Slovenská technická univerzita v Bratislave
Fakulta Informatiky a Informačných Technológií

Datové štruktúry a algoritmy

Zadanie 1 – Správca pamäti

Emma Macháčová

Meno cvičiaceho : Ing. Dominika Dolhá
Čas cvičení : pondelok 9:00
Dátum vytvorenia : 03. marec 2021

Emma Macháčová
ID: 103037

Obsah

Emma Macháčová
ID: 103037

Cieľ práce

Cieľom projektu bolo implementovať v programovacom jazyku C štyri funkcie pre alokáciu a uvoľnenie pamäti :

- `void memory_init (void *ptr, unsigned int size);`
- `void *memory_alloc (unsigned int size);`
- `int memory_check (void *ptr);`
- `int memory_free (void *valid_ptr);`

Funkcia memory_init slúži na inicializáciu spravovanej voľnej pamäte. Funkcia sa volá práve raz pred všetkými inými volaniami memory_alloc, memory_free a memory_check. Ako vstupný parameter funkcie príde blok pamäte, ktorú môžete použiť pre organizovanie a aj pridelenie voľnej pamäte.

Funkcia memory_alloc má poskytovať služby analogické štandardnému malloc. Vstupné parametre sú veľkosť požadovaného súvislého bloku pamäte a funkcia vráti ukazovateľ na úspešne alokovaný kus voľnej pamäte, ktorý sa vyhradil, alebo NULL, keď nie je možné súvislú pamäť požadovanej veľkosť vyhradiť.

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_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.

Emma Macháčová
ID: 103037

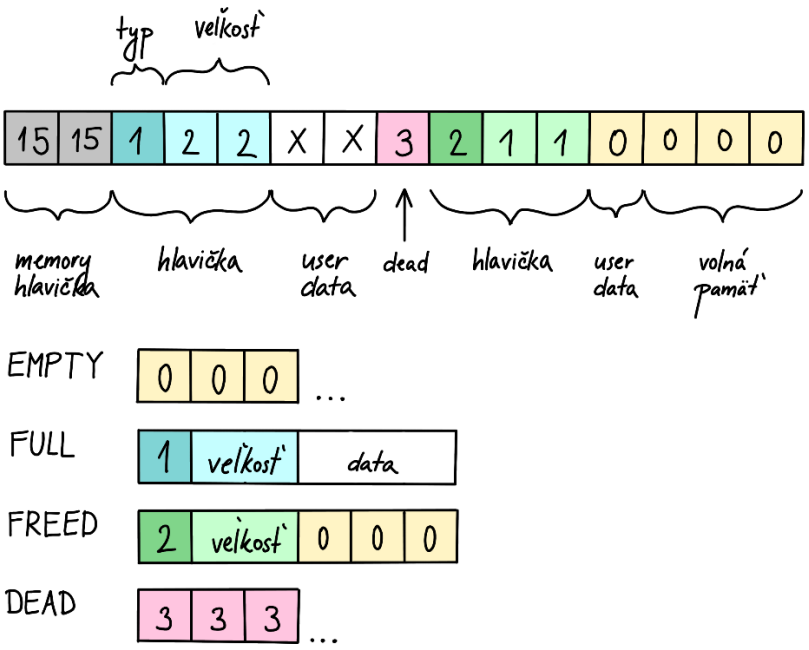
Opis riešenia

V projekte pracujem so štyrmi **rôznymi typmi blokov pamäte**, pre ktoré mám definované označenia (pseudo-enum). Tieto informácie sa ukladajú v hlavičke blokov. Pre lepšiu prehľadnosť kódu sú definované aj rozmery hlavičiek.

```
9   // pseudo-enum
10  #define EMPTY 0
11  #define FULL 1
12  #define FREED 2
13  #define DEAD 3
14
15  // format
16  #define HLAVICKA 3
17  #define MEMORY_HLAVICKA 2
```

EMPTY (0) sú nepoužité, individuálne prázdne bloky (byty) pamäte. **FULL** (1) sú segmenty pamäte, ktoré obsahujú aktívne užívateľské informácie (alokovaná a zatiaľ neuvoľnená pamäť). **FREED** (2) sú segmenty, ktoré obsahovali užívateľské dáta, a už boli uvoľnené funkciou memory_free. **DEAD** (3) sú individuálne bloky (byty), vznikajúce ako fragmenty nevyužitej pamäte. Bloky pamäte označené ako EMPTY, FREED a DEAD sú pripravené na (opätovnú) alokáciu pre užívateľa.

Spravovaná pamäť počas behu programu má nasledovnú **štruktúru**:

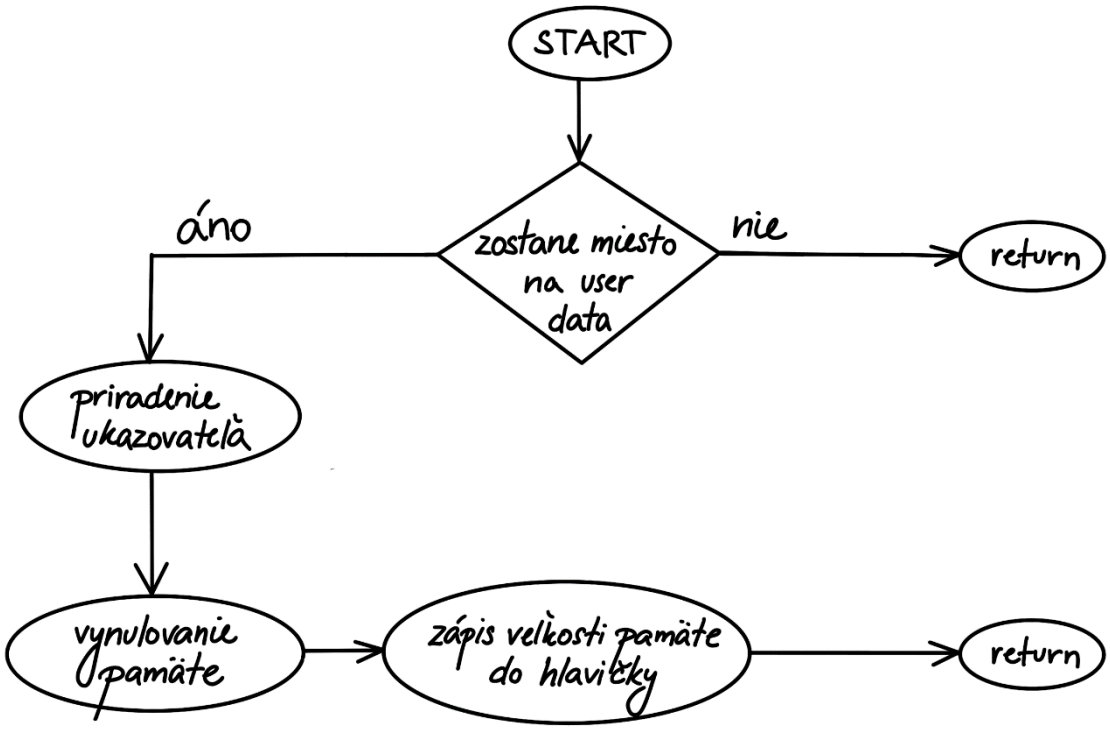


Keďže pracujem s pamäťou priamo, a využívam v hlavičke uint16 číslo, ktoré sa uloží na dva „chary“, tak bude nastávať konverzia pointrov ktorá bude vyhadzovať warning Wincompatible-pointer-types. Keďže ale takáto konverzia je pod kontrolou a je kompatibilná, tak som si dovolila tento warning potlačiť pragmou na tých konkrétnych miestach. Táto pragma potláča warning len na konkrétnom úseku kódu. V iných častiach kódu by túto situáciu compiler vyhodnotil ako chybnú.

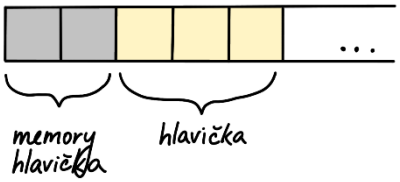
```
38  #pragma GCC diagnostic push
39  #pragma GCC diagnostic ignored "-Wincompatible-pointer-types"
40  uint16_t *pomocny_pointer = GLOB_MEM; // pomocný pointer, na pretypovanie GLOB
41  #pragma GCC diagnostic pop
42  *pomocny_pointer = (uint16_t) size; // veľkosť pamäte v prekonvertovanom tvare
```

Emma Macháčová
ID: 103037

Memory init



Funkcia **memory_init** slúži na prvotné inicializovanie poľa pamäte.

