Wpływ nanorurek węglowych na tarcie na styku metal/metal: symulacje metodą dynamiki molekularnej

Maciej Pestka 170088,
Damian Szopinski 185394

18 marca 2022

1 Wymagania podstawowe

Wymagania od 1 od 4 są wszystkie wypełnione. Oraz wszystkie metody zostały zrealizowanie. Wszystkie deklaracje metod klasy DynamicaArray znajdują się w pliku nagłówkowym ”.h”. A ciała metod w pliku DynamicArray.cpp. Oraz są umieszczone w przestrzeni nazw AiSD

1. bool IsEmpty() jest zrealizowania według instrukcji do projektu. Czas wykonania jest stały O(1). Zwraca true kiedy tablica jest pusta oraz false w przeciwnym razie. W ciele metody znajduję się warunek, który sprawdza czy size równa się zero. Jeśli warunek jest spełniony to metoda zwraca true, czyli funkcja jest pusta. W przeciwnym razie false oddala. Definicja znajduję się w pliku ”DynamicArray.h” w linii 52 a ciało w ”DynamicArray.cpp” od 91 do 97 linii.

2. bool IsFull(). Także jest zrealizowania w pełny i wykonuję się w czasie o(1). Działa podobnie jak metoda IsEmpty(), tylko z różnicą. Zwraca true jeśli size równa się capacity, jest wykonania przez warunek. W przeciwnym razie zwraca false. Definicja znajduję się w pliku ”DynamicArray.h” w linii 57 a ciało w ”DynamicArray.cpp” od 98 do 104 linii.

3. size_t Space(). Wykonuje się w czasie O(1). Zwraca ilość wolnego miejsca tablicy, jest wykonany poprzez operacje odejmowania capacity od size. Definicja znajduję się w pliku ”DynamicArray.h” w linii 62 a ciało w ”DynamicArray.cpp” od 105 do 107 linii.

4. operator [] został przeciążony i działa w czasie O(1). Działa poprawnie i wykonuję podobnie operacja jak zwykły operator tablicy. Definicja znajduję się w pliku ”DynamicArray.h” w linii 98 a ciało w ”DynamicArray.cpp” od 285 do 287 linii.

5. void PushBack (T t). Metoda wykonuje się w czasie stałym O(1). Metoda przyjmuję argument t. Metoda dodaje na końcu wartość t oraz zwiększa size o jeden. Definicja znajduję się w pliku ”DynamicArray.h” w linii 68 a ciało w ”DynamicArray.cpp” od 108 do 115 linii.

6. void PopBack(). Metoda wykonuje się w czasie O(1). Prawidłowo usuwa wartość na końcu tabelki. Ostatnią wartość tabeli metoda zastępują wartością 0 oraz zmniejsza size o jeden. Definicja znajduję się w pliku ”DynamicArray.h” w linii 73 a ciało w ”DynamicArray.cpp” od 116 do 121 linii.

7. void PushFront(T t). Metoda działa podobnie jak metoda PushBack(T t). Tylko z różnica ze dodaje element t na początku tabelki. Na początku funkcja zwiększa size o jeden i przesuwa wszystkie elementy w tablicy o jeden. Na końcu działania metody wartość, której metoda przyjmuję w argumencie zostaje przypisywania na początku tablicy. Metoda wykonuje się w czasie O(size). Definicja znajduję się w pliku ”DynamicArray.h” w linii 79 a ciało w ”DynamicArray.cpp” od 122 do 135 linii.

8. void PopFront(). Metoda jest wykonywania w czasie O(size). Metoda działa poprzez przesuwania wszystkich wartości o jeden do przodu. Następne ostatni element jest zerowany i rozmiar size zmniejszany jest o jeden. Definicja znajduję się w pliku ”DynamicArray.h” w linii 84 a ciało w ”DynamicArray.cpp” od 136 do 144 linii.

9. void Insert(T t, size_t i). Metoda wykonuje się w czasie O(size). Metoda przyjmuję dwa argumenty, pierwszy to jest wartość do wstawienie do tablicy, a drugi argument pozycja, gdzie ma wstawić element t. Na początku metody rozmiar size jest zwiększany o jeden. Następnie wartości od końca do pozycji i są przesuwanie o jeden dalej. A na końcu wstawia wartość t na pozycji t. Definicja znajduję się w pliku ”DynamicArray.h” w linii 89 a ciało w ”DynamicArray.cpp” od 146 do 163 linii.

10. void Erase(size_t i). Metoda wykonuje się w czasie O(size) i przyjmuję jeden argument. Metoda usuwa wartość na i-tej pozycji. Metoda przesuwa wszystkie elementy od i-tej pozycji do elementy tablicy przesuwa o jeden do przodu. Oraz ostatni element niepotrzebny zeruje i size jest zmniejszany o jeden. Definicja znajduję się w pliku ”DynamicArray.h” w linii 94 a ciało w ”DynamicArray.cpp” od 165 do 182 linii.

