Laboratorium nr 5

Program ma na celu przetestowanie czasu zapisu i usuwania z tablicy asocjacyjnej z tablicą z hashowaniem.
Słowa zostają tworzone losowo przy użyciu metody w pliku test.cpp(RandomString) i mają od 3 do 10 znaków, wliczając liczby oraz znaki specjalne.

Pomiar	10el.	100el.	1.000el.	10.000el.	100.000el.	1.000.000el.	10.000.000
Czas zapisu[ms]	0,01	0,063	0,417	5,192	53,067	407,736	3853,88
Czas odczytu[ms]	0,002	0,002	0,004	0,048	0,6-4,2	0,012-12,2	0,03-22,7

Liczba bucket'ów: 10

Pomiar	10el.	100el.	1.000el.	10.000el.	100.000el.	1.000.000el.	10.000.000
Czas zapisu[ms]	0,01	0,056	0,403	5,035	50,567	405,096	3856,21
Czas odczytu[ms]	0,002	0,003	0,003	0,012	0,095	1,137	0,7-17,0

Liczba bucket'ów: 100

Pomiar	10el.	100el.	1.000el.	10.000el.	100.000el.	1.000.000el.	10.000.000
Czas zapisu[ms]	0,01	0,055	0,415	5,123	49,905	410,009	3905,89
Czas odczytu[ms]	0,002	0,003	0,003	0,005	0,032	0,125	1,366

Liczba bucket'ów: 1.000

Pomiar	10el.	100el.	1.000el.	10.000el.	100.000el.	1.000.000el.	10.000.000
Czas zapisu[ms]	0,01	0,049	0,431	5,644	59,858	412,22	3887,1
Czas odczytu[ms]	0,002	0,002	0,002	0,004	0,004	0,016	0,112

Liczba bucket'ów: 10.000

Wnioski:

Zaletą tablicy asocjacyjnej jest szybkość odczytu danych. Podczas implementacji tablicy uznano, że nie zostanie zmodyfikowana wcześniej napisana implementacja listy i zostanie wymyślony sposób na konwersję danych typu string na int. Z pewnością poczyniona metoda hashowania nie jest bez wad i przy niesprzyjających warunkach mogą powstać kolizje, jednak podczas testów wykryto, iż jest to bardzo rzadkie zjawisko przy obecnych warunkach zadania. Minusem korzystania z wcześniejszej implementacji list jest konieczność wykorzystywania metody haszowania przy, nie tylko, każdorazowym zapisie danych ale również odczycie(kasowaniu), co spowalnia proces.
Uznano jednak, że skoro zastała konieczność zaimplementowania metody hashującej do programu, jest to dodatkowy atut zabezpieczający dane.

Nowe elementy zostają dodawane na miejsce pierwsze w każdej tablicy asocjacyjnej, jednak każde słowo musi zostać uprzednio zakodowane, co wiąże się z wydłużeniem czasu działania.
Usuwanie elementów następuje zazwyczaj bardzo szybko i przy sprzyjających warunkach(element jest na miejscu pierwszym) złożoność obliczeniowa wynosi O(1), a w podejściu pesymistycznych(funkcja musi przejść przez całą listę) złożoność wynosi O(n).