

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

**Zadanie 2**

**VYHLADÁVANIE V DYNAMICKÝCH MNOŽINÁCH**

Anna Yuová

**Predmet:** Dátové štruktúry a algoritmy

**Akademický rok:** 2020/2021

**Semester:** letný

## 1. ÚVOD

V mojom zadaní robím avl strom, prevzala som splay strom, hash som robila pomocou zreteženia a prevzala som hash s otvoreným adresovaním. Implementácia je v programe DEV-C++ (verzia 5.11) a kódy sú spustiteľné.

## 2. MÔJ AVL STROM

V tomto samo-vyvažovacom binárnom vyhládávacom strome je dôležité, že rozdiel výšok pravého a ľavého podstromu nemôže byť väčší ako 1. Na začiatku si vytvorím štruktúru, v ktorej mám kľúč (daná hodnota), výšku a viem sa pohybovať doľava a doprava.

**INSERT** - Vkladám prvky vo funkciu init\_tree(), ak ešte predtým strom neboli vytvorený. Ak už som ho vytvárala, musím hľadať tým, že sa pozerám doľava a doprava a posúvam sa po týchto uzloch. Potom rekurzívne vkladám do stromu ako pri klasickom binárnom vyhládávacom strome. Ak je dané číslo menšie ako koreň, vložím ho na ľavú stranu a ak je väčšie ako koreň vloží ho na pravú stranu. Po tomto vložení sa viem vrátiť vždy zdola nahor na vyššie úrovne. Pri vkladaní si postupne aj počítam aktuálnu výšku, aby som sa vedela dopracovať k výške pravého a ľavého podstromu. Podľa týchto výšok si vypočítam faktor vyváženia = výška ľavého podstromu - výška pravého podstromu. Ak je faktor vyváženia väčší ako 1 (alebo menší ako -1), tak strom treba vyvážiť lebo je nevyvážený. Môžu nastat 4 prípady pri rotácii – doľava → doľava doľava alebo doľava doprava alebo doprava → doprava doprava alebo doprava doľava. Ak je faktor väčší ako 1, treba upraviť ľavú stranu a teda rotujem doprava. Skontrolujem teda či je vľavo tak, že porovnám aktuálny kľúč s kľúčom v ľavom koreni podstromu. Ak je faktor menší ako -1, treba upraviť pravú stranu a rotujem doľava. Skontrolujem teda či je vpravo tak, že porovnám aktuálny kľúč s kľúčom v pravom koreni podstromu.

**Pri rotovaní doprava** sa mi vymení ľavé dieťa s koreňom, koreň sa dá na pravú stranu a ľavé dieťa koreňa sa presunie do koreňa. Musím si preto zapamätať pravé dieťa ľavého dieťaťa (to pravé dieťa je väčšie ako ľavé dieťa, preto je napravo), pretože po presunutí bude ako menšie (ľavé) dieťa pôvodného koreňa. Tak isto funguje aj rotovanie doľava.

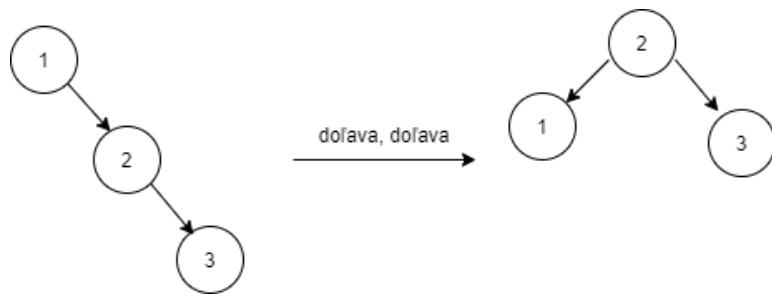
**Ak rotujem najskôr doľava a potom doprava**, dostanem pravé dieťa ľavého dieťaťa do koreňa, a preto si musím do pomocnej uložiť jeho deti. Toto pravé dieťa posuniem miesto pôvodného ľavého dieťaťa a ľavé dieťa mu ostane a ako pravé dieťa sa mu nastaví pôvodné ľavé dieťa a tomuto ľavému dieťaťu sa nastaví ako pravé dieťa, dieťa pôvodného ľavého dieťaťa. Potom nové ľavé dieťa chcem dostať do koreňa a postupujem klasicky ako pri rotácii doprava.

**SEARCH** - Hľadanie prvkmu som robila tak, že zadanú hodnotu porovnáva postupne s hodnotami ako prechádza strom rekurzívne doľava/doprava (podľa toho či je hľadaný prvek väčší alebo menší ako koreň) ak nájde hodnotu v strome, vráti tú hodnotu a vypíše, že sa tam nachádza. Ak nie, vráti null a vypíše, že sa nenachádza.

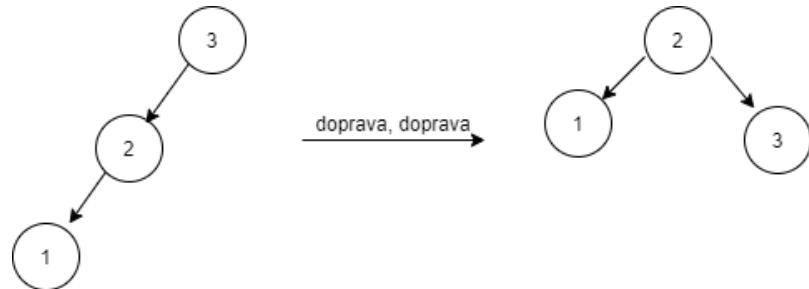
Výpis som robila **INORDER**, najprv sa rekurzívne posúvam na najviac ľavé dieťa, vypísem hodnotu, prejdem na rodiča a vypísem a prejdem na ľavé dieťa a vypísem, ak už nie je ďalšie iné dieťa, potom vraciám sa späť o úroveň vyššie a prejdem na pravé dieťa. Takže tie čísla vypísem akoby od najmenšieho po najväčšie.

### 2.1. MÔJ AVL STROM - PRÍKLAD

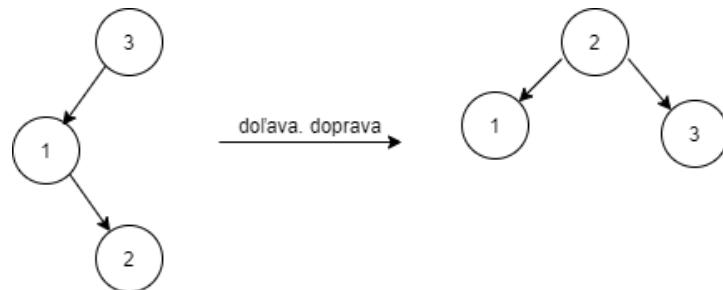
Rotácia doľava, doľava – napr. ak vkladám 1, 2, 3



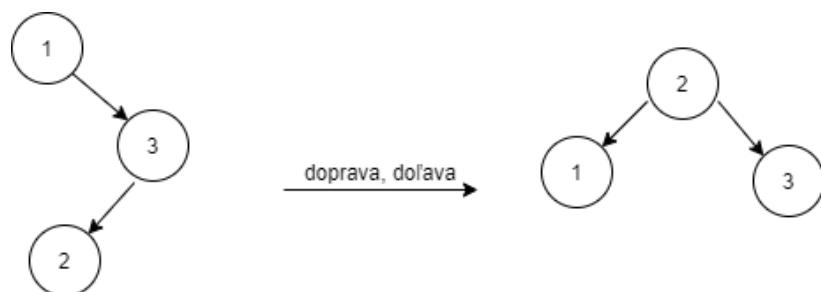
**Rotácia doprava, doprava – napr. ak vkladám 3, 2, 1**



**Rotácia dol'ava, doprava – napr. ak vkladám 3, 1, 2**



**Rotácia doprava, dol'ava – napr. ak vkladám 1, 3, 2**



Časová zložitosť funkcií insert a search majú zložitosť  $O(\log n)$ .

