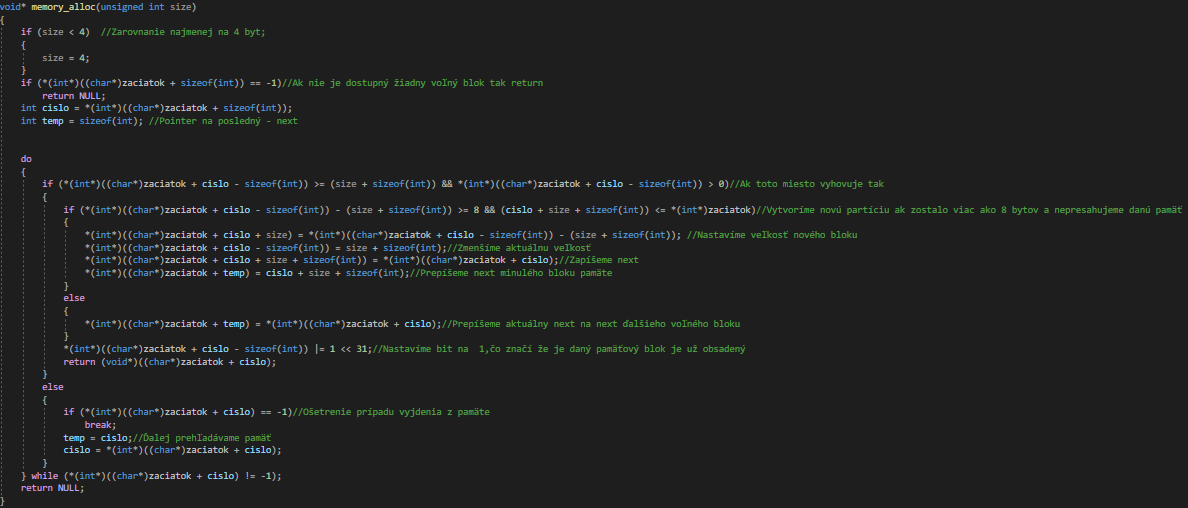


**Časová zložitosť:** O(n)

**Pamäťová zložitosť:** O(1)

* 1. **memory\_alloc**

Funkcia memory\_alloc má za úlohu alokovať miesto jednotlivým blokom do pamäte. Funkcia postupne prechádza zoznamom až kým nenájde miesto o vyhovujúcej veľkosti a alokuje toto miesto pre daný pamäťový blok, ktorý následne spojí so susednými blokmi v zozname.

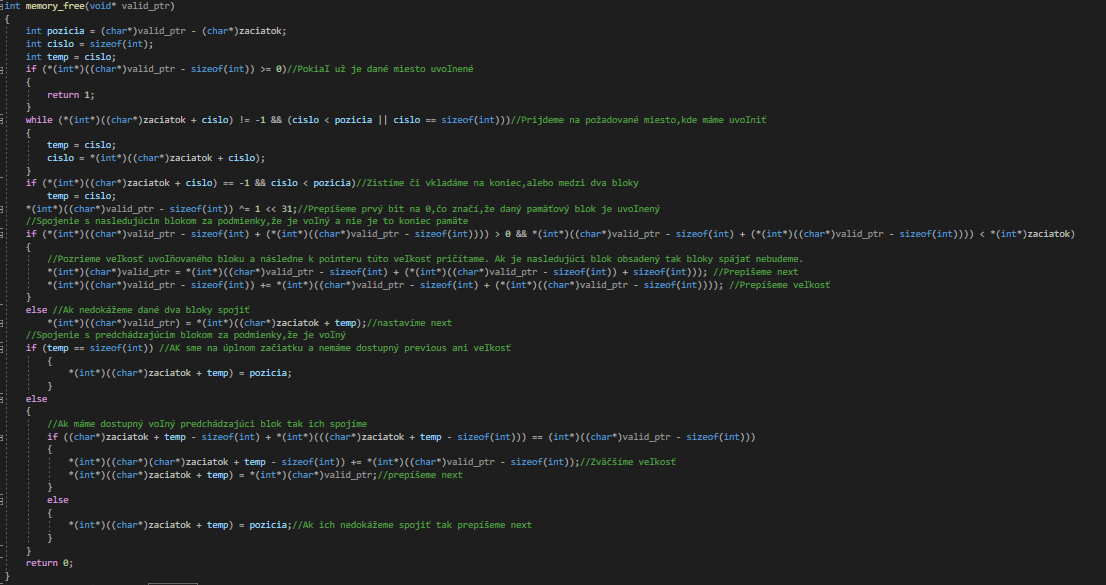


**Časová zložitosť**: O(n),pretože program totiž prechádza celým blokom pamäte

**Pamäťová zložitosť**: O(1)

* 1. **memory\_free**

Funkcia **memory\_free** slúži na uvoľnenie jednotlivých blokov pamäte z celkovej pamäte. Ak funkcia dostane ako argument platný pointer, tak potom sa vykoná funkcia **memory\_free**. Program využíva spájanie voľných blokov, ktoré sú vedľa seba. Program sa nastaví na blok, ktorý ide uvoľniť a zmení hodnotu v hlavičke na 0, čo značí že daný blok je uvoľnený.

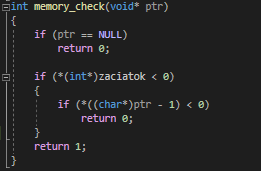


**Časová zložitosť:** O(1)

**Pamäťová zložitosť:** O(1)

* 1. **memory\_check**

Táto funkcia je pomocná pre funkciu **memory**­\_**check** a jej úlohou je overiť, že či pointer patrí do bloku pamäte a či je **platným pointrom** na niektorý z alokovaných blokov.

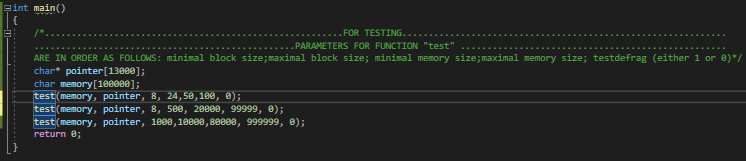


**Časová zložitosť:** O(1)

**Pamäťová zložitosť:** O(1)

1. **Testovanie**

Vo funkcii **main**, ktorá je na nasledujúcom obrázku sme si skúsili tri testovacie scenáre.



1. scenár: Alokovali sme bloky o veľkosti 8 – 24 bytov pri použití malého celkového bloku pamäte (91bytov).



Tu nám úspešne alokovalo 4 zo 6 požadovaných blokov a celkový počet alokovaných bytov bol 71.1%

2. scenár: Alokovali sme bloky o veľkosti 8 – 500 bytov pri použití celkového bloku pamäte o veľkosti 20041 bytov.



Tu nám úspešne alokovalo 81 z 86 požadovaných blokov a celkový počet alokovaných bytov bol 98.27%

3.scenár: Alokovali sme bloky o veľkosti 1000 – 10000 bytov pri použití celkového bloku pamäte o veľkosti 80041bytov.



Tu nám úspešne alokovalo 15 z 15 požadovaných blokov a celkový počet alokovaných bytov bol 100%