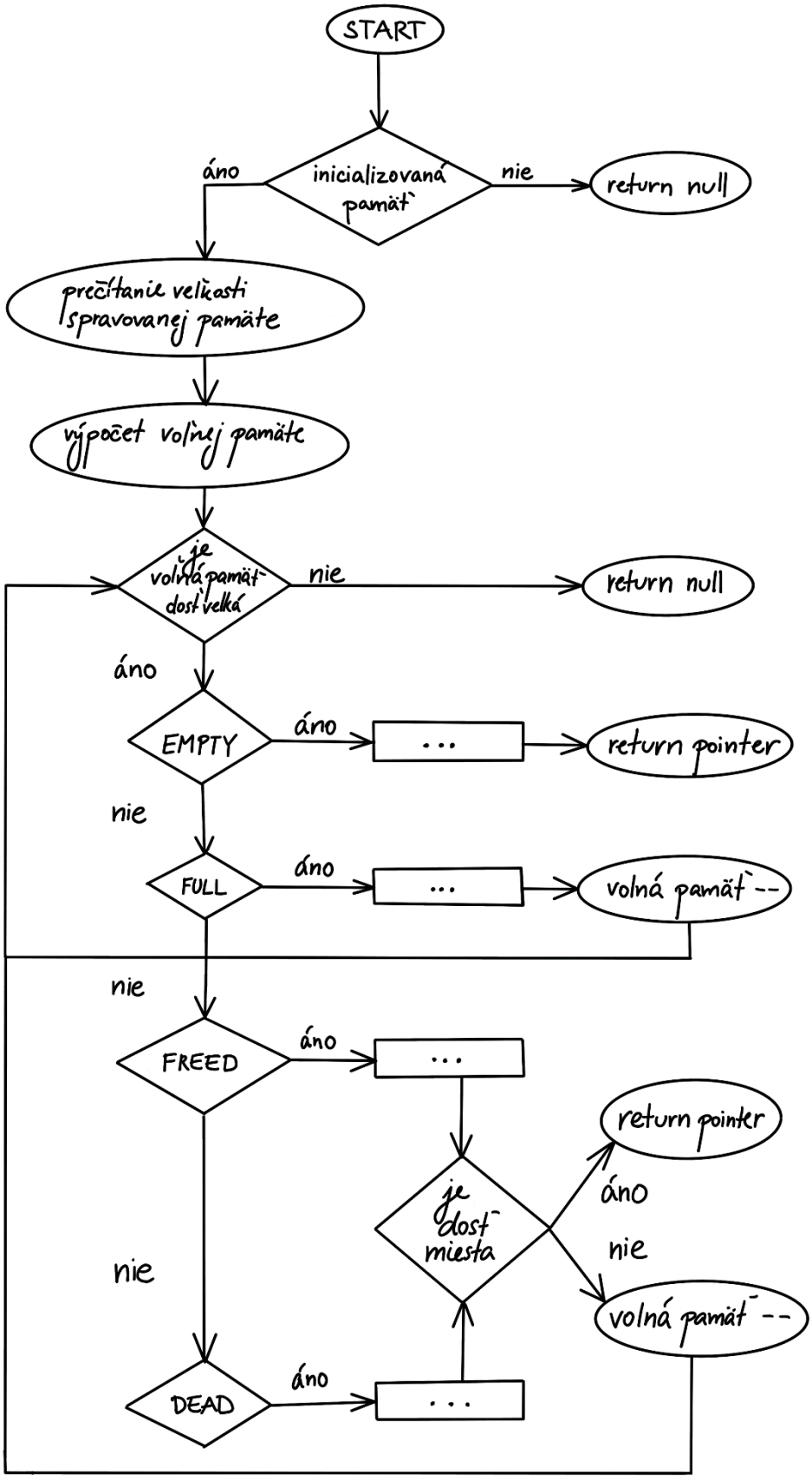


Ako prvé skontroluje, či je dostupná **voľná pamäť dostatočne veľká** na vytvorenie hlavnej hlavičky poľa celej dostupnej pamäte (MEMORY_HLAVICKA), a či po inicializovaní bude možné alokovať pamäť aj pre užívateľa (kontroluje aj potrebné miesto na hlavičku dát užívateľa). Ak táto podmienka nie je splnená, pamäť sa neinicializuje.

Ak je podmienka pamäte splnená, funkcia **priradí ukazovateľ** vyhradenej pamäte a globálny ukazovateľ (GLOB_MEM). Pole pamäte sa potom vynuluje, a do hlavičky sa zapíše **informácia o veľkosti poľa** vo formáte uint16_t, pre optimalizáciu veľkosti hlavičky.

Emma Macháčová
ID: 103037

Memory alloc



Funkcia memory_alloc alokuje pre užívateľa pamäť požadovanej veľkosti.

Na začiatku skontroluje, či bola pamäť **inicializovaná** vo funkcii memory_init. Ak áno, prečíta veľkosť spravovanej pamäte z memory hlavičky, a do premennej volna_pamat uloží hodnotu veľkosti pamäte z ktorej sa postupne odrátava. Je to potencionálna voľná pamäť, ktorá je k dispozícií na alokovanie. Premenná „pozicia“ slúži na **indexovanie** poľa spravovanej pamäte.

```
55  // získanie veľkosti pamate
56  #pragma GCC diagnostic push
57  #pragma GCC diagnostic ignored "-Wincompatible-pointer-types"
58  uint16_t *velkost_pamate = GLOB_MEM; // pomocný pointer, na pretypovanie GLOB
59  #pragma GCC diagnostic pop
60
61  // aktualne miesto v pamati; 2 lebo na zaciatku mame hlavicku celeho pola
62  uint16_t volna_pamat = *(velkost_pamate) - MEMORY_HLAVICKA;
63  int pozicia = MEMORY_HLAVICKA; // od indexu 2 začína pamäť
```

Pre optimálne využívanie pamäte sú inicializované premenné free_ptr, pocet_free a velkost_free. Ukazovateľ „**free_ptr**“ uchováva adresu na prvý blok voľnej pamäte typu FREED alebo DEAD ktorý funkcia stretne pri snahe alokovať nový blok pamäte. Premenná „**pocet_free**“ si pamätá počet segmentov typu FREED. Slúži na výpočet veľkosti dostupnej pamäte na prepísanie (**velkost_free**), pretože segmenty FREED obsahujú aj hlavičky, ktoré sa pri ich zlúčení môžu využiť na uchovanie užívateľských informácií (na rozdeľ od blokov DEAD, ktoré sú samostatné a bez hlavičky).

```
65  // pre využívanie fragmentov volnej pamäte
66  char *free_ptr = NULL; // pointer na prvú medzeru
67  int pocet_free = 0; // pocet medzier
68  int velkost_free = 0; // velkost pamate medzier
```

Jadro funkcie prebieha vo while cykle, ktorý postupne jedenkrát prejde pole spravovanej pamäte, a hľadá, či nájde dosť priestoru pre alokovanie nového bloku pre užívateľa o želanej veľkosti (size). Cyklus skončí keď funkcia príde na koniec pamäte, alebo ak sa podarí alokovať požadovaný blok.

```
76  // kym nie sme na konci pamate
77  while ( 0 <= (volna_pamat + velkost_free) && (volna_pamat + velkost_free) >= size ) {
```

```
// while ( 0 <= (volna_pamat + velkost_free - HLAVICKA) && (volna_pamat + velkost_free - HLAVICKA) >= size ) {
```