Jest dostępna metoda wyświetlania zawartości tablicy na ekranie. Metoda jest nazwana Print(). Na początku wyświetla ilość wolnego miejsca tablicy. Przez pętle wykonuje pokazują się wartości tablicy. Definicja znajduję się w pliku ”DynamicArray.h” w linii 43 a ciało w ”DynamicArray.cpp” od 68 do 77 linii.

2 Metody dodatkowe

Udało się zrealizować powiększanie tablicy w przypadku przepełnienia. Kosztem czasu operacji dodawania elementów do tablicy. Gdy tablica jest przepełnienie to metody ”PushBack”, ”PushFront” oraz ”Insert” działają dwukrotnie wolnej. Czas ten jest poświęcony na operacje powiększanie operacji. Powiększenie tablicy ma warunek, który sprawdza czy może jeszcze zwiększyć tablicę, jeśli już nie tablica ma maksymalny rozmiar to wyświetla komunikant, że już nie może zwiększyć tablicy. Jeśli nie ma warunku to wywala błąd z powodu, że rozmiar został zbyt mocno powiększony i wyszedł poza zakres.
Wszystkie metody, które dodaje elementy do tablicy sprawdzają czy tablica jest pełna, jeśli tak to zwiększają rozmiar tablicy. A we wszystkich metody, które usuwają elementy jest dodawany warunek, który sprawdza czy tablica jest pusta. By nie usunąć elementy z pustej tablicy. Więc każda metoda sprawdza zakres indeksów oraz dostępność miejsca. Także wszystkie operacja gwarantują brak wycieków pamięci. (2 i 3 punkt w miarę możliwości został wykonany)
Nie udało się zrealizować 4 do końca. Struktura została utworzona.
Punkt 5 cały został zrealizowany. Udało się tworzyć konstruktory klasy i działają prawidłowo. Podpunkt a tworzy pustą strukturę. Capacity przyjmuję wartość z argumentu. Size jest przypisania wartość zero. Podpunkt b także prawidłowo działa. Przyjmuję trzy argumenty. Pierwszy argument określa rozmiar capacity, jaką tablica ma mieć. Drugi argument przypisuje wartość do size. A ostatni argument przyjmuję adres zmiennej t. Inicjuje tablicę N kopi elementu t. Definicja metod znajduję się w pliku .h od 27 do 28. A ciało w pliku .cpp od 13 do 27.
Punkt 6 został wykonany w pełni.
Podpunkt a, metoda at(size_t i). Działa w czasie O(1) i daję dostęp do elemtnu i-tym indeksu. Definicja znajduję się w pliku .h w 137, a ciało w pliku .cpp w od 327 do 329.
Podpunkt b metoda Clear() wykonuje się w czasie O(size). Usuwa wszystkie zapisanie wartości tabelki oraz ustawa size na zero. Definicja znajduję się w pliku .h w 104, a ciało w pliku .cpp w od 201 do 208.
Podpunkt c, metoda size_t Search(const& t). Szuka indeks elementu t w czasie O(size). Gdy znalazł indeks to kończy działanie i zwraca wartość indeksu. Definicja znajduję się w pliku .h w 109, a ciało w pliku .cpp w od 327 do 329.
Punkt 7 jest w pełni wykonani. Funkcja bool EraseFirst(const T& t). Usuwa pierwszy element równy t. Metoda wykonuje się w czasie O(size). Na początku w pętli funkcja biegnie od początku do końca tablicy. Jeśli w trakcie znajdzie element t. To przekazuję pozycje na którym znalazł element metodzie Erase(size_t i). Gdy znalazł się element i usunął go to kończy działanie, i zwraca true, jeśli wykonał operacje.
Podpunkt b także jest wykonany w O(size). Metoda szuka po wszystkich wartościach tabeli, jak znajdzie to usuwa ten element i biegnie dalej. Także podczas jednego przebiegu funkcji ustawia

