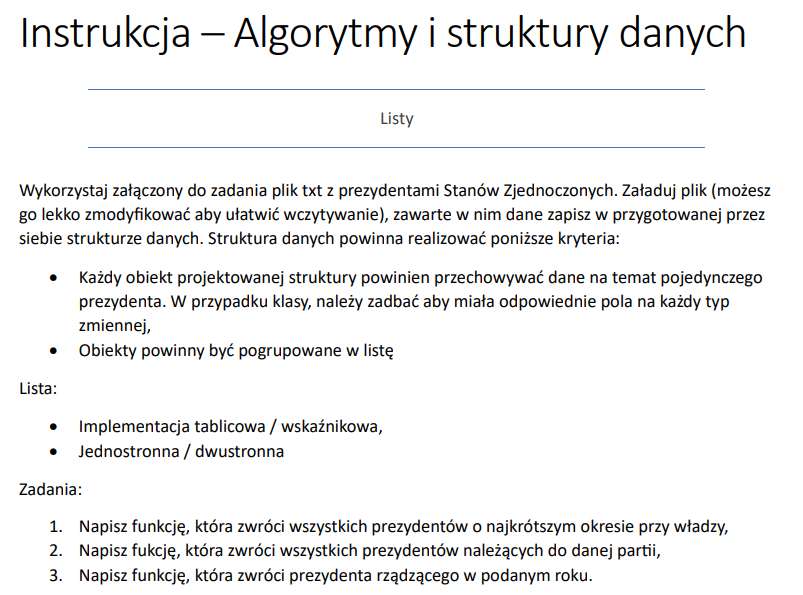
# Sprawozdanie, Zajęcia nr 1 – Listy, stosy i kolejki.

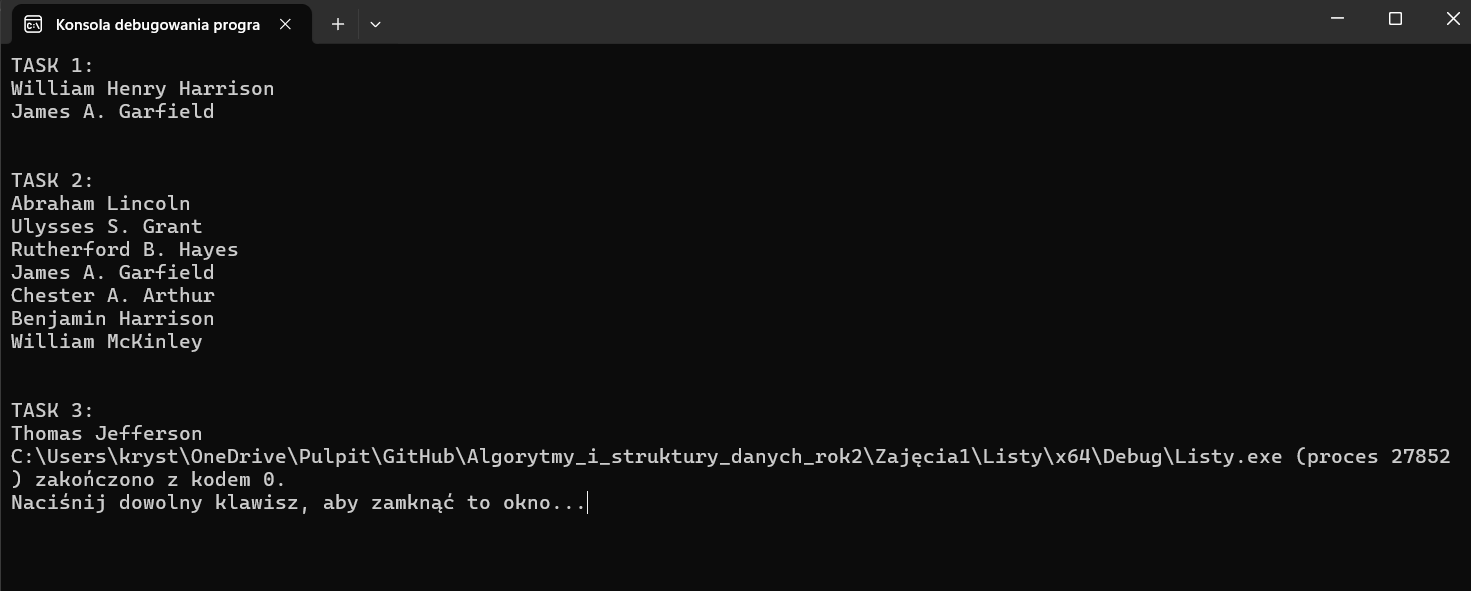
## Krystian Kostrzewa, 418845. WIMiIP, Informatyka Techniczna, sn. Data zajęć: 23.03.2024