### 3. PREVZATÝ SPLAY STROM

Kód je prevzatý zo stránky:

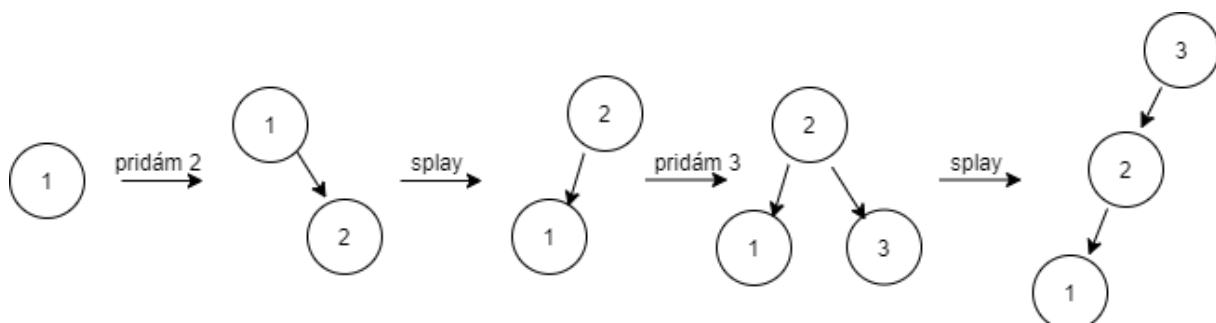
Amit Kumar, Developer, Founder of [www.codesdope.com](http://www.codesdope.com)

<https://www.codesdope.com/course/data-structures-splay-trees/>

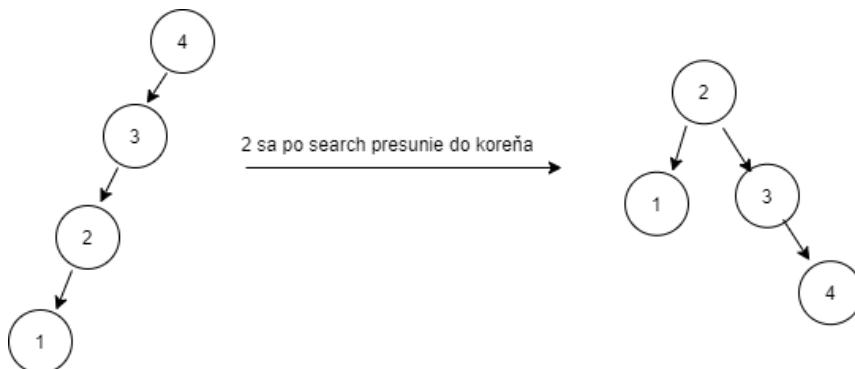
Splay strom je tiež samovyvažujúci binárny vyhľadávací strom, ktorý má vlastnosť, že sa viem čo najrýchlejšie dostať k tým prvkom, ktoré sme pridali naposledy, lebo každý pridaný prvak sa dá do koreňa. Splay funkcia zabezpečí to, že usporiada ten strom tak, aby sa daný uzol dostał do koreňa. Vyhľadá daný uzol v strome a potom vykoná rotácie tak, aby sa dostal do koreňa. Vďaka tomu bude daný prvak prístupný rýchlejšie ( $O(1)$ ), pretože klasické vkladanie a vyhľadávanie trvajú  $O(\log n)$ . Insert vkladá klasicky prvky, ako pri inom strome, akurát vďaka splay funkcie posunie ten vkladaný prvak vždy do koreňa. **SEARCH** hľadá prvky ako klasický vyhľadávací binárny strom akurát vráti nový koreň, pretože ak tam ten kľúč je tak sa prenesie do koreňa. Výpis je **INORDER**, najprv sa rekurzívne posúva na najviac ľavé dieťa, vypíše hodnotu, prejde na rodiča a vypíše a prejde na ľavé dieťa a vypíše, ak už nie je ďalšie iné dieťa, potom sa vracia späť o úroveň vyššie a prejde na pravé dieťa. Takže tie čísla vypíše akoby od najmenšieho po najväčšie.

### 3.1. PREVZATÝ SPLAY STROM – PRÍKLAD

Vkladanie 1, 2, 3, 4 -> najprv sa vloží 1, 2 sa vloží ako pravé dieťa, prebehne splay a vymenia sa do koreňa, 1 je teraz menšia ako 2, tak bude ľavé dieťa a vypíše, ak už nie je ďalšie iné dieťa, potom sa vracia späť o úroveň vyššie a prejde na pravé dieťa. Takže tie čísla vypíše akoby od najmenšieho po najväčšie.



**search(2):**



### 4. MÔJ HASH SO ZREŤAZENÍM

Na môj hash som si vybrała hash so zreťazením. Najprv si dám všade do tabuľky NULL. Prvky potom vkladám do tabuľky na index, ktorý si vypočítam hashovacím kľúčom (číslo mod veľkosť tabuľky).

V tejto metóde ide o to, že tie prvky sa môžu nachádzať aj mimo tabuľky pomocou spájaného zoznamu. Ak je v tabuľke NULL, čiže ešte som nevkladala žiadnený prvak na daný index, vložím tam zadaný prvak a ten zatial' ukazuje na ďalší - null. Ak mi výjde druhýkrát rovnaký index, v tabuľke už mám vložené jedno číslo a to číslo mi v spájanom zozname bude odkazovať na toto nové číslo (akoby vložím na koniec reťazca) a novému číslu priradím zatial' ukazovateľa na null.

Napr. – veľkosť tabuľky je 13, čiže pole je od pole[0] po pole[12] a vkladám prvky z obrázka

```
velkosť tabuľky: 13
tabulka[0] -> NULL
tabulka[1] -> NULL
tabulka[2] -> NULL
tabulka[3] -> NULL
tabulka[4] -> 30 -> NULL
tabulka[5] -> 5 -> 70 -> NULL
tabulka[6] -> NULL
tabulka[7] -> 20 -> NULL
tabulka[8] -> NULL
tabulka[9] -> NULL
tabulka[10] -> 10 -> 140 -> NULL
tabulka[11] -> NULL
tabulka[12] -> 12 -> NULL
→ tabulka[12] -> 12 -> NULL
```

Napr. – veľkosť tabuľky je 6, čiže pole je od pole[0] po pole[6] a vkladám prvky z obrázka

```
insert(50);
insert(700);
insert(85);
insert(92);
insert(73);
insert(101); →
insert(76);    tabulka[0] -> 700 -> NULL
                tabulka[1] -> 50 -> 85 -> 92 -> NULL
                tabulka[2] -> NULL
                tabulka[3] -> 73 -> 101 -> NULL
                tabulka[4] -> NULL
                tabulka[5] -> NULL
                tabulka[6] -> 76 -> NULL
```

Ak hľadám prvok, pošlem si do funkcie search hľadané číslo a vypočítam si index, na ktorom by sa to číslo malo nachádzať a prehľadávam ten spájaný zoznam na tom indexe. Ak sa tam nachádza, vypíšem „nasiel“ inak „nenasiel“.

Napr. – hľadám či sa tam nachádza číslo 10

```
int hladany = 10;
printf("Hľadany: %d\n", hladany);

if(search(hladany))
{
    printf("Nasiel\n");
}
else
{
    printf("Nenasiel\n");
}
```

C:\Users\Anna\Desktop\Moj hash\Projekt1.exe

```
velkosť tabuľky: 7
tabulka[0] -> 700 -> NULL
tabulka[1] -> 50 -> 85 -> 92 -> NULL
tabulka[2] -> NULL
tabulka[3] -> 73 -> 101 -> NULL
tabulka[4] -> NULL
tabulka[5] -> NULL
tabulka[6] -> 76 -> NULL
Hľadany: 10
Nenasiel
```

## 5. PREVZATÝ HASH S OTVORENÝM ADRESOVANÍM

Kód je prevzatý zo stránky:

Neeraj Mishra, Fouder of www.thecrazyprogrammer.com, Ultimate List of Software Developer Blogs

<https://www.thecrazyprogrammer.com/2017/06/hashing.html>

V tejto hashovacej funkcií nemôže byť žiadny prvok mimo tabuľky. Ak mám index, na ktorý chcem daný prvok vložiť, tak ho vložím. Ak potom na ten istý index chcem vložiť iný prvok, musím prechádzať tabuľku od daného indexu a hľadať najbližšie voľné miesto, kde ho môžem vložiť – skúšanie (hľadanie voľnej pozície).

Napr. - hashovacia tabuľka má veľkosti 5, ktorá má funkciu : mod 5 a už je obsadená na indexoch 0,2,3.

index 0	index 1	index 2	index 3	index 4	index 5
3		6	1		

Teraz chcem vložiť 10.

$10 \bmod 5 = 0$ , ale 0 je obsadená, idem ďalej, index 1 je voľný, takže 10 vloží na 1.



index 0	index 1	index 2	index 3	index 4	index 5
3	10	6	1		

Teraz chcem vložiť 11.

$11 \bmod 5 = 1$ , na indexe 1 už je 10, skontroluje ďalej, 2 je obsadené, skontroluje vedľa, 3 je obsadené, ide vedľa, 4 je voľné, teda tam vloží 11.



index 0	index 1	index 2	index 3	index 4	index 5
3	10	6	1	11	

## 6. TESTOVANIE

Vytvorila som si 4 hlavičkové súbory (avl.h, chain.h, splay.h, open.h) do ktorých som si vložila iba funkcie a potom jeden hlavný main v ktorom som definovala tieto .h súbory a volala ich funkcie.

