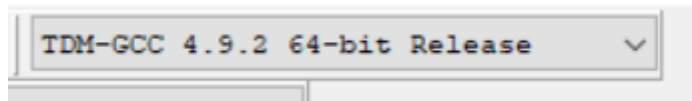


Zadanie 1 – DSA – Tomáš Rafaj

Netuším ako bude aktuálne prebiehať prezentovanie vzhľadom na coronu a zatvorenú školu a preto chcem povedať, že som to celé skúšal a implementoval v programe dev c++



Tento release kompilera a klient dev-c++, pretože by mohli nastať problémy s nejakými vecami v iných kompileroch a pôvodne bolo povedané, že to budeme prezentovať u seba.

Pre implementáciu mojeho zadania som si vybral implicitné organizovanie blokov, v prednáškach označované ako metóda 1.

1. Funkcia init

V projekte sa začína initom, ktorý označí pole znakom, zapíše veľkosť pôvodnej pamäte.

V inite sa taktiež vytvorí prvý voľný blok, už o niečo menší než je zadaná veľkosť poľa, pretože sa odrátajú hlavičky, pätičky a taktiež rozlišovací znak array-u.

Po inite je odhadované, že prebehne malloc, ktorý bude vytvárať a obsádzať bloky ak mu to pamäť dovolí.

Takto vyzerá pole po inite

\$	VELKOST	HLAVICKA							PATICKA
----	---------	----------	--	--	--	--	--	--	---------

Takto vyzerá pole po inite s hodnotou napr. 100

\$	100	87							87
----	-----	----	--	--	--	--	--	--	----

2. Funkcia memory_alloc a firstFit

Malloc preto začne hľadať block, ak bol pointer initovaný. Hľadanie prebieha relatívne jednoducho, pretože sa jedná o implicitné, ide podľa mojich počtov o najhorší prípad (N hľadání) a najlepší prípad (1 hľadanie). Rádovo sa dá povedať, že efektívnosť je približne $n/2$. Preto pri príchode do funkcie memory_alloc sa zavolá funkcia FirstFit, do ktorej sa pošle globálny ukazateľ na pole, ktoré definované v main.

Funkcia prebieha zľava doprava. Vykonáva metódu FirstFit. Prvý pointer ukazuje na začiatok poľa, preto ak je hneď voľné tak to našlo na prvý pokus. Ak však prvý pointer nie je voľný tak sa funkcia rekurzívne zavolá sama na seba a posunie sa na nasledujúci blok, nekontrolujúc, či je voľný alebo prázdny. Funkcia vykoná opätovne rovnaké kroky a takto to robí až pokiaľ nenájde dostatočne voľný blok pre alokáciu pamäte alebo nepríde po konečný rozsah poľa, ktorý si vypočíta ako vzdialenosť od pôvodného pointeru + veľkosť pôvodná uložená o 1 bajt ďalej.

Čiže v prípade, že funkcia vráti platný pointer malloc sa vykoná analogicky na základe algoritmu. Ak sa jej nepodarí nájsť vhodný blok, vráti NULL.

V prípade, že prišiel platný pointer, vyhodnotia sa viaceré prípady ako napr. či blok nie je presne rovnako veľký ako je požadovaný malloc, čo by ušetrilo viacero aritmetiky a mohol by sa rovno zapísať. Taktiež sa kontroluje prípad, že ak je blok veľký tak, že po malloc by už nemal priestor na hlavičku a pätičku tak sa zoberie rovno celý, mallocne sa teda o niečo viac pamäte.

V prípade, že sa vykonalo všetko úspešne tak mallocne returne pointer ukazujúci na začiatok voľného bloku, resp. na jeho hlavičku. A v prípade, že nie tak vráti NULL pointer.

3. Funkcia memory_free

Funkcia free zistí, či je prebehol init, tým, že neni pointer NULL a pozrie, či sme mu poslali platný pointer a či náhodou nie je tiež NULL. Ak ani jedno kontroluje, či sa jedná o obsadený block.