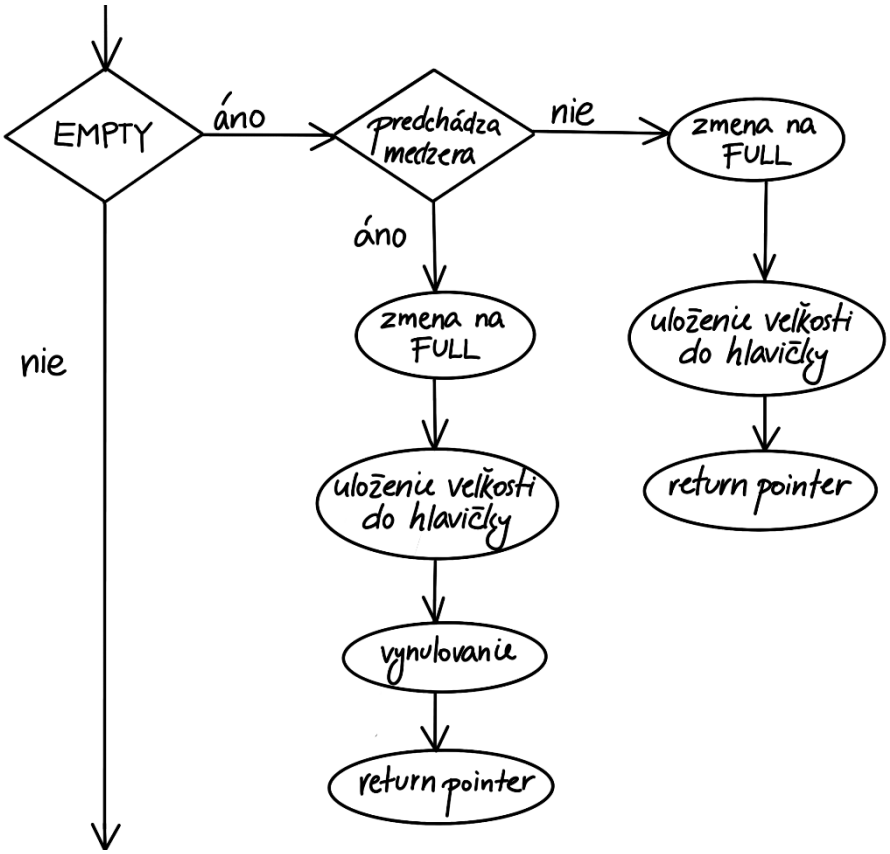
Podmienka zámerne nekontroluje na tomto mieste aj to, či by sa zmestila aj hlavička bloku, pretože o tú veľkosť hlavičky skôr by cyklus skončil, a v niektorých scénaroch by sa nevyužila pamäť kompletne. Podmienka kontrolujúca nepretekanie poľa je implementovaná ďalej v kóde.

Pri tomto prechádzaní funkcia pracuje v štyroch rôznych „**módoch**“, v závislosti od toho aký blok prečíta:

- GLOB_MEM[pozicia] == EMPTY
- GLOB_MEM[pozicia] == FULL
- GLOB_MEM[pozicia] == FREED
- GLOB_MEM[pozicia] == DEAD

Emma Macháčová
ID: 103037

Case A: GLOB_MEM[pozicia] == EMPTY

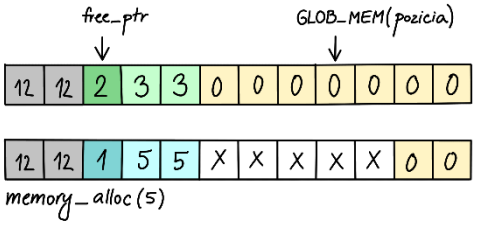


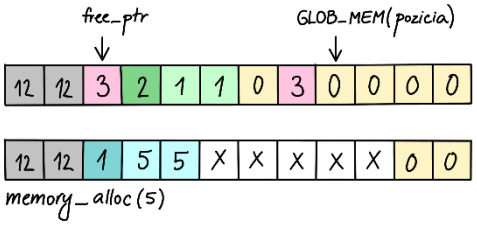
Funkcia najskôr skontroluje, či má **dostatok pamäte k dispozícií** na to, aby mohla alokovať blok tak, aby celý bol vo vnútri spravovanej pamäte.

```
82        // aby na konci pamate neprislo k leaku - treba kontrolovat
83        if ( 0 > (volna_pamat + velkost_free - HLAVICKA) || (volna_pamat + velkost_free - HLAVICKA) < size ) {
```

V tomto prípade funkcia narazí v poli na blok typu **EMPTY**. Buď prešiel za už alokované bloky a pamäť je voľná až do konca, alebo ešte nebola alokovaná žiadna pamäť.

Ak už pamäť alokovaná bola, je možné že blok EMPTY **predchádza „medzera“.** To môže byť buď segment FREED, blok DEAD, alebo určitá ich kombinácia. V takom prípade pamäť nealokuje od aktuálnej pozície, ale **od pozície free_ptr.** V prípade ak predchádzala medzera tiež dôjde k vynulovaniu pamäte ktorá bude k dispozícií používateľovi, ale tento krok je možné vynechať (nie je nutne potrebný).





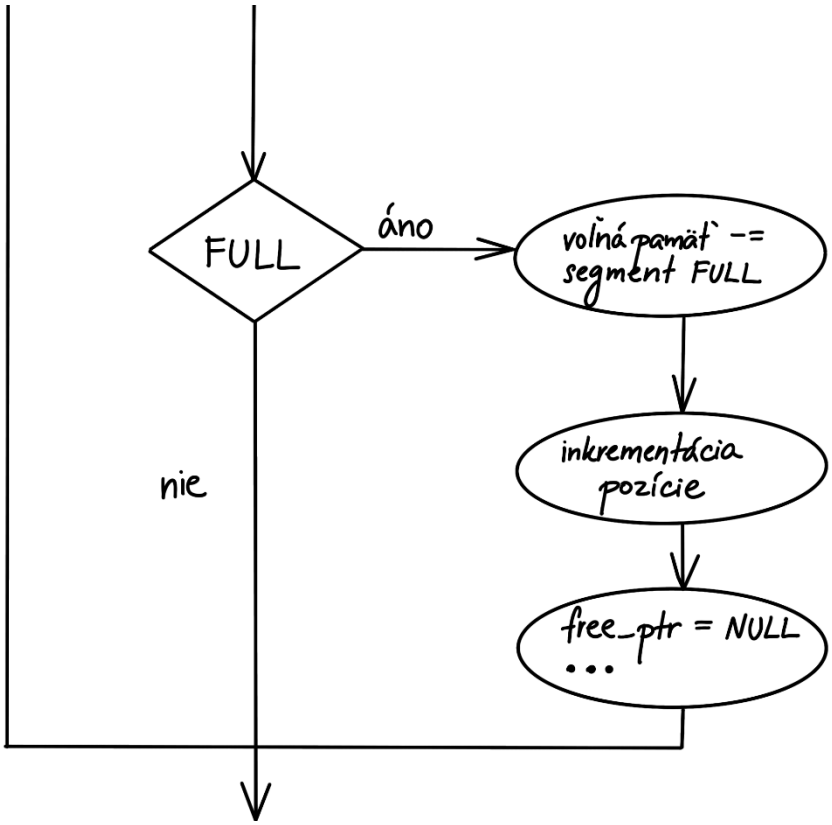
Ak je free_ptr = NULL, znamená to, že bloku EMPTY priamo **nepredchádza „medzera“** – buď žiadna neexistuje alebo bola prerušená segmentom FULL.

```
82        // ak nepredchádza blok uvolnenenj pamate
83        if (free_ptr == NULL) {
84          GLOB_MEM[pozicia] = FULL; // zmena stavu bloku pamate
```

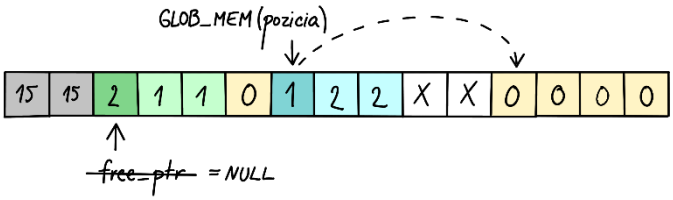
V oboch prípadoch, funkcia pokračuje **zmenou informácie o typu a veľkosti segmentu** v jeho hlavičke (rozdiel je len v tom, kde tá hlavička začína).

Emma Macháčová
ID: 103037

Case B: GLOB_MEM[pozicia] == FULL



V tomto prípade funkcia narazí na segment typu **FULL**. V tejto iterácií teda k alokovaniu určite nedôjde, a je potrebné **posunúť sa v poli o veľkosť zabraného segmentu**.