Čas som testovala pomocou funkcie clock().

```

start = clock();
printf("Zaciatok casu = %ld\n", start);
for (a = 1; a < 10000; a++)
{
    binary_tree = insert(binary_tree, a);
}
end = clock();
printf("Koniec = %ld\n", end);

printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);

```

## 6.1. TESTOVANIE AVL A SPLAY STROMOV

- porovnám vkladanie (funkcia insert) medzi avl a splay stromom
- pre čísla (1-999, 1-9999, 1-99999, 1-999999, 1-49999, 1-499999, 1-4999999, 1-79999)
- čísla idú postupne, čiže sú zoradene vzostupne

-vkladanie zoradených čísel od 1-999 do avl stromu (iba insert)

čas: 0,0010

```
9 int main() {
10     STROM* binary_tree = NULL;
11     int a;
12     clock_t start, end;
13
14     start = clock();
15     printf("Zaciatok casu = %ld\n", start);
16     for (a = 1; a < 1000; a++)
17     {
18         binary_tree = insert(binary_tree, a);
19     }
20     end = clock();
21     printf("Koniec = %ld\n", end);
22
23     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
```

```
C:\Users\Anna\Desktop\Testy\main.exe
Zaciatok casu = 1
Koniec = 2
Cas: 0.0010

-----
Process exited after 0.05958 seconds with return value 0
Press any key to continue . . .
```

-vkladanie zoradených čísel od 1-999 do splay stromu (iba insert)

čas: 0,0010

```
13     splay_tree *t = new_splay_tree();
14
15     start = clock();
16     printf("Zaciatok casu = %ld\n", start);
17     for (a = 1; a < 1000; a++)
18     {
19         insertSplay(t, new_node(a));
20     }
21     end = clock();
22     printf("Koniec = %ld\n", end);
23
24     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
```

```
C:\Users\Anna\Desktop\Testy\main.exe
Zaciatok casu = 1
Koniec = 2
Cas: 0.0010

-----
Process exited after 0.03538 seconds with return value 0
Press any key to continue . . .
```

-vkladanie zoradených čísel od 1-9999 do avl stromu (iba insert)

čas: 0,0030

```
8
9 int main() {
10     STROM* binary_tree = NULL;
11     int a;
12     clock_t start, end;
13
14     start = clock();
15     printf("Zaciatok casu = %ld\n", start);
16     for (a = 1; a < 10000; a++)
17     {
18         binary_tree = insert(binary_tree, a);
19     }
20     end = clock();
21     printf("Koniec = %ld\n", end);
22
23     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
```

```
C:\Users\Anna\Desktop\Testy\main.exe
Zaciatok casu = 1
Koniec = 4
Cas: 0.0030

-----
Process exited after 0.05078 seconds with return value 0
Press any key to continue . . .
```

-vkladanie zoradených čísel od 1-9999 do splay stromu (iba insert)

čas: 0,0010

```

13     splay_tree *t = new_splay_tree();
14
15     start = clock();
16     printf("Zaciatok casu = %ld\n", start);
17     for (a = 1; a < 10000; a++)
18     {
19         insertSplay(t, new_node(a));
20     }
21     end = clock();
22     printf("Koniec = %ld\n", end);
23
24     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);

```

C:\Users\Anna\Desktop\Testy\\

Zaciatok casu = 1  
Koniec = 2  
Cas: 0.0010

-----  
Process exited after 0.  
Press any key to continue

-vkladanie zoradených čísel od 1-99999 do avl stromu (iba insert)  
čas: 0,0340

```

8
9 int main()
10    STROM* binary_tree = NULL;
11    int a;
12    clock_t start, end;
13
14    start = clock();
15    printf("Zaciatok casu = %ld\n", start);
16    for (a = 1; a < 10000; a++)
17    {
18        binary_tree = insert(binary_tree, a);
19    }
20    end = clock();
21    printf("Koniec = %ld\n", end);
22
23    printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
24

```

C:\Users\Anna\Desktop\Testy\\

Zaciatok casu = 1  
Koniec = 35  
Cas: 0.0340

-----  
Process exited after 0.07  
Press any key to continue

-vkladanie zoradených čísel od 1-99999 do splay stromu (iba insert)  
čas: 0,0060

```

13     splay_tree *t = new_splay_tree();
14
15     start = clock();
16     printf("Zaciatok casu = %ld\n", start);
17     for (a = 1; a < 10000; a++)
18     {
19         insertSplay(t, new_node(a));
20     }
21     end = clock();
22     printf("Koniec = %ld\n", end);
23
24     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
25

```

C:\Users\Anna\Desktop\

Zaciatok casu = 0  
Koniec = 6  
Cas: 0.0060

-----  
Process exited after  
Press any key to continue

-vkladanie zoradených čísel od 1-999999 do avl stromu (iba insert)  
čas: 0,3190

```

8
9 int main()
10    STROM* binary_tree = NULL;
11    int a;
12    clock_t start, end;
13
14    start = clock();
15    printf("Zaciatok casu = %ld\n", start);
16    for (a = 1; a < 1000000; a++)
17    {
18        binary_tree = insert(binary_tree, a);
19    }
20    end = clock();
21    printf("Koniec = %ld\n", end);
22
23    printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
24

```

C:\Users\Anna\Desktop\Testy\\

Zaciatok casu = 2  
Koniec = 321  
Cas: 0.3190

-----  
Process exited after 0.390  
Press any key to continue

-vkladanie zoradených čísel od 1-999999 do splay stromu (iba insert)

čas: 0,0750

```
13     splay_tree *t = new_splay_tree();
14
15     start = clock();
16     printf("Zaciatoč casu = %ld\n", start);
17     for (a = 1; a < 1000000; a++)
18     {
19         insertSplay(t, new_node(a));
20     }
21     end = clock();
22     printf("Koniec = %ld\n", end);
23
24     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
25
```

```
C:\Users\Anna\Desktop\1
Zaciatoč casu = 0
Koniec = 75
Cas: 0.0750
-----
Process exited after 6
Press any key to continue
```

-vkladanie zoradených čísel od 1-49999 do avl stromu (iba insert)

čas: 0,0210

```
9 int main()
10 STROM* binary_tree = NULL;
11 int a;
12 clock_t start, end;
13
14 start = clock();
15 printf("Zaciatoč casu = %ld\n", start);
16 for (a = 1; a < 50000; a++)
17 {
18     binary_tree = insert(binary_tree, a);
19 }
20 end = clock();
21 printf("Koniec = %ld\n", end);
22
23 printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
```

```
C:\Users\Anna\Desktop\Testy
Zaciatoč casu = 1
Koniec = 22
Cas: 0.0210
-----
Process exited after 0.08
Press any key to continue
```

-vkladanie zoradených čísel od 1-49999 do splay stromu (iba insert)

čas: 0,0050

```
13     splay_tree *t = new_splay_tree();
14
15     start = clock();
16     printf("Zaciatoč casu = %ld\n", start);
17     for (a = 1; a < 50000; a++)
18     {
19         insertSplay(t, new_node(a));
20     }
21     end = clock();
22     printf("Koniec = %ld\n", end);
23
24     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
25
```

```
C:\Users\Anna\Desktop
Zaciatoč casu = 0
Koniec = 5
Cas: 0.0050
-----
Process exited after
Press any key to cont
```

-vkladanie zoradených čísel od 1-499999 do avl stromu (iba insert)

čas: 0,1580

```

8
9 int main() {
10     STROM* binary_tree = NULL;
11     int a;
12     clock_t start, end;
13
14     start = clock();
15     printf("Zaciatok casu = %ld\n", start);
16     for (a = 1; a < 500000; a++)
17     {
18         binary_tree = insert(binary_tree, a);
19     }
20     end = clock();
21     printf("Koniec = %ld\n", end);
22
23     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
24
25

```

```

C:\Users\Anna\Desktop\Testy\n
Zaciatoč casu = 2
Koniec = 160
Cas: 0.1580

-----
Process exited after 0.207
Press any key to continue

```

-vkladanie zoradených čísel od 1-499999 do splay stromu (iba insert)

čas: 0,0370

```

13     clock_t start, end,
14     splay_tree *t = new_splay_tree();
15
16     start = clock();
17     printf("Zaciatok casu = %ld\n", start);
18     for (a = 1; a < 500000; a++)
19     {
20         insertSplay(t, new_node(a));
21     }
22     end = clock();
23     printf("Koniec = %ld\n", end);
24
25     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);

```

```

C:\Users\Anna\Desktop\Testy\n
Zaciatoč casu = 0
Koniec = 37
Cas: 0.0370

-----
Process exited after 0.110
Press any key to continue

```

-vkladanie zoradených čísel od 1-4999999 do avl stromu (iba insert)

čas: 1,6990

```

9 int main() {
10     STROM* binary_tree = NULL;
11     int a;
12     clock_t start, end;
13
14     start = clock();
15     printf("Zaciatok casu = %ld\n", start);
16     for (a = 1; a < 5000000; a++)
17     {
18         binary_tree = insert(binary_tree, a);
19     }
20     end = clock();
21     printf("Koniec = %ld\n", end);
22
23     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
24

```

```

C:\Users\Anna\Desktop\T
Zaciatoč casu = 1
Koniec = 1700
Cas: 1.6990

-----
Process exited after 1
Press any key to conti