Potom sa pozerá ako sú na tom vedľajšie blocky. To jest ten napravo a naľavo. Následne na základné možnosti obsadenia susedných blockov sa vykoná spájanie a freeovanie.

3.1 Funkcja merge_ptr

Táto funkcia slúži na spájanie blockov pri funkcii `memory_free`. Prvý prípad je spájanie aktuálneho blocku a spájanie blocku napravo. Druhý prípad je analogicky rovnaký, ide len o to, že voľný blok je naľavo, a preto sa funkcia len posunie a zavolá sa rekurzívne ako keby to bolo spájanie napravo. Špeciálny prípad je spájanie oboch strán naraz. Preto sa najprv vykoná spájanie rekurzívne doprava a potom posunutie a zasa rekurzívne doprava. Ošetril som aj prípady ľavého okrajového pointeru, ktorý niekedy zobral pamäť z ľavej strany a spôsobovalo to problémy. Bola tam uložená veľkosť a char znak na definovanie začiatku poľa.

4. Funkcia memory_check

Zo zadania mi nebolo úplne jasné, čo je cieľom `memory_check` funkcie, a preto som si to interpretoval nasledovne

Funkcia zavolá funkciu, ktorá prehľadá všetky bloky vrámci array-u a hľadá daný pointer v mallocu. Ostatné vykoná ako je napísané v zadaní.

Výzor volného a plného bloku.

[illegible]

Šírka header a footer je int (4 na 64bitovom).

Dáta obsahujú hodnotu zapísanú vo footer a header.

Ukázkový příklad

Volný blok

[illegible]

Plný blok

-8									-8
----	--	--	--	--	--	--	--	--	----

Výzor array-u

\$	SIZE	HEADER							FOOTER
----	------	--------	--	--	--	--	--	--	--------

Takže ako som už písal v inite, na prvom bajte je uložený char na definovanie mojego array-u, čo využívam v `memory_check`. Potom je pôvodná nezmená veľkosť array-u využívaná na určovanie hraníc array-u, pretože pointer ukazuje iba na začiatok a nemá koniec.

Potom už nasledujú voľné / plné blocky a ich hlavičky a pätičky.

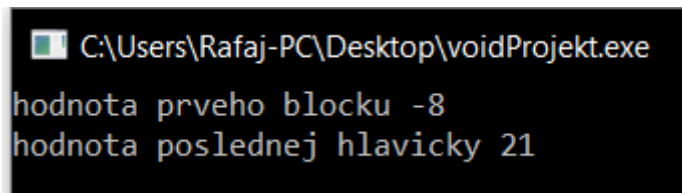
Testovania

1. Pridelovanie 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),
2. Pridelovanie 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),
3. Pridelovanie nerovnakých blokov väčšej veľkosti (veľkosti 500 až 5000 bytov) pri použití väčších celkových blokov pre správcu pamäte (aspoň veľkosti 1000 bytov),
4. Pridelovanie 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).

1. Rovnaké bloky (8-24) do blokov (50-200)

Prípad 1 – pridelovanie `memory_alloc(8)` do `region[50]`

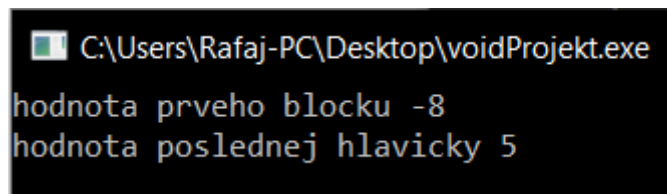
- 1x `memory_alloc(8)` a `region[50]`
 - Hodnota voľného blocku ostala 21
 - Úspešna `memory_alokacia`



```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
hodnota prveho blocku -8
hodnota poslednej hlavicky 21
```

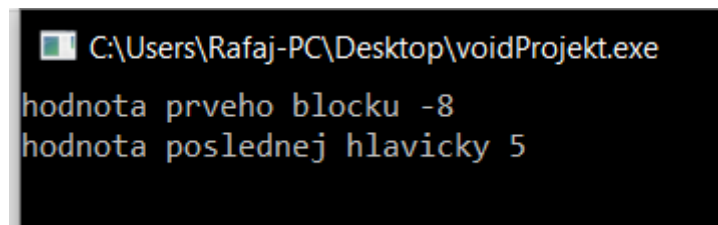
- 2x `memory_alloc(8)` a `region[50]`
 - Hodnota voľného blocku ostala 5