1. **Listy**

**Polecenia:**

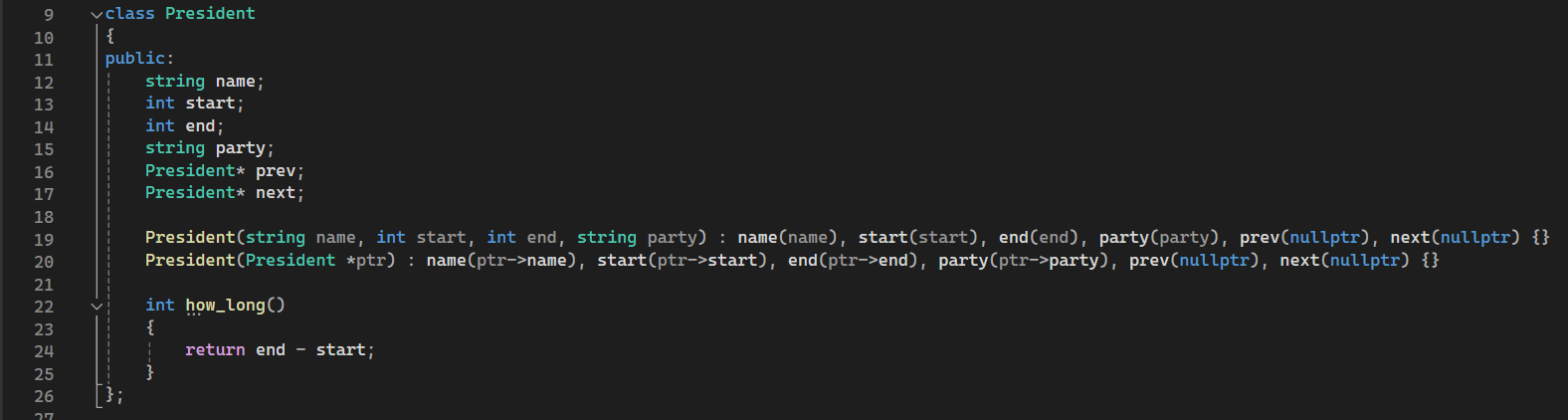


**Wynik w konsoli:**

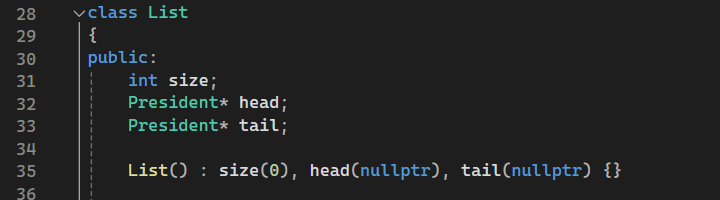


**Kod i opis rozwiązania:**

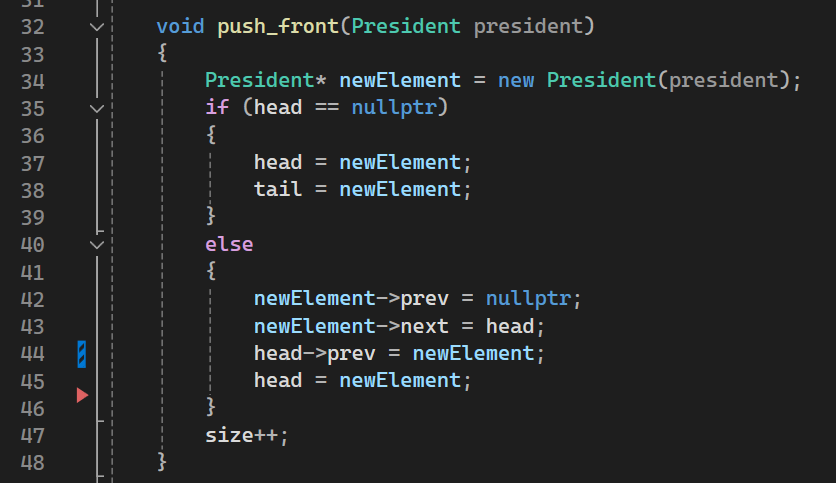
Do mojego kodu wybrałem implementację wskaźnikową dwustronną. Po skończonych pracach nad kodem uważam, że do tego zadania implementacja wskaźnikowa byłaby równie dobra, a może nawet lepsza. Zawsze to parę linii kodu mniej.