```

-vkladanie zoradených čísel od 1-4999999 do splay stromu (iba insert)

čas: 0,3740

```

13     splay_tree *t = new_splay_tree();
14
15     start = clock();
16     printf("Zaciatok casu = %ld\n", start);
17     for (a = 1; a < 5000000; a++)
18     {
19         insertSplay(t, new_node(a));
20     }
21     end = clock();
22     printf("Koniec = %ld\n", end);
23
24     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);

```

```

C:\Users\Anna\Desktop\T
Zaciatoč casu = 0
Koniec = 374
Cas: 0.3740

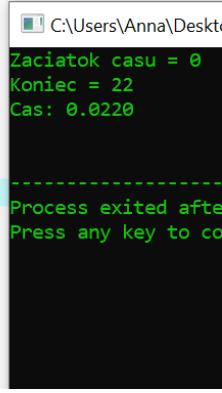
-----
Process exited after
Press any key to conti

```

-vkladanie zoradených čísel od 1-79999 do avl stromu (iba insert)

čas: 0,0220

```
9  int main() {
10    STROM* binary_tree = NULL;
11    int a;
12    clock_t start, end;
13
14    start = clock();
15    printf("Zaciatoč casu = %ld\n", start);
16    for (a = 1; a < 80000; a++)
17    {
18      binary_tree = insert(binary_tree, a);
19    }
20    end = clock();
21    printf("Koniec = %ld\n", end);
22
23    printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
24
```



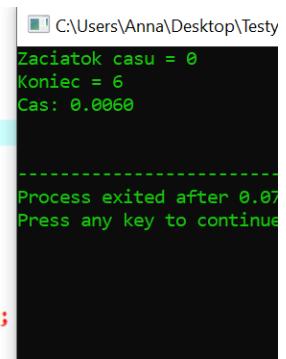
Zaciatoč casu = 0  
Koniec = 22  
Cas: 0.0220

Process exited after  
Press any key to continue

-vkladanie zoradených čísel od 1-79999 do splay stromu (iba insert)

čas: 0,0060

```
12
13    splay_tree *start, end;
14    splay_tree *t = new_splay_tree();
15
16    start = clock();
17    printf("Zaciatoč casu = %ld\n", start);
18    for (a = 1; a < 80000; a++)
19    {
20      insertSplay(t, new_node(a));
21    }
22    end = clock();
23    printf("Koniec = %ld\n", end);
24
25    printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
26
```



Zaciatoč casu = 0  
Koniec = 6  
Cas: 0.0060

Process exited after 0.07  
Press any key to continue

## ZHODNOTENIE

Pri vkladaní čísel do avl a splay stromu vyšiel čas pri splay menší, čiže vkladanie do splay stromu trvá kratšie ako do avl stromu.

-porovnám hľadanie (funkcia search) medzi avl a splay stromom

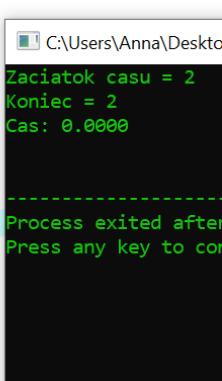
-vložím čísla (1-5000, 1-50000, 1-500000, 1-1000000)

-vo vložených číslach hľadám číslo približne v polovici

-hľadanie v zoradených číslach od 1-5000 do avl stromu (iba search)

čas: málo prvkov

```
12
13    int a;
14    for (a = 1; a < 5000; a++)
15    {
16      binary_tree = insert(binary_tree, a);
17    }
18
19    start = clock();
20    printf("Zaciatoč casu = %ld\n", start);
21
22    STROM* hladany = search(binary_tree, 3000);
23
24    end = clock();
25    printf("Koniec = %ld\n", end);
26
27    printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
28
```



Zaciatoč casu = 2  
Koniec = 2  
Cas: 0.0000

Process exited after  
Press any key to continue

-hľadanie v zoradených číslach od 1-5000 do splay stromu (iba search)

čas: 0,0010

```
12     splay_tree *t = new_splay_tree();
13
14     int a;
15     for (a = 1; a < 5000; a++)
16     {
17         insertSplay(t, new_node(a));
18     }
19
20     start = clock();
21     printf("Zaciatoč casu = %ld\n", start);
22
23     | searchSplay(t, new_node(3000),3000);
24
25     end = clock();
26     printf("Koniec = %ld\n", end);
27
28     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
29
... . . .
```

```
C:\Users\Anna\Desktop\Testy\m
Zaciatoč casu = 1
Koniec = 2
Cas: 0.0010
-----
Process exited after 0.0010
Press any key to continue .
```

-hľadanie v zoradených číslach od 1-50000 do avl stromu (iba search)

čas: málo prvkov

```
12
13     int a;
14     for (a = 1; a < 50000; a++)
15     {
16         binary_tree = insert(binary_tree, a);
17     }
18
19     start = clock();
20     printf("Zaciatoč casu = %ld\n", start);
21
22     | STROM* hladany = search(binary_tree, 40000);
23     //printf("hladany: %d\n", hladany);
24
25     end = clock();
26     printf("Koniec = %ld\n", end);
27
28     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
29
```

```
C:\Users\Anna\Desktop\Testy\m
Zaciatoč casu = 13
Koniec = 13
Cas: 0.0000
-----
Process exited after 0.0000
Press any key to continue .
```

-hľadanie v zoradených číslach od 1-50000 do splay stromu (iba search)

čas: málo prvkov

```
12     splay_tree *t = new_splay_tree();
13
14     int a;
15     for (a = 1; a < 50000; a++)
16     {
17         insertSplay(t, new_node(a));
18     }
19
20     start = clock();
21     printf("Zaciatoč casu = %ld\n", start);
22
23     | searchSplay(t, new_node(40000),40000);
24
25     end = clock();
26     printf("Koniec = %ld\n", end);
27
28     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
29
```

```
C:\Users\Anna\Desktop\Testy\m
Zaciatoč casu = 6
Koniec = 6
Cas: 0.0000
-----
Process exited after 0.0000
Press any key to continue .
```

-hľadanie v zoradených číslach od 1-500000 do avl stromu (iba search)

čas: 0,0010

```

12
13     int a;
14     for (a = 1; a < 500000; a++)
15     {
16         binary_tree = insert(binary_tree, a);
17     }
18
19     start = clock();
20     printf("Zaciatok casu = %ld\n", start);
21
22     STROM* hladany = search(binary_tree, 250000);
23     //printf("hladany: %d\n", hladany);
24
25     end = clock();
26     printf("Koniec = %ld\n", end);
27
28     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
29

```

C:\Users\Anna\Desktop\Testy  
Zaciatok casu = 156  
Koniec = 157  
Cas: 0.0010  
-----  
Process exited after 0.21  
Press any key to continue

-hľadanie v zoradených číslach od 1-500000 do splay stromu (iba search)

čas: 0,0010

```

11 // STROM* binary_tree = NULL;
12 splay_tree *t = new_splay_tree();
13
14 int a;
15 for (a = 1; a < 500000; a++)
16 {
17     insertSplay(t, new_node(a));
18 }
19
20 start = clock();
21 printf("Zaciatok casu = %ld\n", start);
22
23 searchSplay(t, new_node(250000), 250000);
24
25 end = clock();
26 printf("Koniec = %ld\n", end);
27
28 printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
29

```

C:\Users\Anna\Desktop  
Zaciatok casu = 43  
Koniec = 44  
Cas: 0.0010  
-----  
Process exited after  
Press any key to cont

-hľadanie v zoradených číslach od 1-1000000 do avl stromu (iba search)

čas: 0,0010

```

11 STROM* binary_tree = NULL;
12
13 int a;
14 for (a = 1; a < 1000000; a++)
15 {
16     binary_tree = insert(binary_tree, a);
17 }
18
19 start = clock();
20 printf("Zaciatok casu = %ld\n", start);
21
22 STROM* hladany = search(binary_tree, 600000);
23 //printf("hladany: %d\n", hladany);
24
25 end = clock();
26 printf("Koniec = %ld\n", end);
27
28 printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
29

```

C:\Users\Anna\Desktop\  
Zaciatok casu = 327  
Koniec = 328  
Cas: 0.0010  
-----  
Process exited after  
Press any key to cont

-hľadanie v zoradených číslach od 1-1000000 do avl stromu (iba search)

čas: 0,1260

```

12     splay_tree *t = new_splay_tree();
13
14     int a;
15     for (a = 1; a < 1000000; a++)
16     {
17         insertSplay(t, new_node(a)) ;
18     }
19
20     start = clock();
21     printf("Zaciatok casu = %ld\n", start);
22
23     searchSplay(t, new_node(600000),600000);
24
25     end = clock();
26     printf("Koniec = %ld\n", end);
27
28     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);

```

C:\Users\Anna\Desktop\  
Zaciatok casu = 79  
Koniec = 205  
Cas: 0.1260  
  
-----  
Process exited after 0 seconds  
Press any key to continue . . .