- Uspesna memory_alokacia



```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
hodnota prveho blocku -8
hodnota poslednej hlavicky 5
```

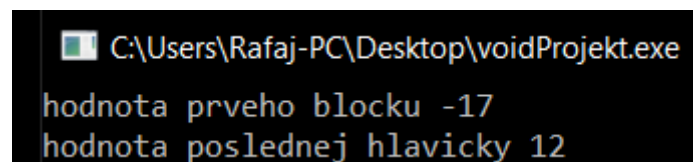
- 3x memory_alloc(8) a region[50]
 - Hodnota voľného blocku ostala 5
 - 2 úspešná memory_alokácia a 1 neúspešná, pretože už nebolo dostatočne veľa voľnej pamäte



```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
hodnota prveho blocku -8
hodnota poslednej hlavicky 5
```

Prípad 2 – pridelenie memory_alloc(17) do region[50]

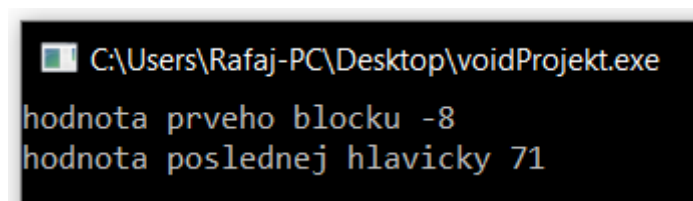
- 2x memory_alloc(17) do region[50]
 - Hodnota voľného blocku ostala 12
 - 1 úspešná memory_alokácia a 1 neúspešná, pretože už nebolo dostatočne veľa voľnej pamäte, t.j. bolo 12 a potrebovalo 17



```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
hodnota prveho blocku -17
hodnota poslednej hlavicky 12
```

Prípad 3 – pridelenie memory_alloc(8) do region[100]

- 1x memory_alloc(8) a region[100]
 - Hodnota voľného blocku ostala 71
 - 1 úspešná memory_alokácia



```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
hodnota prveho blocku -8
hodnota poslednej hlavicky 71
```

- 7x memory_alloc(8) a region[100]
 - Hodnota voľného blocku ostala 7
 - 5x úspešná alokácia veľkosti 8, 2x sa už nezmestila

```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
hodnota prveho blocku -8
hodnota poslednej hlavicky 7
```

- Grafická ukážka (D je ako DATA)

\$	100	-8	D	-8	-8	D	-8	-8	D	-8	-8	D	-8	-8	D	-8	7		7
----	-----	----	---	----	----	---	----	----	---	----	----	---	----	----	---	----	---	--	---

Prípad 4 – Pridelovanie memory_alloc(24) do region[200]

- 1x memory_alloc(24) do region[200]
 - Hodnota voľného blocku ostala 155
 - 1x úspešná alokácia

```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
hodnota prveho blocku -24
hodnota poslednej hlavicky 155
```

2. Rôzne bloky (8-24) do blokov (50-200)

Prípad 1 – 8,7,23 do region[50]

- Memory_allocy(poradie ako napísané) : 8, 7, 23
- 2x úspešná alokácia veľkostí 8, 7
- 1x neúspešná alokácia veľkosti 23 z dôvodu nedostatku voľnej pamäte

```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
hodnota prveho blocku -8
hodnota poslednej hlavicky 6
```

Prípad 2- 22,9,13 do region[100]

- Memory_allocy : 22, 9, 13
- 19B ostalo voľných

```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
Pocet uspesnych alokacii :3
Hodnota prveho blocku -22
Hodnota poslednej hlavicky 19
```

Prípad 3 – 14,19,11,8 do region[200]

- Memory_allocy: 14, 19, 11, 8
- 103B ostalo voľných

```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
Pocet uspesnych alokacii :4
Hodnota prveho blocku -14
Hodnota poslednej hlavicky 103
```