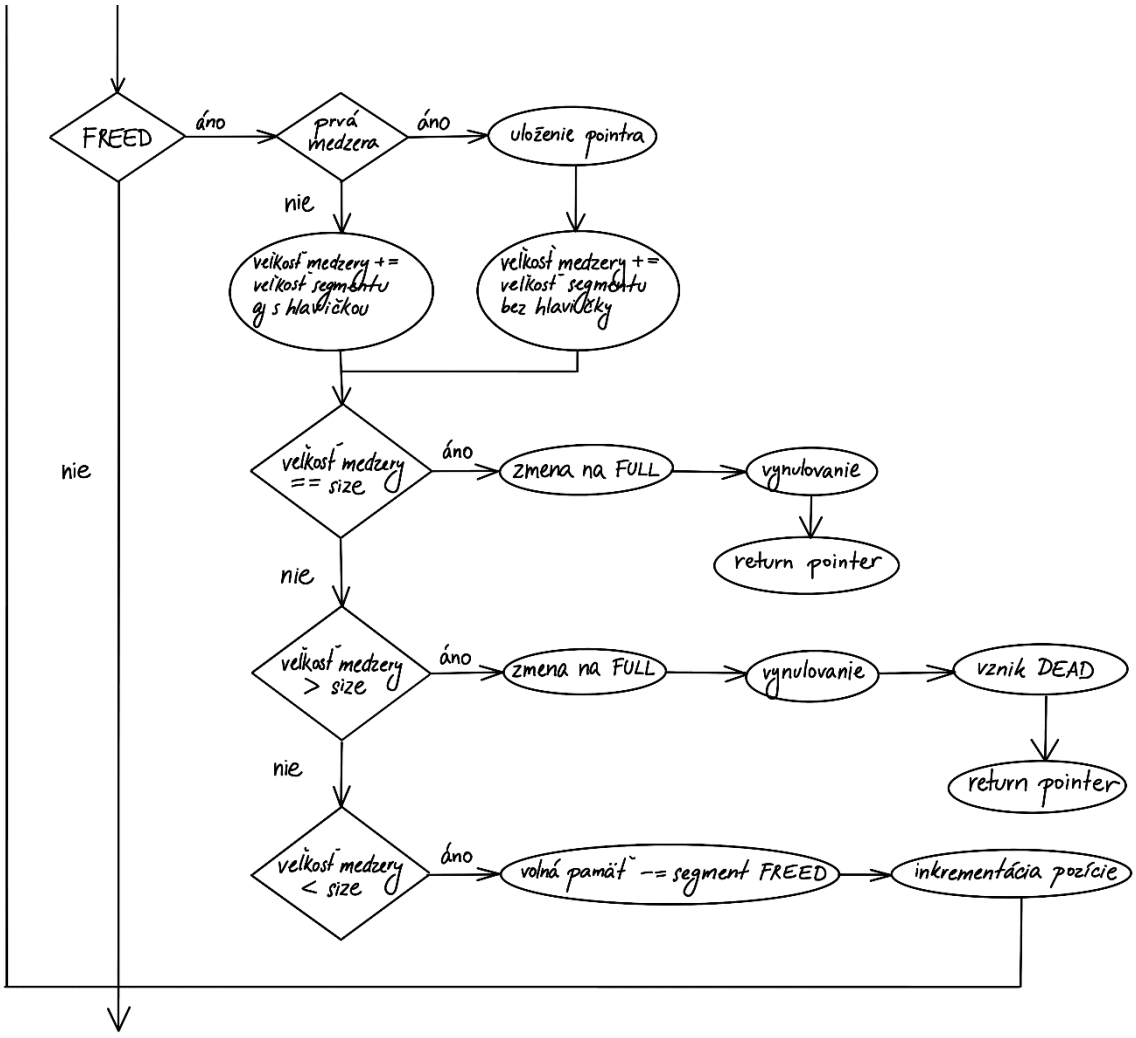
Od zostávajúcej voľnej pamäte (volna_pamat) sa odčíta celá veľkosť segmentu FULL (aj s jeho hlavičkou). O takú istú hodnotu (veľkosť bloku aj s hlavičkou) sa inkrementuje pozícia.

```
144         free_ptr = NULL;
145         pocet_free = 0;
146         velkost_free = 0;
```

Vynulujú sa premenné zabezpečujúce prácu s fragmentami pamäte, pretože ak do tejto pozície aj nejaké medzery boli, nemali dostatočnú veľkosť na to aby sa tam nová pamäť alokovala.

Funkcia ďalej pokračuje v ďalšej iterácií while cyklu.

Case C: GLOB_MEM[pozicia] == FREED



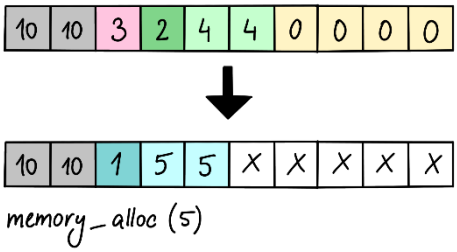
V tomto prípade funkcia narazí na segment typu **FREED**.

Pokiaľ ide o **prvú „medzeru“** ukazovateľ sa priradí do free_ptr, a velkost_free sa nastaví na veľkosť segmentu. Nepriráta sa k nej hlavička, lebo tá sa využije ako je.

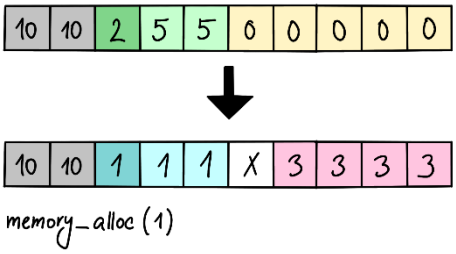
Ak nejde o prvú, velkost_free sa zvýši o veľkosť segmentu FREED, plus jeho hlavička, lebo pri zlučovaní FREED blokov sa priestor ktorý zaberali hlavičky využije na užívateľské dáta.

Ďalej program **porovná veľkost medzery** (velkost_free) s veľkosťou pamäte, ktorú žiada užívateľ (size) :

- Pokiaľ sa **zhodujú**, typ bloku sa zmení na FULL a do hlavičky sa zapíše informácia o veľkosti bloku (je potrebné ju zapísať, lebo aj keď veľkosť sedí presne, môže ísť o viaceré zlúčené bloky). Pamäť sa vynuluje a funkcia vráti ukazovateľ.

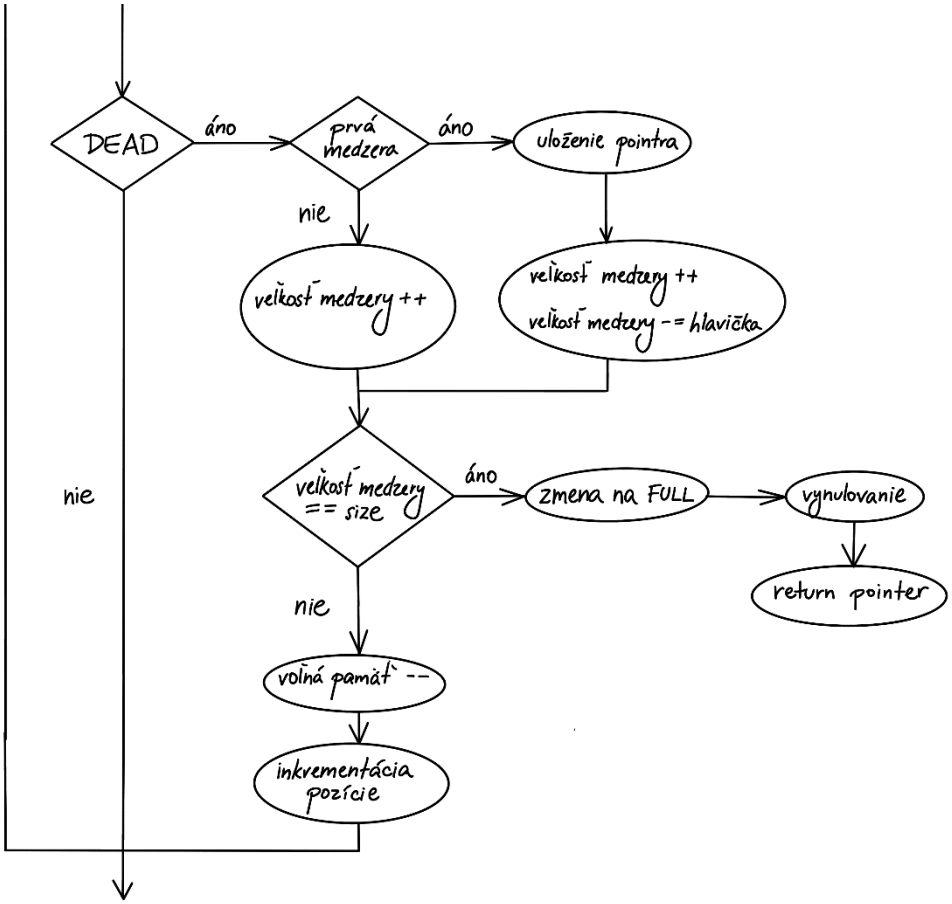


- Pokiaľ je voľného miesta **viac** ako žiada užívateľ, zmení sa typ bloku v hlavičke na FULL, zapíše sa informácia o veľkosti, časť pamäte pre užívateľa sa vynuluje, a zvyšná pamäť sa pripíše na DEAD. Funkcia vráti ukazovateľ.