## ZHODNOTENIE

Pri hľadaní čísel (cca v polovici vloženia) v avl a splay strome, vyšiel dlhší čas pri hľadaní v splay strome, takže v splay to trvá dlhšie ako v avl.

### -vkladám postupne zoradené čísla a hľadám náhodné číslo

-vkladanie a hľadanie(5000) v zoradených číslach od 1-10000 do avl stromu (insert a search)  
čas: 0,0020

```

12     STROM* binary_tree = NULL;
13     int a;
14
15     start = clock();
16     printf("Zaciatok casu = %ld\n", start);
17
18     for (a = 1; a < 10000; a++)
19     {
20         binary_tree = insert(binary_tree, a);
21     }
22     search(binary_tree,5000);
23
24     end = clock();
25     printf("Koniec = %ld\n", end);
26
27     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
28

```

C:\Users\Anna\Desktop\  
Zaciatok casu = 0  
Koniec = 2  
Cas: 0.0020  
  
-----  
Process exited after 0 seconds  
Press any key to continue . . .

-vkladanie a hľadanie(5000) v zoradených číslach od 1-10000 do splay stromu (insert a search)

čas: 0,0030

```

11     STROM* binary_tree = NULL;
12     splay_tree *t = new_splay_tree();
13
14     int a;
15
16     start = clock();
17     printf("Zaciatok casu = %ld\n", start);
18
19     for (a = 1; a < 10000; a++)
20     {
21         insertSplay(t, new_node(a)) ;
22     }
23     searchSplay(t, new_node(5000),5000);
24
25     end = clock();
26     printf("Koniec = %ld\n", end);
27
28     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
29

```

C:\Users\Anna\Desktop\  
Zaciatok casu = 1  
Koniec = 4  
Cas: 0.0030  
  
-----  
Process exited after 0 seconds  
Press any key to continue . . .

-vkladanie a hľadanie(1000) v zoradených číslach od 1-80000 do avl stromu (insert a search)  
čas: 0,0240

```
12     STROM* binary_tree = NULL;
13     int a;
14
15     start = clock();
16     printf("Zaciatok casu = %ld\n", start);
17
18     for (a = 1; a < 80000; a++)
19     {
20         binary_tree = insert(binary_tree, a);
21     }
22     search(binary_tree, 1000);
23
24     end = clock();
25     printf("Koniec = %ld\n", end);
26
27     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
28
```

C:\Users\Anna\Desktop  
Zaciatok casu = 0  
Koniec = 24  
Cas: 0.0240  
-----  
Process exited after 0.0240 seconds  
Press any key to continue . . .

-vkladanie a hľadanie(1000) v zoradených číslach od 1-80000 do splay stromu (insert a search)

čas: 0,0090

```
12     splay_tree *t = new_splay_tree();
13
14     int a;
15
16     start = clock();
17     printf("Zaciatok casu = %ld\n", start);
18
19     for (a = 1; a < 80000; a++)
20     {
21         insertSplay(t, new_node(a));
22     }
23     searchSplay(t, new_node(1000), 1000);
24
25     end = clock();
26     printf("Koniec = %ld\n", end);
27
28     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
29
```

Zaciatok casu = 0  
Koniec = 9  
Cas: 0.0090  
-----  
Process exited after 0.0090 seconds  
Press any key to continue . . .

-vkladanie a hľadanie(132000) v zoradených číslach od 1-250000 do avl stromu (insert a search)

čas: 0,0820

```
12     STROM* binary_tree = NULL;
13     int a;
14
15     start = clock();
16     printf("Zaciatok casu = %ld\n", start);
17
18     for (a = 1; a < 250000; a++)
19     {
20         binary_tree = insert(binary_tree, a);
21     }
22     search(binary_tree, 132000);
23
24     end = clock();
25     printf("Koniec = %ld\n", end);
26
27     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
28
```

C:\Users\Anna\Desktop\  
Zaciatok casu = 4  
Koniec = 86  
Cas: 0.0820  
-----  
Process exited after 0.0820 seconds  
Press any key to continue . . .

-vkladanie a hľadanie(132000) v zoradených číslach od 1-250000 do splay stromu (insert a search)

čas: 0,0200

```

12     splay_tree *t = new_splay_tree();
13
14     int a;
15
16     start = clock();
17     printf("Zaciatok casu = %ld\n", start);
18
19     for (a = 1; a < 250000; a++)
20     {
21         insertSplay(t, new_node(a));
22     }
23     searchSplay(t, new_node(132000), 132000);
24
25     end = clock();
26     printf("Koniec = %ld\n", end);
27
28     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
29

```

```

C:\Users\Anna\Desktop
Zaciatok casu = 1
Koniec = 21
Cas: 0.0200
-----
Process exited after
Press any key to continue...

```

-vkladanie a hľadanie(12000) v zoradených číslach od 1-600000 do avl stromu (insert a search)

čas: 0,1980

```

12     STROM* binary_tree = NULL;
13     int a;
14
15     start = clock();
16     printf("Zaciatok casu = %ld\n", start);
17
18     for (a = 1; a < 600000; a++)
19     {
20         binary_tree = insert(binary_tree, a);
21     }
22     search(binary_tree, 12000);
23
24     end = clock();
25     printf("Koniec = %ld\n", end);
26
27     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
28

```

```

C:\Users\Anna\Desktop
Zaciatok casu = 2
Koniec = 200
Cas: 0.1980
-----
Process exited after
Press any key to continue...

```

-vkladanie a hľadanie(12000) v zoradených číslach od 1-600000 do splay stromu (insert a search)

čas: 0,0460

```

12     splay_tree *t = new_splay_tree();
13
14     int a;
15
16     start = clock();
17     printf("Zaciatok casu = %ld\n", start);
18
19     for (a = 1; a < 600000; a++)
20     {
21         insertSplay(t, new_node(a));
22     }
23     searchSplay(t, new_node(12000), 12000);
24
25     end = clock();
26     printf("Koniec = %ld\n", end);
27
28     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
29

```

```

C:\Users\Anna\Desktop
Zaciatok casu = 1
Koniec = 47
Cas: 0.0460
-----
Process exited after
Press any key to continue...

```

-vkladanie a hľadanie(1562000) v zoradených číslach od 1-2000000 do avl stromu (insert a search)

čas: 0,6650

```
10     clock_t start, end;
11     splay_tree *t = new_splay_tree();
12     STROM* binary_tree = NULL;
13     int a;
14
15     start = clock();
16     printf("Zaciatoč casu = %ld\n", start);
17
18     for (a = 1; a < 2000000 ; a++)
19     {
20         binary_tree = insert(binary_tree, a);
21     }
22     search(binary_tree,1562000);
23
24     end = clock();
25     printf("Koniec = %ld\n", end);
26
27     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
28 
```

Zaciatoč casu = 2  
Koniec = 667  
Cas: 0.6650  
-----  
Process exited after 0.  
Press any key to continue...

-vkladanie a hľadanie(1562000) v zoradených číslach od 1-2000000 do splay stromu (insert a search)

čas: 0,1600

```
12     splay_tree *t = new_splay_tree();
13
14     int a;
15
16     start = clock();
17     printf("Zaciatoč casu = %ld\n", start);
18
19     for (a = 1; a < 2000000; a++)
20     {
21         insertSplay(t, new_node(a));
22     }
23     searchSplay(t, new_node(1562000),1562000);
24
25     end = clock();
26     printf("Koniec = %ld\n", end);
27
28     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
29 
```

C:\Users\Anna\Desktop\Zadanie 2\zadanie2\zadanie2\main.cpp Zaciatoč casu = 3  
Koniec = 163  
Cas: 0.1600  
-----  
Process exited after 0.  
Press any key to continue...

## ZHODNOTENIE

Kvôli vkladaniu, ktoré trvá dlhšie avl stromu, trvá aj teraz vkladanie a hľadanie dlhšie ako pri splay strome.

**-vkladám takmer zoradené čísla blízko seba(napr. 4 3 6 5 8 7)**

-vkladanie skoro zoradených čísel od 1-8000 do avl stromu (insert)

čas: 0,0020

```

        ...
start = clock();
printf("Zaciatok casu = %ld\n", start);

for (a = 1; a < 8000 ; a++)
{
    binary_tree = insert(binary_tree, a+3);
    binary_tree = insert(binary_tree, a+2);
    a++;
}

end = clock();
printf("Koniec = %ld\n", end);

printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);

```

```

C:\Users\Anna\Desktop\Tes
Zaciatok casu = 0
Koniec = 2
Cas: 0.0020
-----
Process exited after 0.0020 seconds
Press any key to continue...