Prípad 4 – 18,20,9,23 do region[200]

- Memory_allocy: 18,20,9,23
- 85B ostalo voľných

```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
Pocet uspesnych alokacii :4
Hodnota prveho blocku -18
Hodnota poslednej hlavicky 85
```

Prípad 5 – 9,13,10,25 do region[200]

- Memory_allocy: 9,13,10,25
- 98B ostalo voľných

```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
Pocet uspesnych alokacii :4
Hodnota prveho blocku -9
Hodnota poslednej hlavicky 98
```

3. Rôzne bloky (500-5000) do blokov (1000+)

Prípad 1 – 638 do region[2000]

- Memory_allocy: 638
- 1341B ostalo voľných

```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
Pocet uspesnych alokacii :1
Hodnota prveho blocku -638
Hodnota poslednej hlavicky 1341
```

Prípad 2 – 2524,923,1000 do region[1000]

- Memory_allocy: 2524, 923, 1000
- 56B ostalo voľných

- Prvý a tretí malloc vrátil NULL, pretože nemal dostatok voľnej pamäte.

```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
Pocet uspesnych alokacii :1
Hodnota prveho blocku -923
Hodnota poslednej hlavicky 56
```

Prípad 3 - 659, 1294 do region[9000]

- Memory_allocy: 659, 1294
- 7018B ostalo voľných

```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
Pocet uspesnych alokacii :2
Hodnota prveho blocku -659
Hodnota poslednej hlavicky 7018
```

Prípad 4- 1319, 614 do region[2000]

- Memory_allocy 1319, 614
- 38B ostalo voľných

```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
Pocet uspesnych alokacii :2
Hodnota prveho blocku -1319
Hodnota poslednej hlavicky 38
```

4. Rôzne bloky(8-50000) do blokov (1000+)

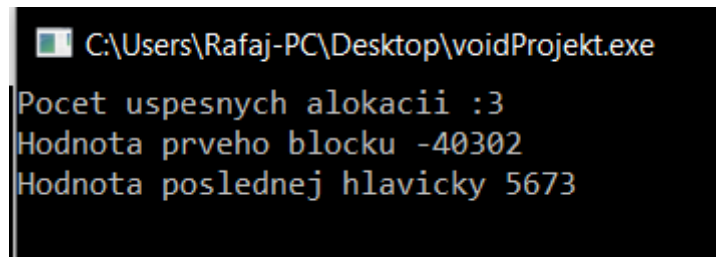
Prípad 1 – 8,4939,47398 do region[7800]

- 2824B ostalo voľných
- 2 úspešné mallocoy, tretí očividne neprešiel pre nedostatok pamäte

```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
Pocet uspesnych alokacii :2
Hodnota prveho blocku -8
Hodnota poslednej hlavicky 2824
```

Prípad 2 – 40302,39,3949 do region[50000]

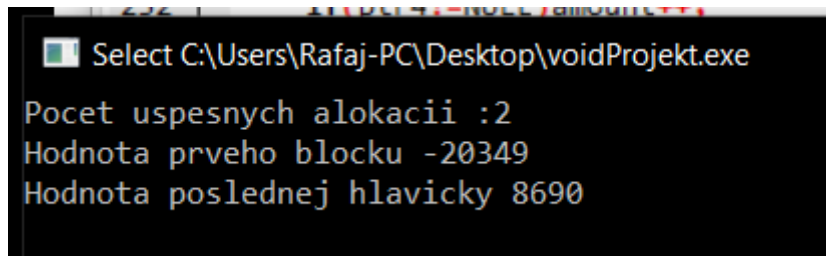
- Memory_allocy : 40302, 39, 3949
- 5673B ostalo voľných



```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
Pocet uspesnych alokacii :3
Hodnota prveho blocku -40302
Hodnota poslednej hlavicky 5673
```

Prípad 3 – 20349,932,9394 do region[30000]