Na powyższym zrzucie ekranu przedstawiona jest klasa *President*, której obiekty będą mieć za zadanie przechowywać dane o jednym, konkretnym prezydencie USA takie jak imię i nazwisko, data rozpoczęcia rządzenia, data zakończenia rządzenia oraz partia, którą reprezentował rządząc. Dodatkowo obiekt tej klasy będą przechowywać wskaźniki na poprzedni i następny element listy, dzięki czemu będzie można za pomocą operacji na wskaźnikach przemieszczać się po strukturze danych. Klasa ta posiada dwa konstruktory. Pierwszy posłuży przede wszystkim do importu danych z pliku. Drugi konstruktor będzie używany częściej, głównie w metodach, w których operuję na wskaźnikach.

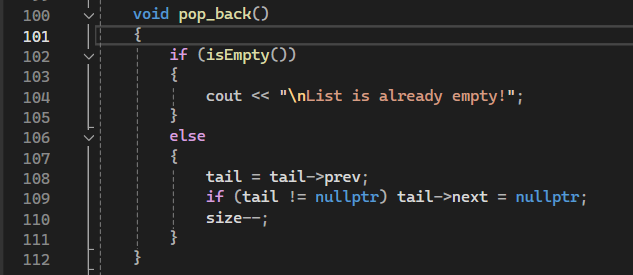
**

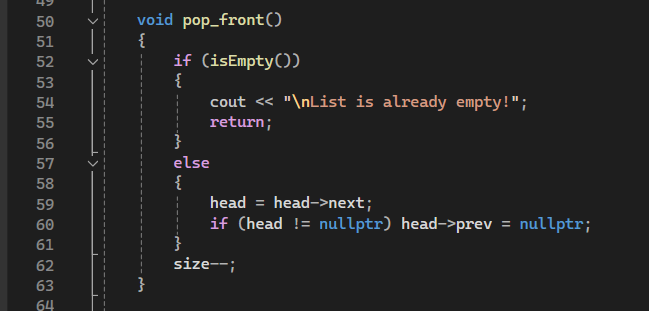
Elementy klasy *List* będą pozwalały zarządzać strukturami danych przechowującymi elementy w taki sposób, jak jest to przewidziane dla listy, czyli dostęp mamy jedynie do pierwszego i ostatniego elementu listy. Dlatego wskaźniki *head*  i *tail* wskazują na kolejno pierwszy i ostatni element struktury. Klasa posiada 1 konstruktor, który tworzy pustą listę. Uzupełniamy ją za pomocą metod.



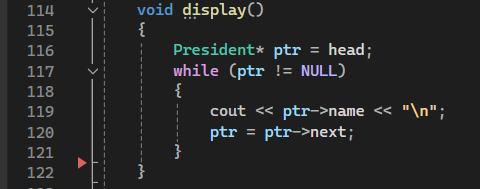


Metody dodające elementy do struktury to *push\_front* i *push\_back* przeciążona, aby można było bez problemu dodawać elementy na końcu listy za pomocą wskaźnika. Na przykładzie funkcji *push\_front* opiszę działanie tych wszystkich metod, gdyż działają analogicznie. Za pomocą konstruktora kopiującego tworzona jest dynamicznie alokowana zmienna *newElement*. Następnie program sprawdza, czy struktura klasy danych jest pusta. Jeśli tak, ustawia element zarówno jako pierwszy i ostatni element listy. Jeśli na liście coś już się znajduje manipulacja wskaźnikami, umieszczając element na 1 miejscu w liście (wskaźnik wskazujący na poprzedni element wskazuje na *nullptr* , a wskazujący na kolejny wskazuje element, który przed wywołaniem metody był headem, już nim nie jest, bo w linii 44 wskaźnik *prev* jest skierowany na inny element (już nie *nullptr*), a w 45 linii już oficjalnie headem zostaje nazwany świeżo dodany element.