- Ak je miesta **málo,** odčíta sa veľkosť bloku od voľnej pamäte, a inkrementuje sa pozícia. Program pokračuje v ďalšej iterácií while cyklu.

Case D: GLOB_MEM[pozicia] == DEAD



V tomto prípade funkcia narazí na blok typu **DEAD**.

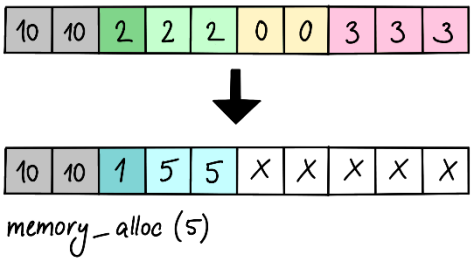
Pokiaľ ide o **prvú „medzeru“** ukazovateľ sa priradí do free_ptr, a velkost_free sa nastaví na 1 – hlavička, pretože na rozdiel od FREED segmentu, nie je „predchystané“ miesto na hlavičku.

Ak nejde o prvú medzeru, velkost_free sa zvýši o jeden blok.

Emma Macháčová
ID: 103037

Ďalej program **porovná veľkost medzery** (velkost_free) s veľkosťou pamäte, ktorú žiada užívateľ (size) :

- Pokiaľ sa **zhodujú**, typ bloku sa zmení na FULL a do hlavičky sa zapíše informácia o veľkosti bloku (je potrebné ju zapísať, lebo aj keď veľkosť sedí presne, môže ísť o viaceré zlúčené bloky). Pamäť sa vynuluje a funkcia vráti ukazovateľ.

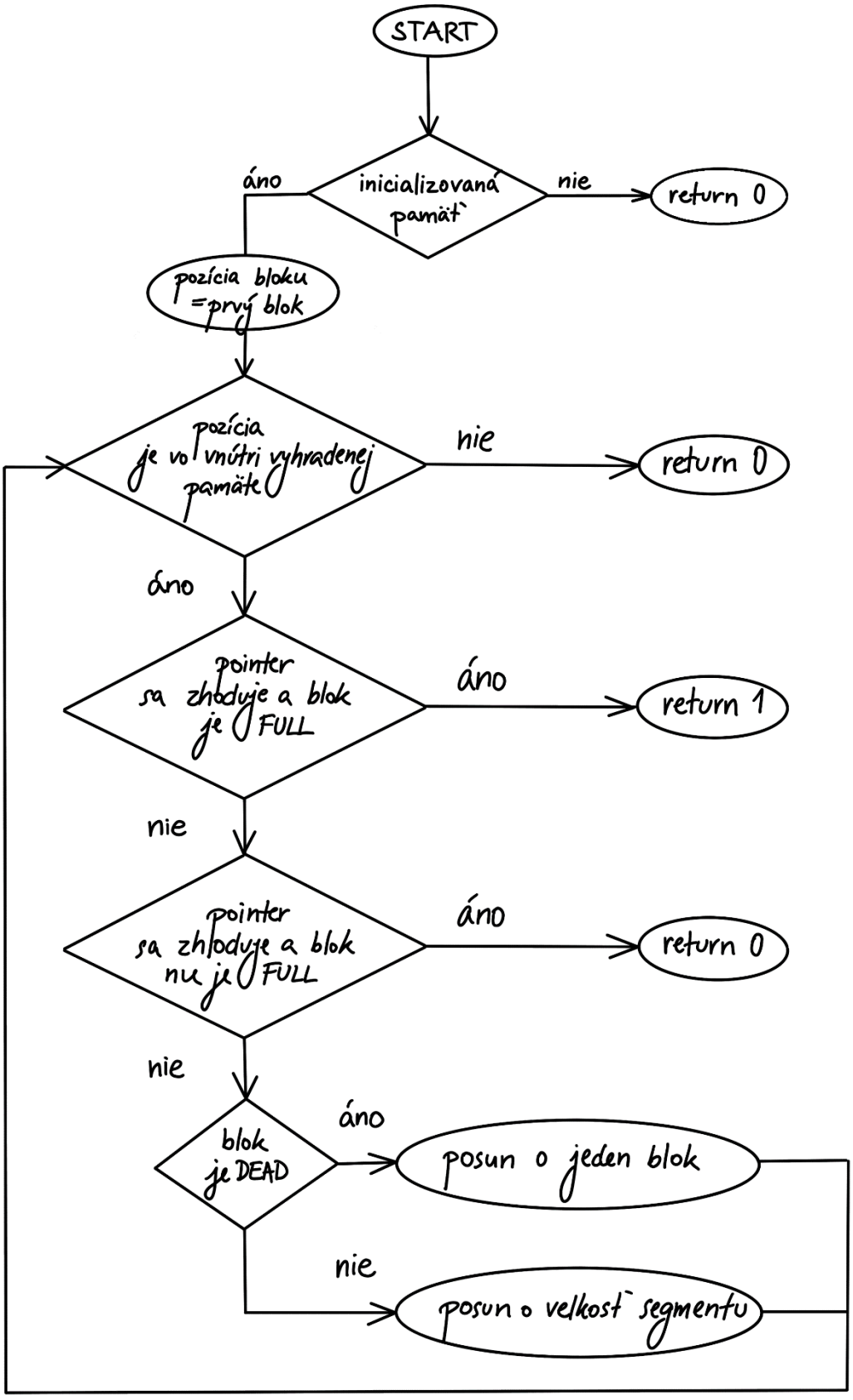


- Situácia, že by bolo **miesta viac nenastane**, lebo pri DEAD blokoch pribúda o jeden blok.

- Ak je miesta **málo,** odčíta sa veľkosť bloku od voľnej pamäte, a inkrementuje sa pozícia. Program pokračuje v ďalšej iterácií while cyklu.

Emma Macháčová
ID: 103037

Memory check



Funkcia **memory_check** kontroluje, či je zadaný ukazovateľ platný.

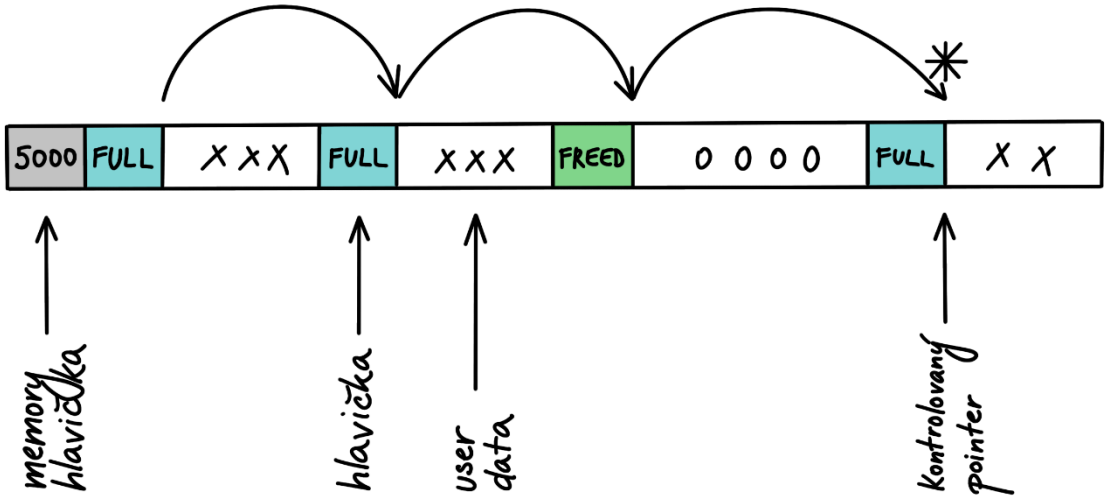
Ako prvé skontroluje, či bola pamäť **inicializovaná** (či funkcia memory_init prebehla a prebehla úspešne). Ak nebola, funkcia sa rovno ukončí a vráti hodnotu 0.