```

-vkladanie skoro zoradených čísel od 1-8000 do splay stromu (insert)  
čas: 0,0010

```

!7
!8     start = clock();
!9     printf("Zaciatok casu = %ld\n", start);
!10
!11    for (a = 1; a < 8000; a++)
!12    {
!13        insertSplay(t, new_node(a+3));
!14        insertSplay(t, new_node(a+2));
!15        a++;
!16    }

!17
!18    end = clock();
!19    printf("Koniec = %ld\n", end);
!20
!21    printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
!22

```

```

C:\Users\Anna\Desktop\1
Zaciatok casu = 0
Koniec = 1
Cas: 0.0010
-----
Process exited after 0.0010 seconds
Press any key to continue...

```

-vkladanie skoro zoradených čísel od 1-50000 do avl stromu (insert)  
čas: 0,0140

```

28     start = clock();
29     printf("Zaciatok casu = %ld\n", start);
30
31     for (a = 1; a < 50000 ; a++)
32     {
33         binary_tree = insert(binary_tree, a+3);
34         binary_tree = insert(binary_tree, a+2);
35         a++;
36     }

37
38     end = clock();
39     printf("Koniec = %ld\n", end);
40
41     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
42

```

```

C:\Users\Anna\Desktop\1
Zaciatok casu = 0
Koniec = 14
Cas: 0.0140
-----
Process exited after 0.0140 seconds
Press any key to continue...

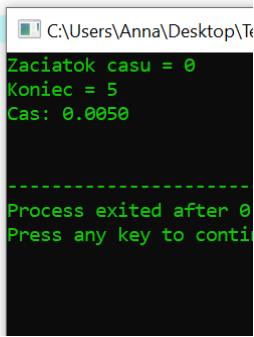
```

-vkladanie skoro zoradených čísel od 1-50000 do splay stromu (insert)  
čas: 0,0050

```

30
31     for (a = 1; a < 50000; a++)
32     {
33         insertSplay(t, new_node(a+3));
34         insertSplay(t, new_node(a+2));
35         a++;
36     }
37
38     end = clock();
39     printf("Koniec = %ld\n", end);
40
41     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
42

```



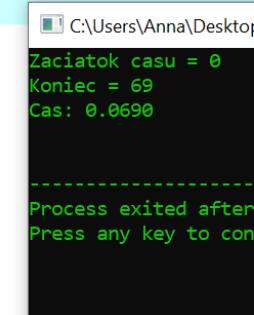
-vkladanie skoro zoradených čísel od 1-250000 do avl stromu (insert)

čas: 0,0690

```

30
31     for (a = 1; a < 250000 ; a++)
32     {
33         binary_tree = insert(binary_tree, a+3);
34         binary_tree = insert(binary_tree, a+2);
35         a++;
36     }
37
38     end = clock();
39     printf("Koniec = %ld\n", end);
40
41     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
42

```



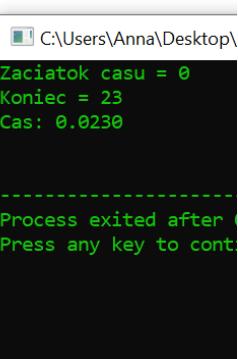
-vkladanie skoro zoradených čísel od 1-250000 do splay stromu (insert)

čas: 0,0230

```

28     start = clock();
29     printf("Zaciatok casu = %ld\n", start);
30
31     for (a = 1; a < 250000 ; a++)
32     {
33         insertSplay(t, new_node(a+3));
34         insertSplay(t, new_node(a+2));
35         a++;
36     }
37
38     end = clock();
39     printf("Koniec = %ld\n", end);
40
41     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
42

```



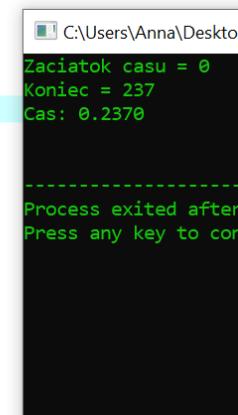
-vkladanie skoro zoradených čísel od 1-800000 do avl stromu (insert)

čas: 0,2370

```

27
28     start = clock();
29     printf("Zaciatok casu = %ld\n", start);
30
31     for (a = 1; a < 800000 ; a++)
32     {
33         binary_tree = insert(binary_tree, a+3);
34         binary_tree = insert(binary_tree, a+2);
35         a++;
36     }
37
38     end = clock();
39     printf("Koniec = %ld\n", end);
40
41     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
42

```



-vkladanie skoro zoradených čísel od 1-800000 do splay stromu (insert)

čas: 0,0860

```
28     start = clock();
29     printf("Zaciatok casu = %ld\n", start);
30
31     for (a = 1; a < 800000 ; a++)
32     {
33         insertSplay(t, new_node(a+3));
34         insertSplay(t, new_node(a+2));
35         a++;
36     }
37
38     end = clock();
39     printf("Koniec = %ld\n", end);
40
41     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
```

```
C:\Users\Anna\Desktop
Zaciatok casu = 0
Koniec = 86
Cas: 0.0860
-----
Process exited after
Press any key to cont
```

-vkladanie skoro zoradených čísel od 1-1500000 do avl stromu (insert)

čas: 0,4640

```
27
28     start = clock();
29     printf("Zaciatok casu = %ld\n", start);
30
31     for (a = 1; a < 1500000 ; a++)
32     {
33         binary_tree = insert(binary_tree, a+3);
34         binary_tree = insert(binary_tree, a+2);
35         a++;
36     }
37
38     end = clock();
39     printf("Koniec = %ld\n", end);
40
41     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
42
```

```
C:\Users\Anna\Desktop
Zaciatok casu = 0
Koniec = 464
Cas: 0.4640
-----
Process exited after
Press any key to cont
```

-vkladanie skoro zoradených čísel od 1-1500000 do splay stromu (insert)

čas: 0,1480

```
27
28     start = clock();
29     printf("Zaciatok casu = %ld\n", start);
30
31     for (a = 1; a < 1500000 ; a++)
32     {
33         insertSplay(t, new_node(a+3));
34         insertSplay(t, new_node(a+2));
35         a++;
36     }
37
38     end = clock();
39     printf("Koniec = %ld\n", end);
40
41     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
42
```

```
C:\Users\Anna\Desktop
Zaciatok casu = 2
Koniec = 150
Cas: 0.1480
-----
Process exited after
Press any key to cont
```

-vkladanie skoro zoradených čísel od 1-6000000 do avl stromu (insert)

čas: 1,9620

```

29     printf("Zaciatoč casu = %ld\n", start),
30
31     for (a = 1; a < 6000000 ; a++)
32     {
33         binary_tree = insert(binary_tree, a+3);
34         binary_tree = insert(binary_tree, a+2);
35         a++;
36     }
37
38     end = clock();
39     printf("Koniec = %ld\n", end);
40
41     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
42

```

C:\Users\Anna\Desktop  
Zaciatoč casu = 1  
Koniec = 1963  
Cas: 1.9620  
-----  
Process exited after  
Press any key to cont

-vkladanie skoro zoradených čísel od 1-6000000 do splay stromu (insert)  
čas: 0,5320

```

18     start = clock();
19     printf("Zaciatoč casu = %ld\n", start);
20
21     for (a = 1; a < 6000000;a++)
22     {
23         insertSplay(t, new_node(a+3));
24         insertSplay(t, new_node(a+2));
25         a++;
26     }
27
28     end = clock();
29     printf("Koniec = %ld\n", end);
30
31     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
32
33 //main.c(binarytree)

```

C:\Users\Anna\Desktop  
Zaciatoč casu = 0  
Koniec = 532  
Cas: 0.5320  
-----  
Process exited after  
Press any key to cont

## ZHODNOTENIE

Aj vkladanie nezoradených čísel do avl trvá dlhšie ako do splay stromu.

## 6.2. TESTOVANIE HASHOV SO ZREŤAZENÍM A SOTVORENÝM ADRESOVANÍM

-vkladanie 100 prvkov do tabuľky veľkej 10 so zreťazením  
čas: málo prvkov

```

12
13     tabulka_init();
14
15     start = clock();
16     printf("Zaciatoč casu = %ld\n", start);
17
18     for (a = 1; a < 100 ; a++)
19     {
20         insertChain(a);
21     }
22     end = clock();
23     printf("Koniec = %ld\n", end);
24
25     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
26

```

C:\Users\Anna\Desktop  
Zaciatoč casu = 2  
Koniec = 2  
Cas: 0.0000  
-----  
Process exited after  
Press any key to cont

-vkladanie 100 prvkov do tabuľky veľkej 10 so zreťazením  
čas: málo prvkov

-vkladanie 2000 prvkov do tabuľky veľkej 100 so zreťazením  
čas: 0,0010

```

12
13     tabulka_init();
14
15     start = clock();
16     printf("Zaciatok casu = %ld\n", start);
17
18     for (a = 1; a < 2000 ; a++)
19     {
20         insertChain(a);
21     }
22     end = clock();
23     printf("Koniec = %ld\n", end);
24
25     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
26

```

```

C:\Users\Anna\Desktop
Zaciatok casu = 2
Koniec = 3
Cas: 0.0010

-----
Process exited after
Press any key to con