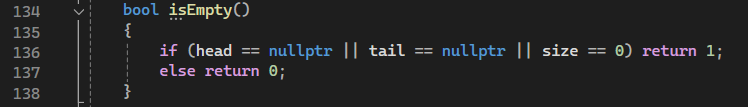


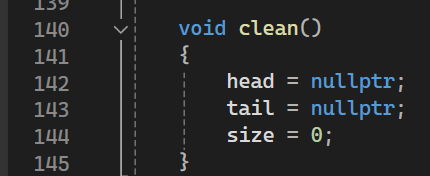


Kolejnymi metodami są *pop\_back* i *pop\_front*, które służą do usuwania kolejno ostatniego lub pierwszego elementu listy. Na przykładzie *pop\_front* opiszę działanie tych metod. Funkcja w pierwszej kolejności sprawdza, czy może lista nie jest pusta, jeśli tak nie robi nic, tylko zwraca odpowiedni komunikat i opuszcza ją. Jeśli znajdują się w niej jakieś elementy funkcja przekierowuje wskaźnik *head* na kolejny element listy i ustawia jego *prev* na *nullptr,* przez co nie ma połączenia z żadnym elementem przed nim.

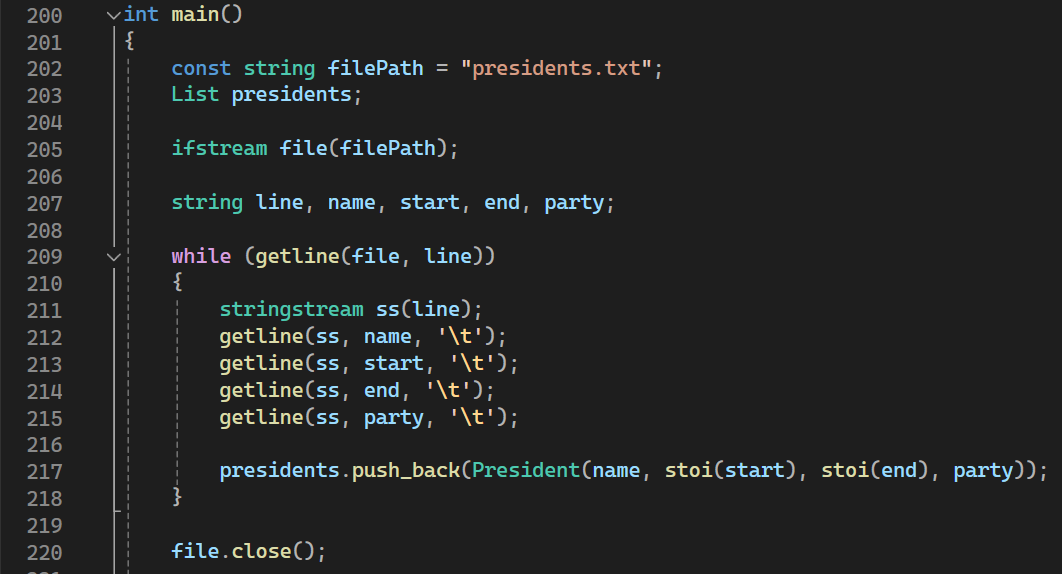


Metoda *display* wyświetla imię i nazwisko każdego prezydenta znajdującego się na liście. Zaczynając od głowy funkcja wypisuje *name* elementu, na który aktualnie skierowany jest wskaźnik *ptr* i przechodzi do następnego elementu na liście aż napotka nullptr.



