

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**  
**FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLOGIÍ**

**Zadanie 1**  
**SPRÁVCA PAMÄTI**

Anna Yuová

**Predmet:** Dátové štruktúry a algoritmy  
**Akademický rok:** 2020/2021  
**Semester:** letný

## Správca pamäti

Pre vypracovanie môjho zadania som si vybrala metódu č.1 - spracovanie pomocou implicitného zoznamu. Na implementáciu som použila program DEV C++ (verzia 5.11) a kód je spustiteľný.

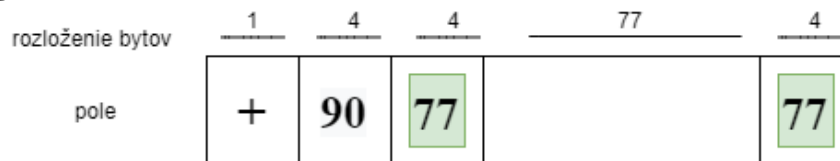
## Opis môjho algoritmu

Na začiatku pri vytváraní pola som si vyhradila 1\*char a 3\*int čiže stratím 13B aj na hlavičku aj pätičku a aj si držím veľkosť pola aj to, kde moje pole začína. V programe mám jednu globálnu premennú (void \*smerník) a je to smerník na začiatok prvej hlavičky celého pola. Tento smerník si v programe pretypujem na dátový typ, ktorý potrebujem.

### Memory\_init()

- vytvorenie a pripravenie pola so zadanou veľkosťou
- nastavenie do globálnej premennej (smerník) adresu na hlavičku – začiatok na tú časť pola, s ktorou idem pracovať
- na prvom mieste je znak (char), aby som vedela, kde to pole začína
- za znakom je veľkosť pola (int)
- za veľkosťou je hlavička(int) a pätička (int)
- v hlavičke a pätičke je na začiatku hodnota, ktorá sa dá alokovať

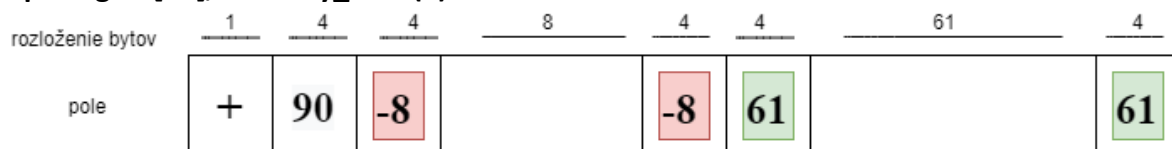
Napr. pre region[90]



### Memory\_alloc

- zavolanie funkcie najdiVolnyBlok() – prechádza celé pole – znak implicitného zoznamu
- voľné bloky hľadám metódou FirstFit – prvý vhodný voľný blok
- nastavím veľkosť a adresu hlavičky a pätičky
- alokovaná veľkosť hlavičky sa nastaví na zápornú hodnotu požadovanej veľkosti
- úspešná alokácia vráti smerník na úspešne alokovaný kus pamäte
- ak nie je dostatok miesta na alokáciu, funkcia vráti NULL

Napr. region[90], memory\_alloc(8)



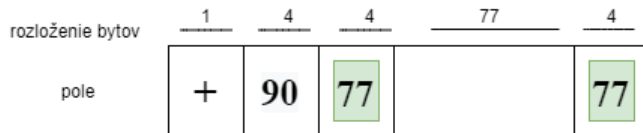
### Memory\_free

- nastavenie do premennej aktualnySmernik pointer na adresu, ktorú chcem uvoľniť (z argumentu funkcie)
- uloženie hodnoty na danej adrese, ktorú chcem uvoľniť
- kontrola či je tá hodnota < 0, čiže bola alokovaná a dá sa uvoľniť
- ak je splnená podmienka, zmení sa hodnota v hlavičke aj pätičke na absolútnu hodnotu
- potrebná kontrola či naokolo nie sú voľné bloky, ktoré by sa tiež dali spojiť od jedného veľkého voľného bloku
- kontrola pätičky pred hlavičkou a za pätičkou, ak sú kladné spojím ich

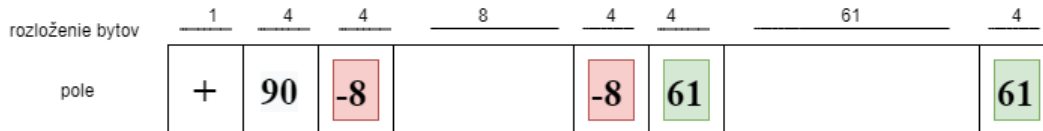
-vypočítam veľkosť spojenej časti, prepíšem hlavičku a pätičku a starú pätičku a hlavičku vynulujem