Pokiaľ bola pamäť **inicializovaná úspešne**, do premennej „kontrolovany“ sa priradí zadaný ukazovateľ ktorý sa ide kontrolovať, a do premennej „pozicia_bloku“ sa uloží veľkost memory hlavičky (t.j. 2), pretože od indexu 2 začína prvá hlavička užívateľských dát.

```
372    char *kontrolovany = ptr; // pretypovanie na pointer typu char
373    unsigned int pozicia_bloku = MEMORY_HLAVICKA;
```

Z memory hlavičky sa potom prečíta veľkosť spravovanej pamäte.

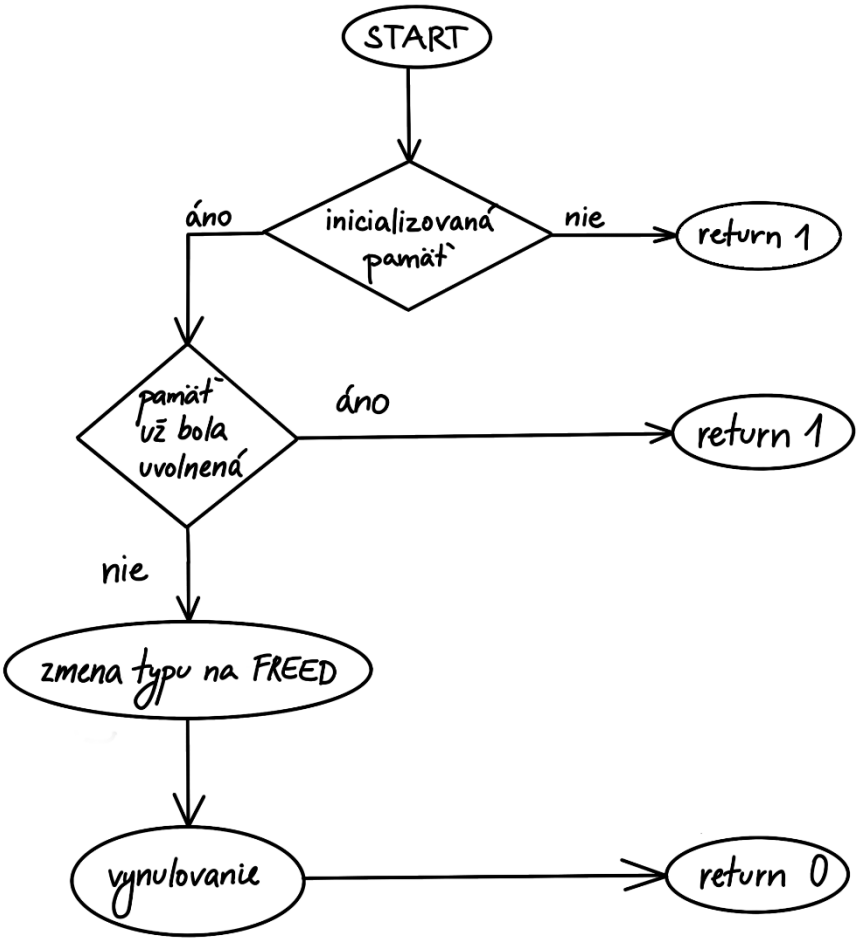
Následne vo while cykle, ktorý kontroluje či je pozícia bloku stále v rámci spravovanej pamäte, porovnávame či **prechádzaním po hlavičkách** segmentov prídeme na blok so zhodujúcim sa ukazovateľom, a či je tento blok typu FULL, a teda je aktívny a nebol ešte uvoľnený funkciou memory_free.



Neprechádzame priamo na zadaný ukazovateľ, pretože užívateľské dáta by mohli **napodobňovať formát platnej hlavičky**. Týmto spôsobom sa overí, že kontrolovaný blok je **alokovaný korektne**, tak ako aj bloky pred ním.

Emma Macháčová
ID: 103037

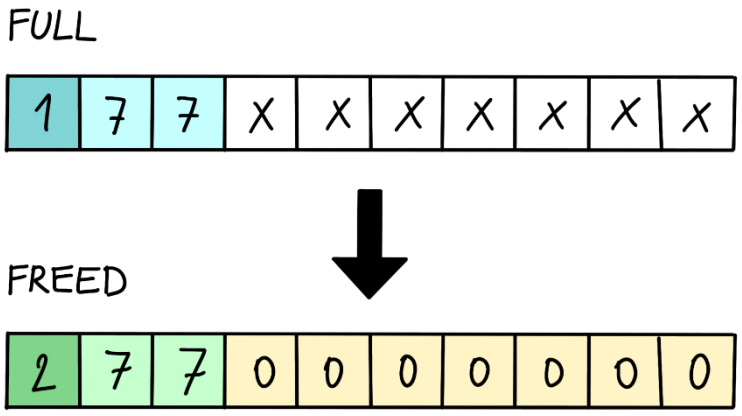
Memory free



Funkcia memory_free slúži na uvoľnenie zabraného bloku pamäti (FULL).

Najskôr skontroluje, či bola spravovaná pamäť **inicializovaná** vo funkcii memory_init. Pokiaľ inicializovaná bola, kontroluje či je daný blok **typu FULL** a či ešte nebol uvoľnený, aby sa predišlo zbytočným operáciám.

Ak je možné vykonať uvoľnenie pamäte, prepíše sa v hlavičke typ bloku **z FULL na FREED**, a vynulujú sa užívateľské informácie. Tento krok pre chod programu nie je potrebný, nemení nič na funkčnosti, pretože program sa nikdy nedostane do takého stavu, že by čítal užívateľské dáta, keďže sa pohybuje iba po hlavičkách – môžem si teda dovoliť vracať užívateľovi bloky typu EMPTY.



Prípad, kedy by do bloku FREED o veľkosti 7 bola zapísaná informácia s veľkosťou napríklad 2, je ošetrený vo funkcii memory_alloc.

Emma Macháčová
ID: 103037

Testy

Scenár 0 – ukážkový test, prideľovanie rôznych blokov malej veľkosti (veľkosti 8 až 24 bytov) pri použití malých celkových blokov pre správcu pamäte (do 50 bytov, do 100 bytov, do 200 bytov)

Ukážkový test	

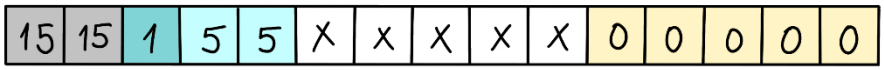
char memory[15];
memory_init (memory, 15);
 	
char *pointer1 = (char *) memory_alloc(5);
memory_check(pointer1);
char *pointer2 = (char *) memory_alloc(2);
memory_check(pointer2);
 	
memory_free(pointer1);
pointer1 = (char *) memory_alloc(2);
 	
memory_free(pointer2);
pointer2 = (char *) memory_alloc(4);
 	
memory_free(pointer2);
memory_free(pointer1);
 	
pointer1 = (char *) memory_alloc(6);
pointer2 = (char *) memory_alloc(1);
 	
memory_free(pointer2);
memory_free(pointer1);
 	
pointer1 = (char *) memory_alloc(7);
	ALOKÁCIA (1) PREBEHLA ÚSPEŠNE,
	alokovaných 5 bitov od pozície 2
	zostávajúca pamäť 5 bitov
	Pointer je aktívny
	
	ALOKÁCIA (1) PREBEHLA ÚSPEŠNE,
	alokovaných 2 bitov od pozície 10
	zostávajúca pamäť 0 bitov
	Pointer je aktívny
	
	Pamäť uvolnená úspešne
	ALOKÁCIA (4) PREBEHLA ÚSPEŠNE,
	alokovaných 2 bitov od pozície 2
	
	Pamäť uvolnená úspešne
	ALOKÁCIA (4) PREBEHLA ÚSPEŠNE,
	alokovaných 4 bitov od pozície 10
	
	Pamäť uvolnená úspešne
	Pamäť uvolnená úspešne
	
	ALOKÁCIA (4) PREBEHLA ÚSPEŠNE,
	alokovaných 6 bitov od pozície 7
	ALOKÁCIA (5) PREBEHLA ÚSPEŠNE,
	alokovaných 1 bitov
	