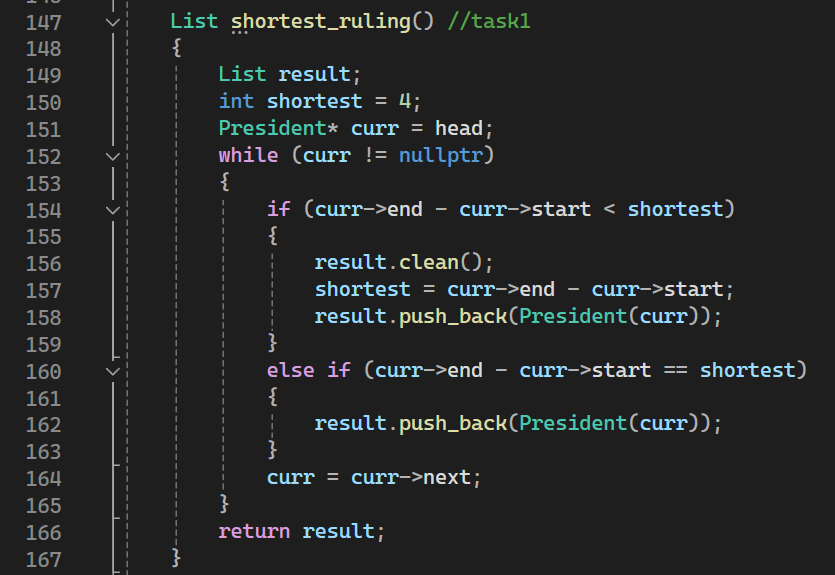


Metoda *isEmpty* po prostu sprawdza, czy w jakikolwiek sposób struktura danych wskazuje na to, że jest pusta. Jeśli tak - zwraca 1 (true), jeśli jednak nie – zwraca 0 (false). Natomiast *clean* służy do opróżniania zawartości listy poprzez ustawienie własności struktury danych na początkowe.

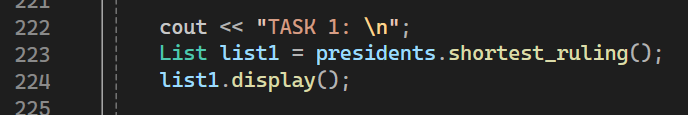


Powyższa część kodu służy do odczytu danych z pliku, zaimplementowanie ich w odpowiedni sposób do elementów klasy *President* oraz połączenia całości w liście klasy List. W skrócie działa to tak, że z całego pliku, linia po linii sczytywany jest tekst, zapisywany do zmiennej *ss*, następnie wszystkie dane oddzielone znakiem tabulacji przypisywane są do odpowiednich zmiennych, a potem za pomocą konstruktora klasy *President* tworzone są elementy tej klasy i przy użyciu metody klasy *List* dodawane są do listy *presidents*.

Rozwiązanie zadania 1:

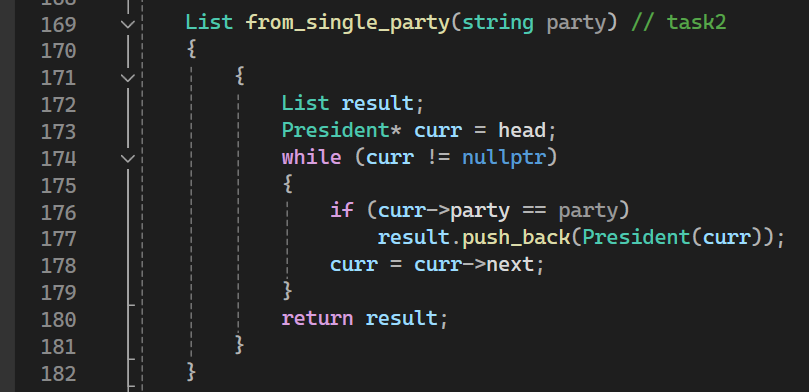


Do wykonania 1 polecenia przygotowałem metodę klasy *List*, która za pomocą wskaźnika *curr* (który po każdym wykonaniu pętli while wskazuje na kolejny element listy, aż napotka *nullptr*) sprawdza, czy aktualnie rozpatrywany prezydent panował najkrócej spośród wcześniej sprawdzanych. Jeśli tak, czyści listę z poprzednich i dopisuje element do listy *result*. Jeśli jednak rządził równie długo – nie usuwa pozostałych prezydentów będących już na liście, tylko dopisuje go, jako kolejny element. Na końcu zwracana jest tablica, która przechowuje wszystkich najkrócej rządzących prezydentów.

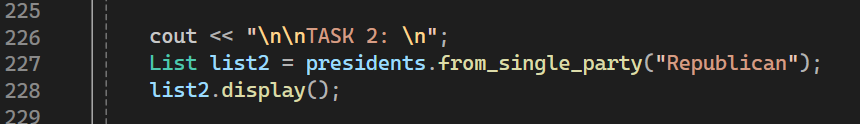


Powyżej wywołanie metody na liście przechowującej wszystkich prezydentów z pliku oraz wyświetlenie jej za pomocą metody *display* w funkcji *main*.