**Napr. region[90]; int \*ptr1 = memory\_alloc(8); int \*ptr2 = memory\_alloc(6); memory\_free(ptr1)**

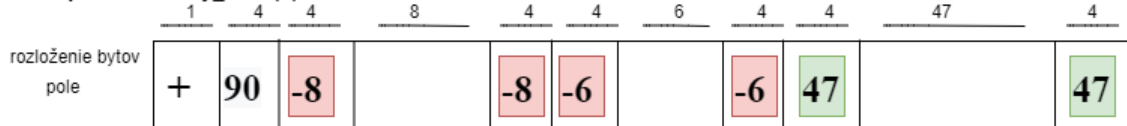
1. region[90]



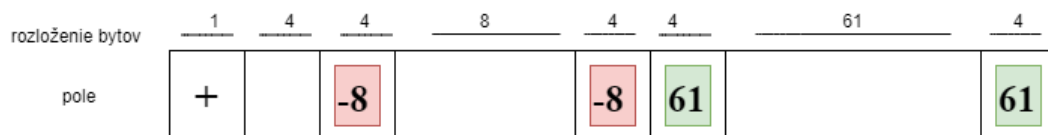
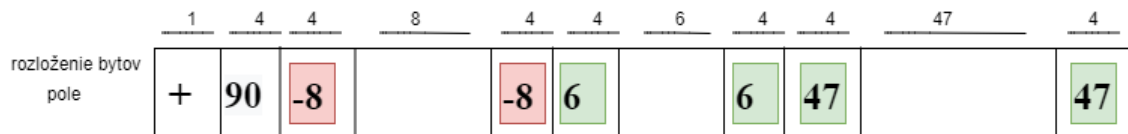
2. int \*ptr1 = memory\_alloc(8)



3. int \*ptr1 = memory\_alloc(6)

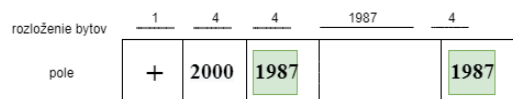


4. memory\_free(ptr1)



**Napr. region[2000]; int \*ptr1 = memory\_alloc(650); int \*ptr2 = memory\_alloc(1000); memory\_free(ptr2); int \*ptr3 = memory\_alloc(300); memory\_free(ptr1); int \*ptr4 = memory\_alloc(250)**

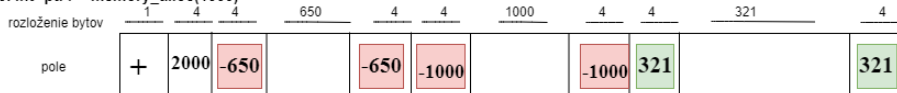
#### 1. region[2000]



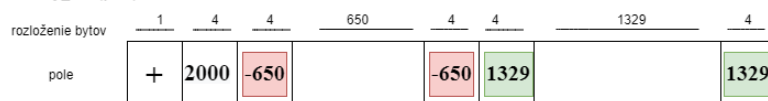
#### 2. int \*ptr1 = memory\_alloc(650)



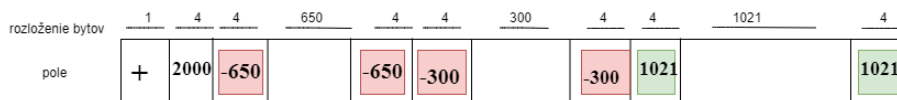
#### 3. int \*ptr1 = memory\_alloc(1000)



#### 4. memory\_free(ptr2)



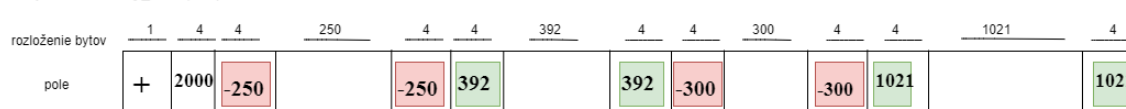
#### 5. int \*ptr3 = memory\_alloc(300)



#### 6. memory\_free(ptr1)



#### 7. int \*ptr4 = memory\_alloc(250)



Vo funkcii **memory\_check** skontrolujem či daný pointer v argumente je v rozsahu môjho poľa a či má zápornú hodnotu, teda či bol alokovaný. Ak to spĺňa tieto podmienky, funkcia vráti 1, inak 0.