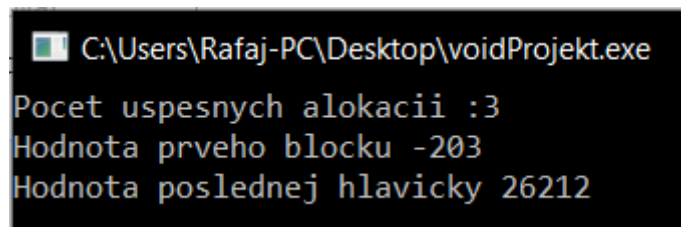
- Memory_allocy: 20349, 932, 9394
- 8690B ostalo voľných
- Posledný malloc neúspešný z dôvodu nedostatku voľnej pamäte



```
Select C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
Pocet uspesnych alokacii :2
Hodnota prveho blocku -20349
Hodnota poslednej hlavicky 8690
```

Prípad 4- 203,3049,10499 do region[40000]

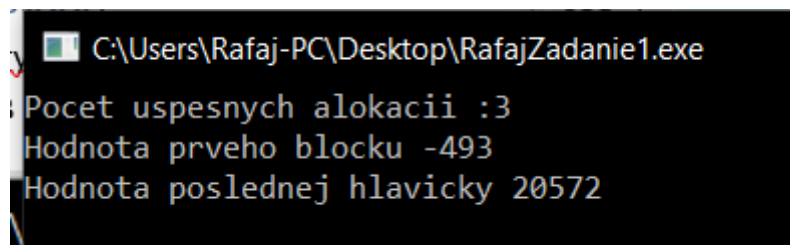
- Memory_allocy: 203,3049,10499
- 26212B ostalo voľných



```
C:\Users\Rafaj-PC\Desktop\voidProjekt.exe
Pocet uspesnych alokacii :3
Hodnota prveho blocku -203
Hodnota poslednej hlavicky 26212
```

Prípad 5- 493,4949,3949 do region[30000]

- Memory_allocy: 493,4949,3949
- 20572B ostalo voľných

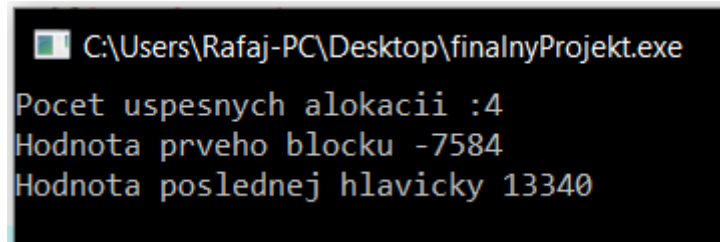


```
C:\Users\Rafaj-PC\Desktop\RafajZadanie1.exe
Pocet uspesnych alokacii :3
Hodnota prveho blocku -493
Hodnota poslednej hlavicky 20572
```


Prípad 6- 7584,9339,299,19393

do region[50000]

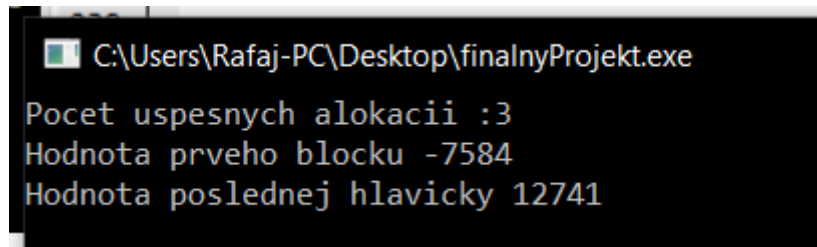
- Memory_allocy: 7584,9339,299,19393
- 13340B ostalo voľných



```
C:\Users\Rafaj-PC\Desktop\finalnyProjekt.exe
Pocet uspesnych alokacii :4
Hodnota prveho blocku -7584
Hodnota poslednej hlavicky 13340
```

do region[30000]

- Memory_allocy: 7584,9339,299,19393
- Posledny malloc nebol úspešný, nedostatok pamäte.
- 12741B ostalo voľných



```
C:\Users\Rafaj-PC\Desktop\finalnyProjekt.exe
Pocet uspesnych alokacii :3
Hodnota prveho blocku -7584
Hodnota poslednej hlavicky 12741
```

Prikladám taktiež žiadané obrázky podľa zadania :

Ukazujem aspoň nejaké vyššie zbehnuté testy.