	Pamäť uvolnená úspešne
	Pamäť uvolnená úspešne
	
	ALOKÁCIA (4) PREBEHLA ÚSPEŠNE,
	alokovaných 7 bitov od pozície 11

Výsledná pamäť:	
15 0 1 7 0 0 0 0 0 0 0 0 3 3 3	

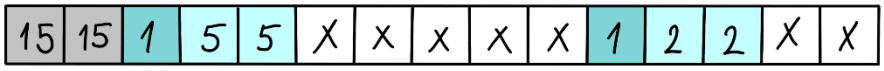
Emma Macháčová
ID: 103037

Priebeh pamäte počas ukážkového testu

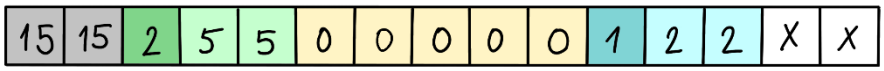
memory_alloc (5)



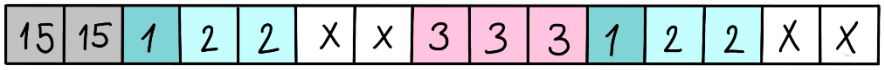
memory_alloc (2)



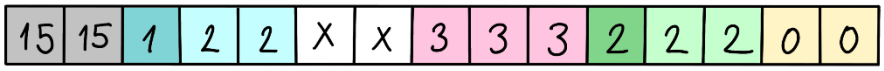
free (prvý blok)



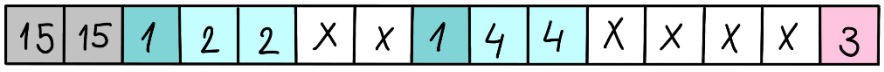
memory_alloc (2)



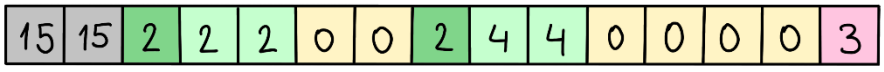
free (druhý blok)



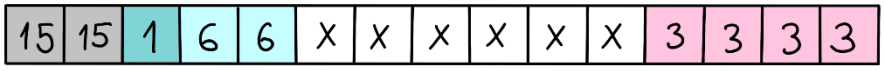
memory_alloc (4)



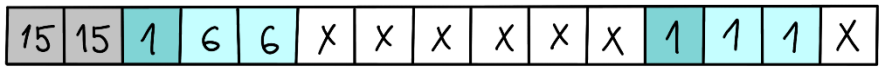
free (prvý aj druhý blok)



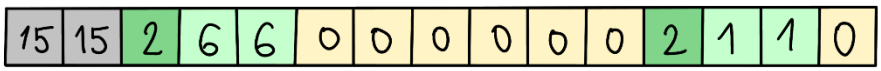
memory_alloc (6)



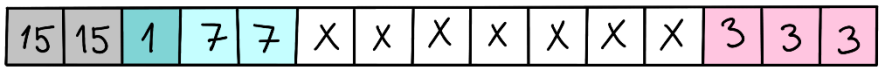
memory_alloc(1)



free (prvý aj druhý blok)



memory_alloc (7)



Scenár 1 - prideľovanie rovnakých blokov malej veľkosti (veľkosti 8 až 24 bytov) pri použití malých celkových blokov pre správcu pamäte (do 50 bytov, do 100 bytov, do 200 bytov)

Bloky	rovnaké, (8 bytov)
Spravovaná pamäť	200 bytov
Náhodné uvoľňovanie	áno
Output: test (1, 1);	Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 2 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 13 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 24 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 35 Pointer je aktívny Pamäť uvolnená úspešne Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (3) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 35 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 46 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 57 Pointer je aktívny Pamäť uvolnená úspešne Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (3) PREBEHLA ÚSPEŠNE,

	alokovaných 8 bitov od pozície 35 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 68 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 79 Pointer je aktívny Pamäť uvolnená úspešne Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (3) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 13 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 90 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 123 Pointer je aktívny Pamäť uvolnená úspešne Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (3) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 57 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 134 Pointer je aktívny ...
Počet uvoľnených blokov	5
Počet alokovaných blokov	18 z 18 (100.00%)
Veľkosť alokovanej pamäte / pokus	198 bytov zo skúšaných 198 bytov (100.00%), v pamäti o velkosti 198

Emma Macháčová
ID: 103037

ÚSPEŠNOSŤ	100%

Bloky	rovnaké, (8 bytov)
Spravovaná pamäť	200 bytov
Náhodné uvoľňovanie	nie
Output: test (1, 0);	Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 2 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 13 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 24 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 35 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 46 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 57 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 68 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 79 Pointer je aktívny Skúšaná veľkosť bloku je 8 -------

	Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 90 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 101 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 112 Pointer je aktívny Skúšaná veľkosť bloku je 8 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 8 bitov od pozície 123 Pointer je aktívny .....
Počet uvoľnených blokov	0
Počet alokovaných blokov	18 z 18 (100.00%)
Veľkosť alokovanej pamäte / pokus	198 bytov zo skúšaných 198 bytov (100.00%), v pamäti o velkosti 198
ÚSPEŠNOSŤ	100%

Bloky	rovnaké, (23 bytov)
Spravovaná pamäť	200 bytov
Náhodné uvoľňovanie	nie
Output: test (1, 0);	Skúšaná veľkosť bloku je 23 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 23 bitov od pozície 2 Pointer je aktívny Skúšaná veľkosť bloku je 23 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 23 bitov od pozície 28 Pointer je aktívny Skúšaná veľkosť bloku je 23 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 23 bitov od pozície 54 Pointer je aktívny Skúšaná veľkosť bloku je 23 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 23 bitov od pozície 80 Pointer je aktívny Skúšaná veľkosť bloku je 23 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 23 bitov od pozície 106 Pointer je aktívny Skúšaná veľkosť bloku je 23 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 23 bitov od pozície 132 Pointer je aktívny ....
Počet uvoľnených blokov	0
Počet alokovaných blokov	7 z 8 (87.50%)
Veľkosť alokovanej pamäte / pokus	182 bytov zo skúšaných 208 bytov (87.50%), v pamäti o velkosti 198
ÚSPEŠNOSŤ	91.92%

Scenár 2 - prideľovanie nerovnakých blokov malej veľkosti (náhodné veľkosti 8 až 24 bytov) pri použití malých celkových blokov pre správcu pamäte (do 50 bytov, do 100 bytov, do 200 bytov),

Bloky	rôzne, (8 až 24 bytov)
Spravovaná pamäť	200 bytov
Náhodné uvoľňovanie	áno
Output: test (2, 1);	Skúšaná veľkosť bloku je 18 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 18 bitov od pozície 2 Pointer je aktívny Skúšaná veľkosť bloku je 17 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 17 bitov od pozície 23 Pointer je aktívny Skúšaná veľkosť bloku je 23 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 23 bitov od pozície 43 Pointer je aktívny Skúšaná veľkosť bloku je 11 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 11 bitov od pozície 69 Pointer je aktívny Pamäť uvolnená úspešne Skúšaná veľkosť bloku je 19 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 19 bitov od pozície 83 Pointer je aktívny Skúšaná veľkosť bloku je 13 ------- Pointer nie je aktívny ALOKÁCIA (4) PREBEHLA ÚSPEŠNE, alokovaných 13 bitov od pozície 2 Pointer je aktívny Skúšaná veľkosť bloku je 11 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 11 bitov od pozície 105 Pointer je aktívny Pamäť uvolnená úspešne Skúšaná veľkosť bloku je 20 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE,

	alokovaných 20 bitov od pozície 119 Pointer je aktívny Skúšaná veľkosť bloku je 19 ------- Pointer nie je aktívny ALOKÁCIA (3) PREBEHLA ÚSPEŠNE, alokovaných 19 bitov od pozície 83 Pointer je aktívny Skúšaná veľkosť bloku je 14 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 14 bitov od pozície 142 Pointer je aktívny Pamäť uvolnená úspešne Skúšaná veľkosť bloku je 9 ------- Pointer nie je aktívny ALOKÁCIA (4) PREBEHLA ÚSPEŠNE, alokovaných 9 bitov od pozície 105 Pointer je aktívny
Počet uvoľnených blokov	3
Počet alokovaných blokov	11 z 11 (100.00%)
Veľkosť alokovanej pamäte / pokus	207 bytov zo skúšaných 207 bytov (100.00%), v pamäti o velkosti 198
ÚSPEŠNOSŤ	100%