indeksy tablicy. Zwraca liczbę, ile metoda usunęła elementów.
Podpunkt c także działa dobrze. Metoda zaczyna działanie od from i usuwa elementy i ustawia poprawnie indeksy. Chyba działa w czasie O(size+(last-first)). Definicja metod z punktu 7 znajdują się od 114 do 124, a ciało w pliku .cpp w liniach od 209 do 284 Punkt 8 jest wykonany. Metoda nazywa się Save(). Odczytuję każdą linię pliku i przypisuje wartości tablic według pliku. Oraz rozmiar size ustawi na prawidłową wartość. Dla tej metody metoda zapisują do pliku została zmodyfikowana. Która teraz zapisuje tylko wartości tabeli i na podstawie ich ustawia wartości tablicy i size. Definicja znajduję się w pliku ”DynamicArray.h” w linii 128 a ciało w ”DynamicArray.cpp” od 288 do 307 linii.
Punkt 9 (*) jest wykonany. Metoda wygląda Przyjmuję ”void Insert(T t, size_t iloscElementow, size_t i)” Pierwszy argument określa jaki element ma być wstawiony do tablicy. Natomiast drugi element określa ilość kopii elementu t. Wszystkie kopie t zostaną dodanie na i-tej pozycji, który określa trzeci argument. Definicja znajduję się w pliku ”DynamicArray.h” w linii 142 a ciało w ”DynamicArray.cpp” od 330 do 344 linii.
Punkt 10 (*) został także w pełny wykonany. Wszystkie konstruktory zostały utworzone. Deklaracja znajdują się pomiędzy konstruktorem ”DynamicArray(size_t capacity, size_t N,const T& t)” a destuktorem. Definicje znajduję się w pliku ”DynamicArray.h” w linii 32 do 35 a ciało w ”DynamicArray.cpp” od 28 do 61 linii.

3 Funkcje Frends w klasie Dynamic Array

- auto DoFunction (DynamicArray &, int, T, size_t, size_t) Ta funkcja wykonuje funkcje Dynamic Array, gdzie NO to numer funkcji od 0. Zapisuje wszystkie operacje do pliku (LogFileName ”Log.txt”) na wypadek crashu. Mierzy także czas wykonywania operacji w mikrosekundach. Zwraca variant<bool,size_t,noneV (nothing)> (jako to co pierwotna funkcja Dynamic Array nam zwracala).
- void OverflowTable (DynamicArray &) Wykonuje te same funkcje Dynamic Array do pusta i pełna. Pesymistyczny scenariusz jest taki, ze przyjmujemy zawsze te największe liczby jakie mogę wprowadzić. Mierzy czas dla funkcji powtarzanej tyle razy jaka jest wielkość tablicy. Przy okazji testuje takie sytuacje kiedy dodajemy kolejny element do pełnej tablicy oraz usuwamy element z pustej tablicy. Robi to poza mierzeniem czasu. Zmierzony czas jest w mikrosekundach i wyświetla się na ekranie.

4 Funkcje testujące

- void AiSD::Log(std::string src, std::string in). Ta funkcja zapisuje do pliku o ścieżce src zawartość in.
- auto AiSD::DoFunction (DynamicArray &arr, int NO, T t, size_t i). Ta funkcja wykonuje funkcje Dynamic Array, gdzie NO to numer funkcji od 0. Zapisuje wszystkie operacje do pliku (LogFileName ”Log.txt”) na wypadek crashu. Mierzy także czas wykonywania operacji w mikrosekundach. Zwraca variant<bool,size_t,noneV (nothing)> (jako to co pierwotna funkcja Dynamic Array nam zwracała).
- auto AiSD::FunctionByNO (int NO, size_t cap). Zwraca makro funkcji z Dynamic Array. A dokładniej std::function<variant<bool, size_t,noneV>(AiSD::DynamicArray& a,T t1,size_t i1)>.

- void AiSD::DistortionsSimulation (DynamicArray &arr, int t). Symulacja zakłóceń funkcji. Wykonuje losowe funkcje dla losowych argumentów. Mierzy czas. Zapisuje dane do pliku log, na wypadek crashu.
- void AiSD::OverflowTable (DynamicArray &arr). Wykonuje te same funkcje Dynamic Array do pusta i pełna. Pesymistyczny scenariusz jest taki, ze przyjmujemy zawsze te największe liczby jakie mogę wprowadzić. Mierzy czas dla funkcji powtarzanej tyle razy jaka jest wielkość tablicy. Przy okazji testuje takie sytuacje kiedy dodajemy kolejny element do pełnej tablicy oraz usuwamy element z pustej tablicy. Robi to poza mierzeniem czasu. Zmierzony czas jest w mikrosekundach i wyświetla się na ekranie.
- void AiSD::Presentation (DynamicArray &arr). NIESKOŃCZONA PĘTLA!!! Korzystanie z klasy Dynamic Array z poziomu wiersza poleceń. (Ręczne testowanie). Oprócz informacji na konsoli każde zapisuje informacje w pliku ”log.txt”.
- auto AiSD::_setNow (). Mała funkcja do otrzymania bieżącego czasu.
- std::string AiSD::_timeTook (auto a, auto b) Zwraca długość czasu pomiędzy dwoma chrono podany w string w mikrosekundach.
- void AiSD::ClearLogTxt (). Opróżnia plik Log.