Obrázok obsahuje náčrt obsahu array-u.

Na začiatku vždy na dĺžke 1B ako vidieť nad tým napísané, je uložený definujúci char.

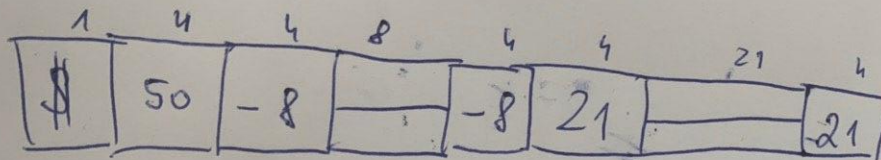
Potom nasleduje uložená pôvodná veľkosť uložená na 4 bajtoch.

Následne nasledujú bloky, aj voľné aj plné. Voľné majú kladné znamienko a plné záporné. Náčrty som

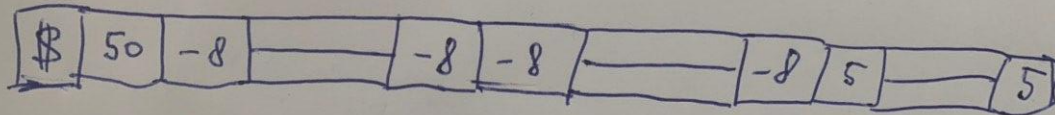
využil ku kontrole výstupov programu.

mem_malloc(8) region(50)