Rozwiązanie zadania 2:

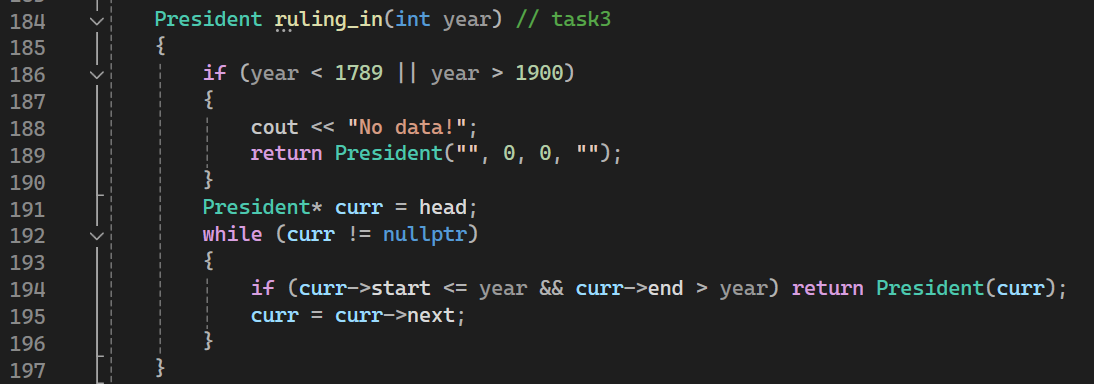


Do wykonania 2 polecenia przygotowałem metodę klasy *List*, która za pomocą wskaźnika *curr* (który po każdym wykonaniu pętli while wskazuje na kolejny element listy, aż napotka *nullptr*) sprawdza, czy aktualnie rozpatrywany prezydent należał do partii, której nazwę przekazaliśmy w argumencie, przy wywołaniu metody. Jeśli tak - dopisuje element do listy *result*. Na końcu zwracana jest tablica, która przechowuje wszystkich prezydentów, którzy należeli do wskazanej partii.

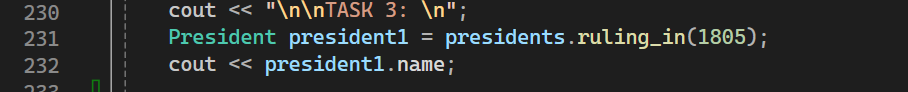


Powyżej wywołanie metody na liście przechowującej wszystkich prezydentów z pliku oraz wyświetlenie jej za pomocą metody *display* w funkcji *main*.

Rozwiązanie zadania 3:



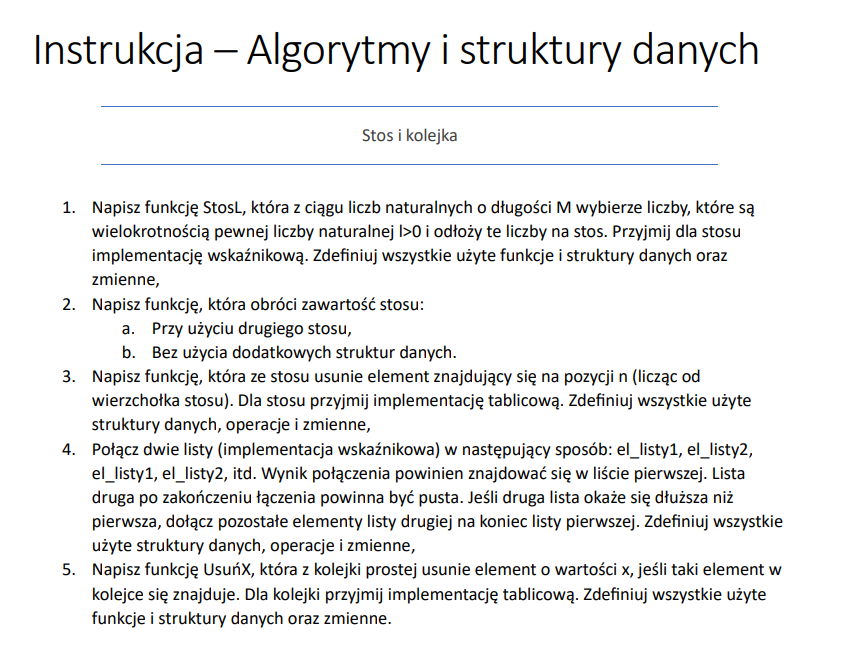
Do wykonania 2 polecenia przygotowałem metodę klasy *List*, która w pierwszej kolejności sprawdza, czy rok wpisany przez użytkownika mieści się w przedziale lat, zgodnym z danymi z pliku wejściowego, jeśli nie -zwracany jest pusty element klasy *President* oraz odpowiedni komunikat o braku danych. Jeśli jednak rok znajduje się w odpowiednim przedziale sprawdzane jest, czy zmienna *party* w tym elemencie jest równa argumentowi podanemu przy wywołaniu funkcji. Jeśli tak – element klasy President jest zwracany i następuje wyjście z funkcji.