## Testovanie

### Vysvetlenie k výpisu:

**Červená** = ukáže adresu prvej hlavičky v poli (smerníka) v memory\_init a tá sa zhoduje s adresou prvej hlavičky pri mojich testoch a aj v testovacej funkcii

**Modrá** = ukáže adresu poslednej pätičky v memory\_init a tá sa zhoduje s adresou poslednej pätičky pri mojich testoch a aj v testovacej funkcii

```
C:\Users\Anna\Desktop\Zadanie 1\devv\Projekt1.exe
adresa smernika v init: 6487477   adresa paticka v init: 6487558
hodnota smernika: -5
adresa smernika: 6487477

hodnota paticky: 64
adresa paticky: 6487558
Adresy a hodnoty hlavicky a paticky sa zhoduju

-----MOJE TESTY NA OVERENIE-----
Takto vyzeru konecne alokovane pole

+ 90 -5 -5 64 64

-----KONIEC MOJICH TESTOV NA OVERENIE-----

-----TESTOVACIA FUNKCIA-----
Dostal hodnotu: -5 a ockaval hodnotu: -5
Rovnaju sa!

Dostal hodnotu: -5 a ockaval hodnotu: -5
Rovnaju sa!

Dostal hodnotu: 64 a ockaval hodnotu: 64
Rovnaju sa!

Dostal hodnotu: 64 a ockaval hodnotu: 64
Rovnaju sa!

-----
Process exited after 0.06892 seconds with return value 0
Press any key to continue . . .
```

1. Testovala som rovnaké bloky malej veľkosti (8-24 bytov) pri použití celkových malých blokov pre sprácu pamäte(50-200 bytov)

```
C:\Users\Anna\Desktop\Zadanie 1\devv\Projekt1.exe
adresa smernika v init: 6487365   adresa paticka v init: 6487556
hodnota smernika: -22
adresa smernika: 6487365

hodnota paticky: 97
adresa paticky: 6487556
Adresy a hodnoty hlavicky a paticky sa zhoduju

-----MOJE TESTY NA OVERENIE-----
Takto vyzeru konecne alokovane pole

+ 200 -22 -22 -22 -22 -22 -22 97 97
```

2. Testovala som nerovnaké bloky malej veľkosti (8-24 bytov) pri použití celkových malých blokov pre sprácu pamäte(50-200 bytov)

C:\Users\Anna\Desktop\devv\Projekt1.exe

```
adresa smernika v init: 6487509   adresa paticka v init: 6487550
hodnota smernika: -8
adresa smernika: 6487509

hodnota paticky: 3
adresa paticky: 6487550

+ 50 -8 -8 -10 -10 3 3
```

3. Testovala som nerovnaké bloky väčšej veľkosti (500-5000bytov) pri použití väčších celkových blokov pre správcu pamäte (aspoň 1 000bytov)

C:\Users\Anna\Desktop\devv\Projekt1.exe

```
adresa smernika v init: 6486581   adresa paticka v init: 6487572
hodnota smernika: -600
adresa smernika: 6486581

hodnota paticky: 379
adresa paticky: 6487572

+ 1000 -600 -600 379 379
-----
```

4. Testovala som nerovnaké bloky väčšej aj menšej veľkosti (8- 50 000bytov) pri použití väčších celkových blokov pre správcu pamäte (aspoň 1000bytov)

C:\Users\Anna\Desktop\devv\Projekt1.exe

```
adresa smernika v init: 6442565   adresa paticka v init: 6487556
hodnota smernika: -28000
adresa smernika: 6442565

hodnota paticky: 63
adresa paticky: 6487556

+ 45000 -28000 -28000 -15000 -15000 -1900 -1900 63 63
```

## Priestorová a časová zložitosť

### Priestorová zložitosť

- Na začiatku pri vytváraní pola som si vyhradila 1\*char a 3\*int čiže stratím 13B aj na hlavičku aj pätičku a aj si držím veľkosť pola aj to, kde moje pole začína -> STRATA 13B
- Priestorová zložitosť je lineárna =  $O(n)$

### Časová zložitosť

- memory\_init má časovú zložitosť  $O(1)$
- memory\_alloc má časovú zložitosť v najhoršom prípade  $O(n)$
- memory\_check má časovú zložitosť  $O(1)$ , iba overuje či je smerník vo vyhradenej pamäti a či ešte nebol uvoľnený, teda jeho hodnota je záporná
- memory\_free má časovú zložitosť v najhoršom prípade  $O(n)$