

# PAMSI - Sprawozdanie 1

Filip Guzy 218672

6 marca 2016

W celu sprawdzenia efektywności różnych metod powiększania tablic dynamicznych podczas ich przepełniania zaimplementowano trzy różne algorytmy wykonujące tę czynność. Zastosowano do nich odpowiednie oznaczenia (gdzie  $k$  to rozmiar tablicy):

1.  $k = k + 1$  - algorytm powiększający rozmiar tablicy dynamicznej o jeden w przypadku każdego przepełnienia, zapisany w pliku o nazwie `tablice_alg_1.cpp`.
2.  $k = 2 * k$  - algorytm podwajający rozmiar tablicy dynamicznej w przypadku każdego przepełnienia, zapisany w pliku o nazwie `tablice_alg_2.cpp`.
3.  $k = k + 100$  - algorytm powiększający rozmiar tablicy dynamicznej o 100 w przypadku każdego przepełnienia, zapisany w pliku o nazwie `tablice_alg_3.cpp`.

Następnie testowano ich efektywność dla różnych ilości danych, odpowiednio: 10, 100, 1000, 1000000, 1000000000. Po uzyskaniu wyników postanowiono sprawdzić ich efektywność również dla liczb 10000 oraz 100000, w celu lepszego ukazania rozbieżności między nimi. Na wejście programów podano ilość danych do wczytania, na wyjściu otrzymano czas (w milisekundach) zapełnienia tablicy odpowiednią ilością zer. Wyniki przedstawiono w poniższej tabeli:

Tabela 1:

Nazwa algorytmu	10	$10^2$	$10^3$	$10^4$	$10^5$	$10^6$	$10^9$
$k = k + 1$	0.00199997	0.0569999	3.737	283.313	28722.3	-	-
$k = 2 * k$	0.00100005	0.0079999	0.027	0.173	2.374	34.052	26231.0
$k = k + 100$	0.00100005	0.00700009	0.0549999	3.010	333.233	32496.9	-

Jak można zauważyć, z wpisaniem miliarda liczb do tablicy w rozsądnym czasie poradził sobie jedynie algorytm  $n = 2 * n$ . Jest to zatem najefektywniejszy z trzech algorytmów. Gorsze wyniki uzyskał algorytm  $n = n + 100$ , który czas alokacji miliona liczb w tablicy miał porównywalny z czasem alokacji miliarda liczb przez algorytm  $k = 2 * k$ . Najmniej efektywnie prezentuje się algorytm  $k = k + 1$ , którego czas alokacji stu tysięcy liczb był porównywalny z czasem alokacji miliona liczb przez algorytm  $k = k + 100$  i miliarda przez algorytm  $k = 2 * n$ . Różnice pomiędzy efektywnością zapełniania tablicy i realokacji jej pamięci zaczynają być widoczne już na poziomie tysiąca wczytanych liczb, gdzie różnica w czasie pomiędzy  $k = k + 1$  a dwoma pozostałymi jest na poziomie dwóch rzędów. Różnica pomiędzy  $k = k + 100$  a  $k = 2 * k$  uwidoczniła się podczas wczytania dziesięciu tysięcy liczb, gdzie czas wykonania  $k = 2 * k$  był mniejszy o jeden rząd wielkości. Różnice te są efektem częstości realokacji pamięci dynamicznej dla tablicy podczas jej przepełniania się. W przypadku  $k = k + 1$  po osiągnięciu pierwszego przepełnienia pamięć realokowana jest za każdym powtórzeniem operacji wpisanie elementu. Kolejny, efektywniejszy  $k = k + 100$  ma stały odstęp wykonywania realokacji po pierwszym przepełnieniu, który wynosi 100 wpisanych liczb, czyli realokacja całej pamięci w tablicy jest wykonywana 100 razy rzadziej niż w algorytmie  $k = k + 1$ . Najefektywniejszy z nich,  $k = 2 * n$ , z czasem wykonuje coraz mniej realokacji, co przy wzroście ilości danych jedynie działa na jego korzyść, powodując co raz to mniejszą ilość realokacji pamięci.