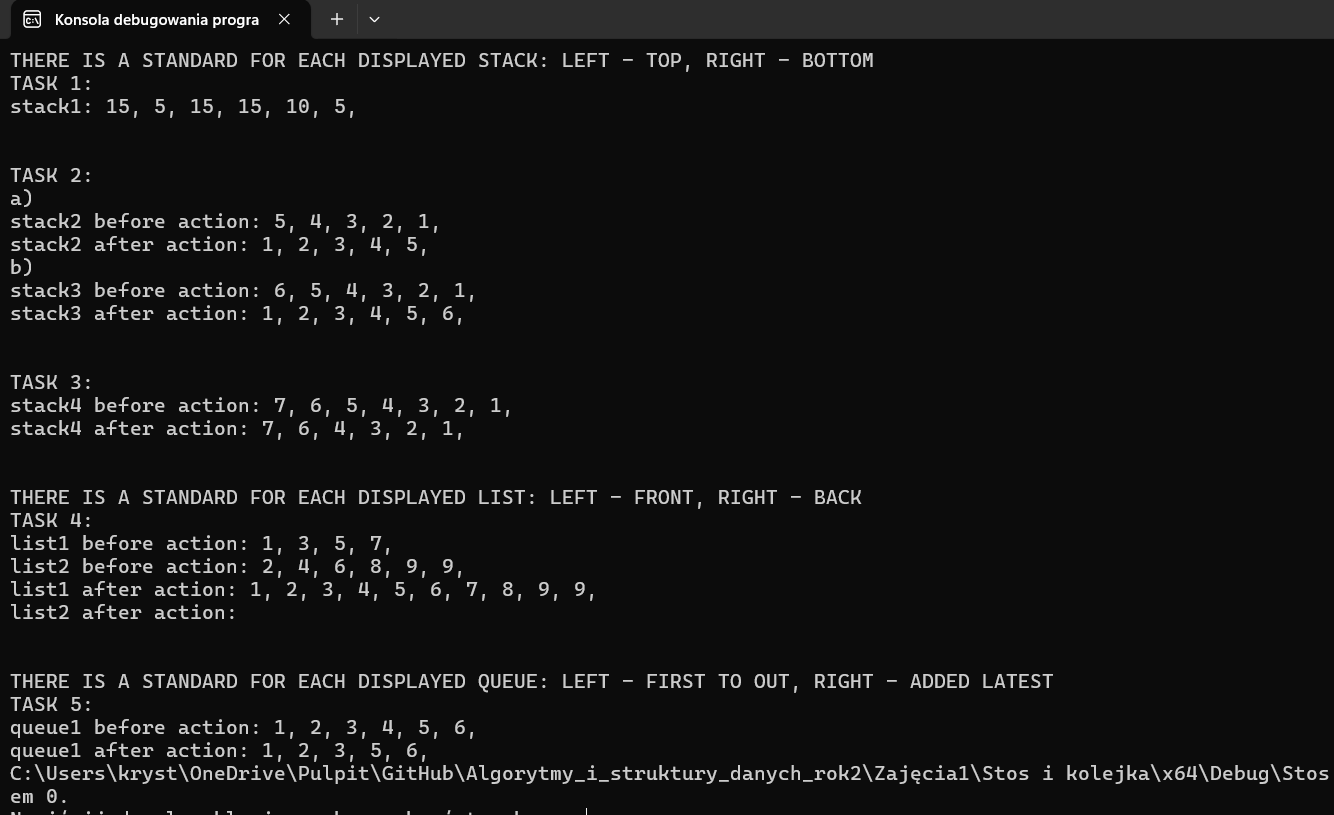
Powyżej wywołanie metody na liście przechowującej wszystkich prezydentów z pliku oraz wyświetlenie jej za pomocą metody *display* w funkcji *main*.

1. **Listy**

**Polecenia:**



**Wynik w konsoli:**



**Kod i opis rozwiązania:**

Do wykonania zadań 1 i 2 użyłem elementów klas *StackElement* oraz