Bloky	rôzne, (8 až 24 bytov)
Spravovaná pamäť	200 bytov
Náhodné uvoľňovanie	nie
Output: test (2, 0);	Skúšaná veľkosť bloku je 18 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 18 bitov od pozície 2 Pointer je aktívny Skúšaná veľkosť bloku je 17 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 17 bitov od pozície 23 Pointer je aktívny Skúšaná veľkosť bloku je 23 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 23 bitov od pozície 43 Pointer je aktívny Skúšaná veľkosť bloku je 11 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 11 bitov od pozície 69

	Pointer je aktívny Skúšaná veľkosť bloku je 19 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 19 bitov od pozície 83 Pointer je aktívny Skúšaná veľkosť bloku je 13 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 13 bitov od pozície 105 Pointer je aktívny Skúšaná veľkosť bloku je 11 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 11 bitov od pozície 121 Pointer je aktívny Skúšaná veľkosť bloku je 20 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 20 bitov od pozície 135 Pointer je aktívny Skúšaná veľkosť bloku je 19 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 19 bitov od pozície 158 Pointer je aktívny Skúšaná veľkosť bloku je 14 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 14 bitov od pozície 180 Pointer je aktívny Skúšaná veľkosť bloku je 9 ------- Pointer nie je aktívny CHYBA - NEPODARILO SA ALOKOVAŤ PAMAŤ chýba 9 bitov / 12 Pointer nie je aktívny
Počet uvoľnených blokov	0
Počet alokovaných blokov	10 z 11 (90.91%)
Veľkosť alokovanej pamäte / pokus	195 bytov zo skúšaných 207 bytov (94.20%), v pamäti o velkosti 198
ÚSPEŠNOSŤ	**98.48%**

Scenár 3 - prideľovanie nerovnakých blokov väčšej veľkosti (veľkosti 500 až 5000 bytov) pri použití väčších celkových blokov pre správcu pamäte (aspoň veľkosti 1000 bytov),

Bloky	rôzne, (500 až 5000 bytov)
Spravovaná pamäť	2000 bytov
Náhodné uvoľňovanie	áno
Output: test (3, 1);	Skúšaná veľkosť bloku je 1019 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 1019 bitov od pozície 2 Pointer je aktívny Skúšaná veľkosť bloku je 2651 ------- Pointer nie je aktívny Pamäť nie je dostatočne veľká Pointer nie je aktívny
Počet uvoľnených blokov	0
Počet alokovaných blokov	1 z 2 (50.00%)
Veľkosť alokovanej pamäte / pokus	1022 bytov zo skúšaných 3676 bytov (27.80%), v pamäti o velkosti 1998
ÚSPEŠNOSŤ	51.15%

Bloky	rôzne, (500 až 5000 bytov)
Spravovaná pamäť	7000 bytov
Náhodné uvoľňovanie	áno
Output: test (3, 1);	Skúšaná veľkosť bloku je 1019 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 1019 bitov od pozície 2 Pointer je aktívny Skúšaná veľkosť bloku je 2651 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 2651 bitov od pozície 1024 Pointer je aktívny Skúšaná veľkosť bloku je 2335 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 2335 bitov od pozície 3678 Pointer je aktívny Skúšaná veľkosť bloku je 4948 ------- Pointer nie je aktívny CHYBA - NEPODARILO SA ALOKOVAŤ PAMAŤ chýba 1629 bitov / 4951 Pointer nie je aktívny Pamäť uvolnená úspešne
Počet uvoľnených blokov	1

Počet alokovaných blokov	3 z 4 (75.00%)
Veľkosť alokovanej pamäte / pokus	6014 bytov zo skúšaných 10965 bytov (54.85%), v pamäti o velkosti 6998
ÚSPEŠNOSŤ	85.94%

Bloky	rôzne, (500 až 5000 bytov)
Spravovaná pamäť	15 000 bytov
Náhodné uvoľňovanie	áno
Output: test (3, 1);	Skúšaná veľkosť bloku je 1019 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 1019 bitov od pozície 2 Pointer je aktívny Skúšaná veľkosť bloku je 2651 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 2651 bitov od pozície 1024 Pointer je aktívny Skúšaná veľkosť bloku je 2335 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 2335 bitov od pozície 3678 Pointer je aktívny Skúšaná veľkosť bloku je 4948 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 4948 bitov od pozície 6016 Pointer je aktívny Pamäť uvolnená úspešne Skúšaná veľkosť bloku je 4120 ------- Pointer nie je aktívny CHYBA - NEPODARILO SA ALOKOVAŤ PAMAŤ chýba 90 bitov / 4123 Pointer nie je aktívny
Počet uvoľnených blokov	1
Počet alokovaných blokov	4 z 5 (80.00%)
Veľkosť alokovanej pamäte / pokus	10965 bytov zo skúšaných 15088 bytov (72.67%), v pamäti o velkosti 14998
ÚSPEŠNOSŤ	73.11%

Scenár 4 - prideľovanie nerovnakých blokov malých a veľkých veľkostí (veľkosti od 8 bytov do 50 000) pri použití väčších celkových blokov pre správcu pamäte (aspoň veľkosti 1000 bytov).

Bloky	rôzne, (8 až 50 000 bytov)
Spravovaná pamäť	50 000 bytov
Náhodné uvoľňovanie	áno
Output: test (4, 1);	Skúšaná veľkosť bloku je 42021 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 42021 bitov od pozície 2 Pointer je aktívny Skúšaná veľkosť bloku je 28251 ------- Pointer nie je aktívny CHYBA - NEPODARILO SA ALOKOVAŤ PAMAŤ chýba 20280 bitov / 28254 Pointer nie je aktívny
Počet uvoľnených blokov	0
Počet alokovaných blokov	1 z 2 (50.00%)
Veľkosť alokovanej pamäte / pokus	42024 bytov zo skúšaných 70278 bytov (59.80%), v pamäti o velkosti 49998
ÚSPEŠNOSŤ	84.05%

Bloky	rôzne, (8 až 50 000 bytov)
Spravovaná pamäť	50 000 bytov
Náhodné uvoľňovanie	áno
Output: test (3, 1);	Skúšaná veľkosť bloku je 1019 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 1019 bitov od pozície 2 Pointer je aktívny Skúšaná veľkosť bloku je 2651 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 2651 bitov od pozície 1024 Pointer je aktívny ..... Skúšaná veľkosť bloku je 1224 ------- Pointer nie je aktívny ALOKÁCIA (4) PREBEHLA ÚSPEŠNE, alokovaných 1224 bitov od pozície 15090 Pointer je aktívny Skúšaná veľkosť bloku je 1385 ------- Pointer nie je aktívny ALOKÁCIA (4) PREBEHLA ÚSPEŠNE, alokovaných 1385 bitov od pozície 16183 Pointer je aktívny

	Skúšaná veľkosť bloku je 3399 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 3399 bitov od pozície 36194 Pointer je aktívny Pamäť uvolnená úspešne Skúšaná veľkosť bloku je 2448 ------- Pointer nie je aktívny ALOKÁCIA (4) PREBEHLA ÚSPEŠNE, alokovaných 2448 bitov od pozície 33488 Pointer je aktívny
Počet uvoľnených blokov	6
Počet alokovaných blokov	20 z 20 (100.00%)
Veľkosť alokovanej pamäte / pokus	50780 bytov zo skúšaných 50780 bytov (100.00%), v pamäti o velkosti 49998
ÚSPEŠNOSŤ	100%

Bloky	rôzne, (8 až 50 000 bytov)
Spravovaná pamäť	50 000 bytov
Náhodné uvoľňovanie	nie
Output: test (2, 0);	Skúšaná veľkosť bloku je 18 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 18 bitov od pozície 2 Pointer je aktívny Skúšaná veľkosť bloku je 14 ------- Pointer nie je aktívny ALOKÁCIA (1) PREBEHLA ÚSPEŠNE, alokovaných 14 bitov od pozície 49901 Pointer je aktívny Skúšaná veľkosť bloku je 16 ------- Pointer nie je aktívny CHYBA - NEPODARILO SA ALOKOVAŤ PAMAŤ chýba 6 bitov / 19 Pointer nie je aktívny .....
Počet uvoľnených blokov	0
Počet alokovaných blokov	2636 z 2637 (99.96%)
Veľkosť alokovanej pamäte / pokus	49985 bytov zo skúšaných 50004 bytov (99.96%), v pamäti o velkosti 49998
ÚSPEŠNOSŤ	99.97%

Pamäťová zložitosť je lineárna, lebo množstvo pamäte narastá lineárne s množstvom alokovaných blokov. Časová zložitosť je v najlepšom prípade O(n).