1x 8B

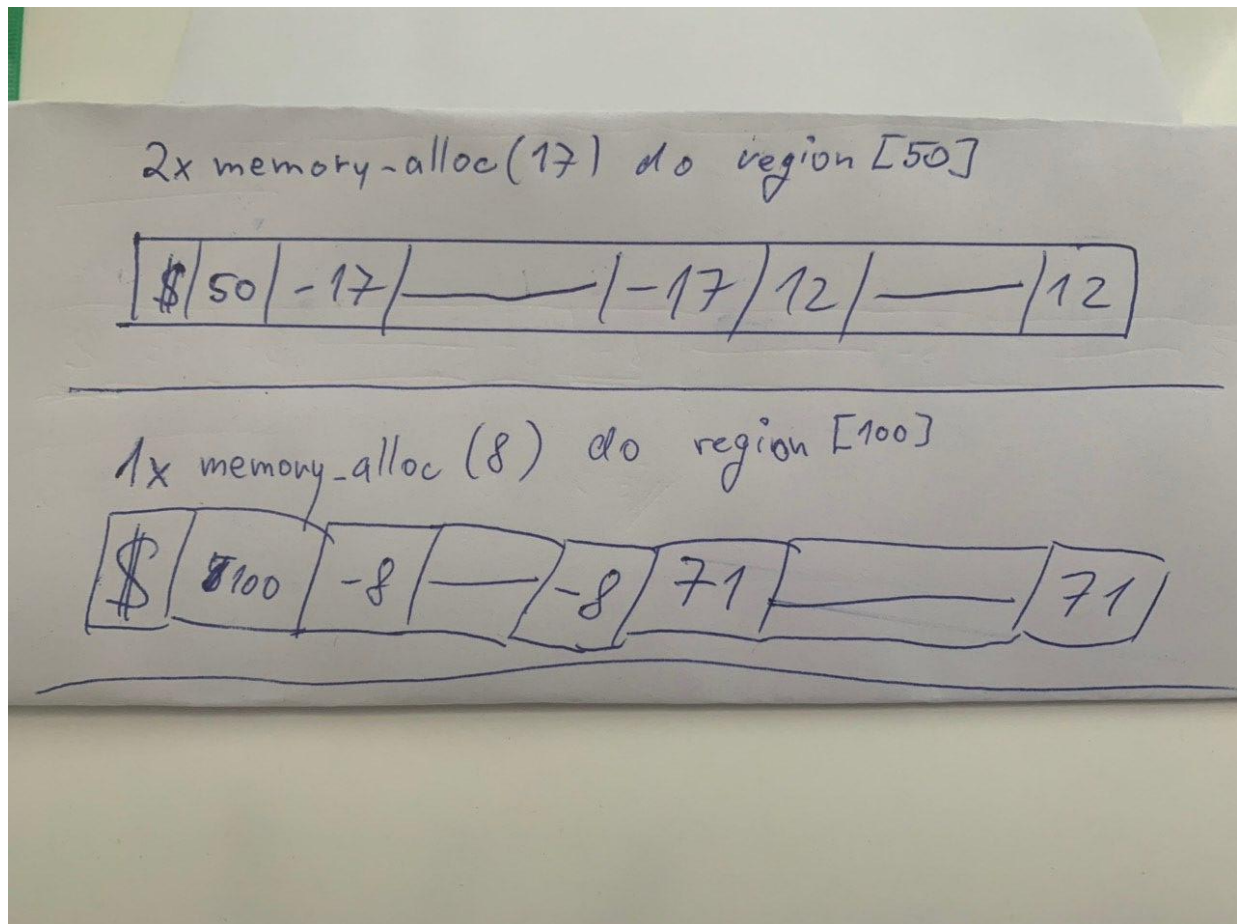


2x 8B



3x 8B

TO 1STE //



Testovanie funkcie memory_free()

V testovaní nebolo napísané, že máme skúšať uvoľňovanie pamäte ale iba jej pridelovanie ale pre istotu sme pridám aj pár prípadov freeovania.

Test 1

```
int main(int argc, char *argv[]) {  
    char region[100];  
    memory_init(region, 100);  
    int amount = 0;  
  
    void *ptr1 = NULL;  
  
    ptr1 = memory_alloc(20);  
    memory_free(ptr1);  
  
    printf("Hodnota prveho blocku %d\n", *(int*)(pointer));  
    printf("Hodnota poslednej hlavicky %d\n", *(int*)(pointer+100-4-5));  
  
    return 0;  
}
```

```
C:\Users\Rafaj-PC\Desktop\voidProjektFinal.exe  
Hodnota prveho blocku 87  
Hodnota poslednej hlavicky 87
```

Test 2

Oba mallocy prebehli úspešne.

1. malloc vyhradil 40B

2. malloc vyhradil 39B (zaokrúhlil to) lebo 5B by už bolo nedostatočné na nový blok

free potom uľovnil 40B pamäte

```
int main(int argc, char *argv[]) {
    char region[100];
    memory_init(region,100);
    int amount = 0;

    void *ptr1 = NULL;
    void *ptr2 = NULL;
    ptr1 = memory_alloc(40);
    ptr2 = memory_alloc(34);
    memory_free(ptr1);

    printf("Hodnota prveho blocku %d\n",*(int*)(pointer));
    printf("Hodnota poslednej hlavicky %d\n",*(int*)(pointer+100-4-5));

    return 0;
}
```

```
C:\Users\Rafaj-PC\Desktop\voidProjektFinal.exe
Hodnota prveho blocku 40
Hodnota poslednej hlavicky -39

-----
Process exited after 0.05907 seconds with return value 0
Press any key to continue . . .
```

Test 3

Oba mallocy prebehli úspešne

Free oboch prebehlo úspešne a array je v pôvodnom stave po inite.

```
int main(int argc, char *argv[]) {
    char region[30000];
    memory_init(region,30000);
    int amount = 0;

    void *ptr1 = NULL;
    void *ptr2 = NULL;
    ptr1 = memory_alloc(7000);
    ptr2 = memory_alloc(8000);
    memory_free(ptr1);
    memory_free(ptr2);

    printf("Hodnota prveho blocku %d\n",*(int*)(pointer));
    printf("Hodnota poslednej hlavicky %d\n",*(int*)(pointer+30000-4-5));

    return 0;
}
```

```
Select C:\Users\Rafaj-PC\Desktop\voidProjektFinal.exe
Hodnota prveho blocku 29987
Hodnota poslednej hlavicky 29987

-----
Process exited after 0.05256 seconds with return value 0
Press any key to continue . . .
```