```

-vkladanie 30000 prvkov do tabuľky veľkej 100 so zretežením  
čas: 0,3920

```

12
13     tabulka_init();
14
15     start = clock();
16     printf("Zaciatok casu = %ld\n", start);
17
18     for (a = 1; a < 30000 ; a++)
19     {
20         insertChain(a);
21     }
22     end = clock();
23     printf("Koniec = %ld\n", end);
24
25     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
26

```

```

C:\Users\Anna\Desktop
Zaciatok casu = 1
Koniec = 393
Cas: 0.3920

-----
Process exited after
Press any key to con

```

-vkladanie 30000 prvkov do tabuľky veľkej 1000 so zretežením  
čas: 0,3910

```

12
13     tabulka_init();
14
15     start = clock();
16     printf("Zaciatok casu = %ld\n", start);
17
18     for (a = 1; a < 30000 ; a++)
19     {
20         insertChain(a);
21     }
22     end = clock();
23     printf("Koniec = %ld\n", end);
24
25     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
26

```

```

C:\Users\Anna\Desktop
Zaciatok casu = 3
Koniec = 394
Cas: 0.3910

-----
Process exited after
Press any key to con

```

-vkladanie 100000 prvkov do tabuľky veľkej 1000 so zretežením  
čas: 5,2220

```

    ... o, ...

tabulka_init();

start = clock();
printf("Zaciatok casu = %ld\n", start);

for (a = 1; a < 100000 ; a++)
{
    insertChain(a);
}
end = clock();
printf("Koniec = %ld\n", end);

printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);

```

C:\Users\Anna\Desktop\

Zaciatok casu = 0  
Koniec = 5222  
Cas: 5.2220

-----  
Process exited after 5.2220 seconds  
Press any key to continue . . .

-vkladanie 200000 prvkov do tabuľky velkej 10000 so zretežením  
čas: 46,4850

```

12
13     tabulka_init();
14
15     start = clock();
16     printf("Zaciatok casu = %ld\n", start);
17
18     for (a = 1; a < 200000 ; a++)
19     {
20         insertChain(a);
21     }
22     end = clock();
23     printf("Koniec = %ld\n", end);
24
25     printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);

```

C:\Users\Anna\Desktop\

Zaciatok casu = 1  
Koniec = 46486  
Cas: 46.4850

-----  
Process exited after 46.4850 seconds  
Press any key to continue . . .

-hľadanie 19 prvkov v tabuľke velkej 100 so zretežením  
čas: 0,0420

```

        }

start = clock();
for (z=1; z<20; z++)
{
    searchChain(z);
}

printf("Koniec = %ld\n", end);
printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);

```

C:\Users\Anna\

Koniec = 42  
Cas: 0.0420

-----  
Process exited after 0.0420 seconds  
Press any key to continue . . .

-hľadanie 100 prvkov v tabuľke velkej 500 so zretežením  
čas: 0,0380

```

    insertChain(a);

}

start = clock();
for (z=1; z<100; z++)
{
    searchChain(z);

}

printf("Koniec = %ld\n", end);
printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);

```

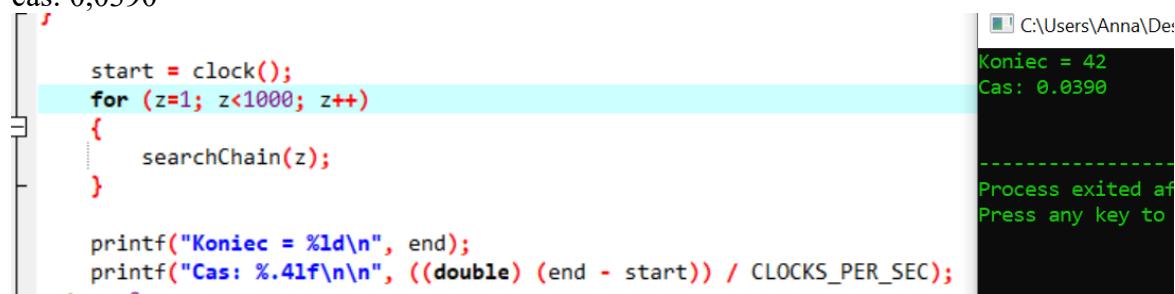
C:\Users\Anna\

Koniec = 42  
Cas: 0.0380

-----  
Process exited after 0.0380 seconds  
Press any key to continue . . .

-hľadanie 1000 prvkov v tabuľke veľkej 2000 so zretežením

čas: 0,0390



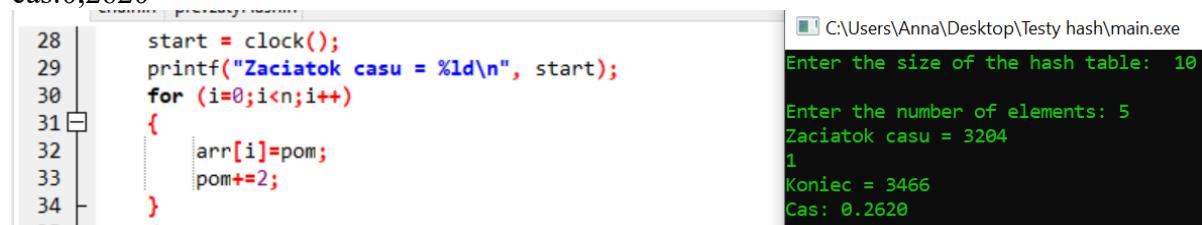
```
start = clock();
for (z=1; z<1000; z++)
{
    searchChain(z);
}

printf("Koniec = %ld\n", end);
printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
```

C:\Users\Anna\Desktop\Des Koniec = 42  
Cas: 0.0390  
-----  
Process exited af  
Press any key to

-vkladanie 5 prvkov do tabuľky veľkej 10 s otvorených adresovaním

čas: 0,2620

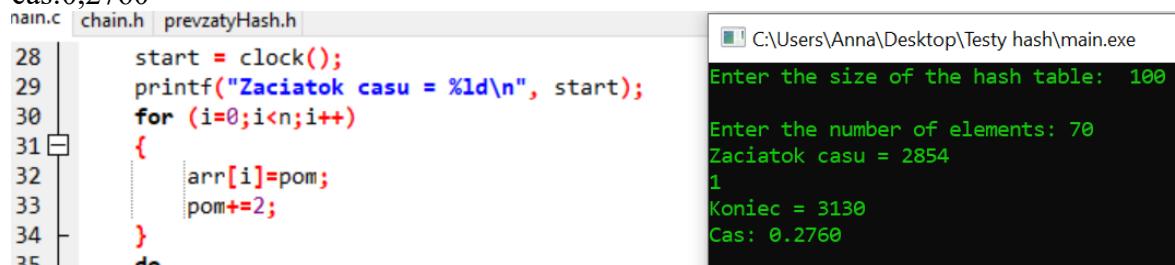


```
start = clock();
printf("Zaciatoč casu = %ld\n", start);
for (i=0;i<n;i++)
{
    arr[i]=pom;
    pom+=2;
}
```

C:\Users\Anna\Desktop\Testy hash\main.exe  
Enter the size of the hash table: 10  
Enter the number of elements: 5  
Zaciatoč casu = 3204  
1  
Koniec = 3466  
Cas: 0.2620

-vkladanie 70 prvkov do tabuľky veľkej 100 s otvorených adresovaním

čas: 0,2760

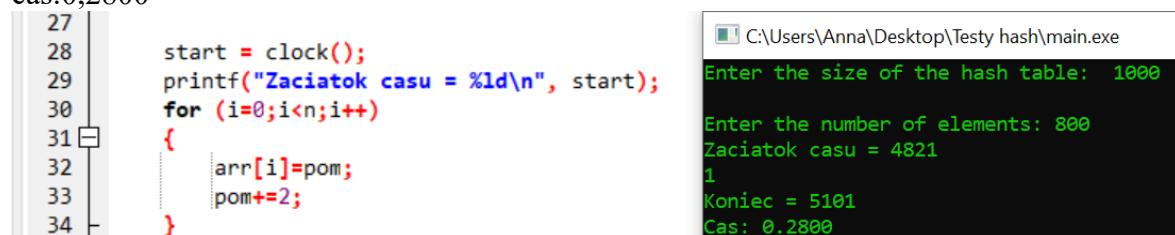


```
start = clock();
printf("Zaciatoč casu = %ld\n", start);
for (i=0;i<n;i++)
{
    arr[i]=pom;
    pom+=2;
}
```

C:\Users\Anna\Desktop\Testy hash\main.exe  
Enter the size of the hash table: 100  
Enter the number of elements: 70  
Zaciatoč casu = 2854  
1  
Koniec = 3130  
Cas: 0.2760

-vkladanie 800 prvkov do tabuľky veľkej 1000 s otvorených adresovaním

čas: 0,2800

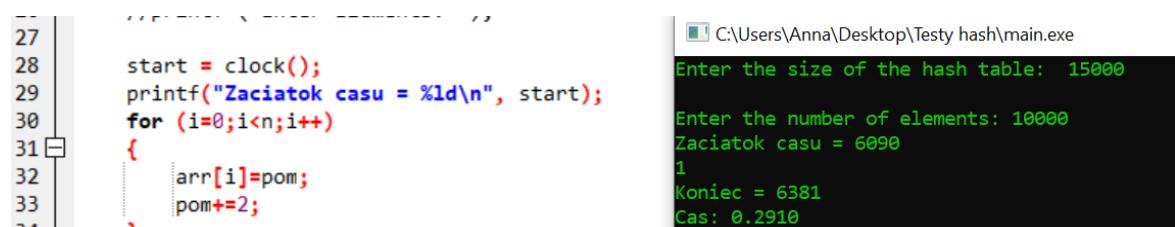


```
start = clock();
printf("Zaciatoč casu = %ld\n", start);
for (i=0;i<n;i++)
{
    arr[i]=pom;
    pom+=2;
}
```

C:\Users\Anna\Desktop\Testy hash\main.exe  
Enter the size of the hash table: 1000  
Enter the number of elements: 800  
Zaciatoč casu = 4821  
1  
Koniec = 5101  
Cas: 0.2800

-vkladanie 10000 prvkov do tabuľky veľkej 15000 s otvorených adresovaním

čas: 0,2910



```
start = clock();
printf("Zaciatoč casu = %ld\n", start);
for (i=0;i<n;i++)
{
    arr[i]=pom;
    pom+=2;
}
```

C:\Users\Anna\Desktop\Testy hash\main.exe  
Enter the size of the hash table: 15000  
Enter the number of elements: 10000  
Zaciatoč casu = 6090  
1  
Koniec = 6381  
Cas: 0.2910

-vkladanie 120000 prvkov do tabuľky veľkej 130000 s otvorených adresovaním  
čas:0,4950

```

25
26     //printf ("Enter Elements: ");
27
28     start = clock();
29     printf("Zaciatok casu = %ld\n", start);
30     for (i=0;i<n;i++)
31     {
32         arr[i]=pom;
33         pom+=2;

```

-hl'adanie 20 prvkov do tabuľky veľkej 100 s otvorených adresovaním  
čas:0,0390

```

start = clock();
for (i = 1; i<20 ;i++)
searchOpen(i,hFn,size);
printf("Koniec = %ld\n", end);
printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);
break;

```

-hl'adanie 500 prvkov do tabuľky veľkej 100 s otvorených adresovaním  
čas:0,0400

```

start = clock();
for (i = 1; i<500 ;i++)
searchOpen(i,hFn,size);
printf("Koniec = %ld\n", end);
printf("Cas: %.4lf\n\n", ((double) (end - start)) / CLOCKS_PER_SEC);

```

## 7. ZÁVER

### Výsledné tabuľky z porovnania medzi AVL a SPLAY stromami

- Tabuľka porovnania vkladania za sebou idúcich čísel do avl a splay stromu  
*Vkladanie do splay je rýchlejšie.*

	<b>AVL insert (zoradené čísla)</b>	<b>SPLAY insert (zoradené čísla)</b>
<b>1 - 199</b>	0,0010s	0,0010s
<b>1 - 9 999</b>	0,0030s	0,0010s
<b>1 - 49 999</b>	0,0210s	0,0050s
<b>1 - 99 999</b>	0,0340s	0,0060s
<b>1 - 499 999</b>	0,1580s	0,0370s
<b>1 - 999 999</b>	0,3190s	0,0750s
<b>1 - 4 999 999</b>	1,6990s	0,3740s
<b>1 - 9 999 999</b>	2,8910s	0,9810s

- Tabuľka porovnania hl'adania za sebou idúcich čísel v avl a splay strome  
*Hľadanie v avl je rýchlejšie.*  
V splay strome hľadá najrýchlejšie tie čísla, ktoré boli vložené nedávno lebo sú blízko koreňa. V avl to záleží od vyváženia.

	<b>AVL search (zoradené čísla)</b>	<b>SPLAY search (zoradené čísla)</b>
<b>1 - 4 999</b>	0,0000s	0,0010s
<b>1 - 49 999</b>	0,0000s	0,0000s
<b>1 - 499 999</b>	0,0010s	0,0010s
<b>1 - 999 999</b>	0,0010s	1,1260s

<b>1 – 4 999 999</b>	0,0680s	2,8900s
<b>1 – 9 999 999</b>	2,8990s	4,2340s

- Tabuľka porovnania vkladania a hľadania za sebou idúcich čísel v avl a splay strome  
Hľadanie a vkladanie v splay je rýchlejšie.

	<b>AVL search (zoradené čísla)</b>	<b>SPLAY search (zoradené čísla)</b>
<b>1 – 1 999</b>	0,0020s	0,0030s
<b>1 – 79 999</b>	0,0240s	0,0090s
<b>1 - 249 999</b>	0,0820s	0,0200s
<b>1 - 599 999</b>	0,1980s	0,0460s
<b>1 – 1 999 999</b>	0,6650s	0,1600s
<b>1 – 9 999 999</b>	1,9910s	1,2700s

- Tabuľka porovnania vkladania a hľadania skoro za sebou idúcich čísel (napr. 4 3 5 6 10 8) v avl a splay strome  
Hľadanie a vkladanie v splay je rýchlejšie.

	<b>AVL search (zoradené čísla)</b>	<b>SPLAY search (zoradené čísla)</b>
<b>1 – 7 999</b>	0,0020s	0,0010s
<b>1 – 49 999</b>	0,0140s	0,0050s
<b>1 - 249 999</b>	0,0690s	0,0230s
<b>1 - 799 999</b>	0,2370s	0,0860s
<b>1 – 1 499 999</b>	0,4640s	0,1480s
<b>1 – 5 999 999</b>	1,9620s	1,5320s

- Výsledná tabuľka porovnania AVL a SPLAY stromov

	<b>AVL STROM</b>	<b>SPLAY STROM</b>
<b>Insert (všeobecné čísla)</b>	Pomalšie	Rýchlejšie
<b>Search (všeobecné čísla)</b>	Rýchlejšie – keďže strom je lepšie vyvážený aj sa tam ľahšie hľadá	Pomalšie – strom nie je až tak dokonale vyvážený ako avl a tāžsie sa hľadá
<b>Insert + search</b>	Ked' samotné vkladanie trvá dlhšie, tak aj vkladanie spolu s hľadaním trvá dlhšie	Ked' samotné vkladanie trvá kratšie, tak aj vkladanie spolu s hľadaním trvá kratšie

### Výsledné tabuľky z porovnania medzi HASHOM SO ZREŤAZENÍM a HASHOM S OTVORENÝM ADRESOVANÍM.

- Tabuľka porovnania vkladania za sebou idúcich čísel do hashu so zreťazením a do hashu s otvoreným adresovaním  
Vkladanie s hashom so zreťazením je rýchlejšie.

<b>Počet prvkov</b>	<b>Veľkosť tabuľky</b>	<b>HASH so zreťazením (zoradené čísla)</b>	<b>HASH s otvoreným adresovaním (zoradené čísla)</b>
<b>99</b>	10	0,0000s	0,0000s
<b>1 999</b>	100	0,0010s	0,0010s

<b>29 999</b>	100	0,2920s	0,2950s
<b>29 999</b>	1000	0,3910s	0,4110s
<b>99 999</b>	1000	5,2220s	5,2390s
<b>199 999</b>	10 000	7,4850s	8,0400s
<b>499 999</b>	10 000	20,0400s	26,9800s
<b>1 499 999</b>	10 0000	39, 9080s	49,7630s

- Tabuľka porovnania hľadania za sebou idúcich čísel do hashu so zretežením a do hashu s otvoreným adresovaním

Hľadanie s hashom s otvoreným adresovaním je rýchlejšie.

Počet prvkov	Veľkosť tabuľky	HASH so zretežením (zoradené čísla)	HASH s otvoreným adresovaním (zoradené čísla)
<b>99</b>	10	0,0300s	0,0390s
<b>1999</b>	100	0,0390s	0,0400s
<b>29 999</b>	100	0,0980s	0,0620s
<b>29 999</b>	1000	0,0890s	0,0620s
<b>99 999</b>	1000	0,1990s	0,0910s
<b>199 999</b>	10 000	1,760s	1,3800s
<b>499 999</b>	10 000	3,9890s	1,9310s
<b>1 499 999</b>	10 0000	9,3400s	5,3070s

- Výsledná tabuľka porovnania hashu so zretežením a hashu s otvoreným adresovaním

	HASH - ZREŽAZENIE	HASH – OTVORENÉ ADRES.
<b>insert</b>	Rýchlejšie nájde voľné miesto	Pomalšie – dlho to trvá keď musí hľadať vhodné miesto
<b>search</b>	Pomalšie – trvá dlhšie keď musí hľadať v spájanom zozname v celom poli	Rýchlejšie – iba prechádza pole