



POLITECHNIKA RZESZOWSKA
im. Ignacego Łukasiewicza
WYDZIAŁ MATEMATYKI I FIZYKI STOSOWANEJ

JAKUB BARAN

Projekt C++
Zadanie 2
kierunek studiów: Inżynieria i Analiza Danych

Opiekun pracy:
Mariusz Borkowski

Rzeszów 2022

Opis Problemu

Należy napisać program szukający powtarzającego się elementu dla zadanej tablicy liczb całkowitych o rozmiarze n zawierającej wartości z przedziału $[1, n-1]$

Przykład.

Wejście: 1, 2, 3, 4, 4

Wyjście powtarzający się element to 4

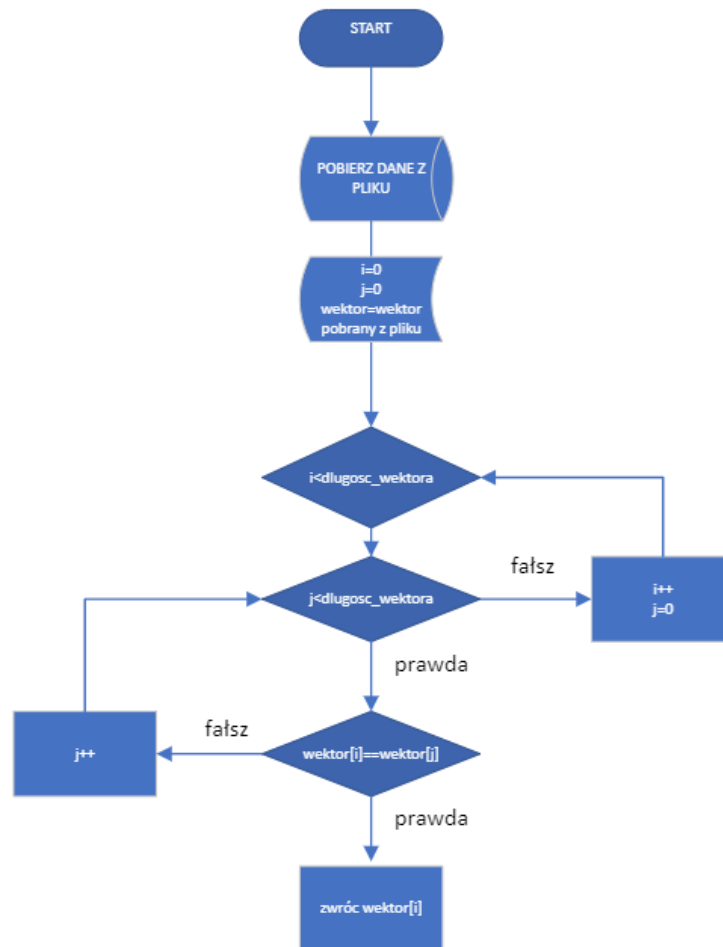
Wejście: 1, 2, 3, 4, 2

Wyjście powtarzający się element to 2

Opis szczegółów implementacji problemu

Program powinien pobierać dane z pliku dla tego lepiej zastosować wektory bo są dynamiczne – nie trzeba deklarować ilości zmiennych (nie wiemy ile użytkownik będzie chciał wpisać danych). Program Główny powinien zawierać pętle szukające powtarzającego się elementu wykonujące się tyle razy ile wynosi długość wektora z danymi. Po znalezieniu właściwego elementu program powinien zwrócić ten element do głównej funkcji abyśmy mogli go zapisać do pliku. Warto by było zmierzyć czas algorytmu. Należy również zrobić dwie funkcję - pierwszą do wczytywania danych z pliku do wektora a drugą zapisującą wyniki z programu do pliku

Schemat Blokowy Algorytmu



Pseudokod

(cały program)

```
funkcja powtarzajacy_element(wektor) {  
    dla (i = 1 do rozmiar wektora) wykonuj  
        dopóki (j <= rozmiar wektora) wykonuj  
            jeżeli (i==j) - przerwij  
            jeżeli (wektor[i]==wektor[j])  
                zwróć wektor[i]  
        }  
    }  
funkcja pobranie_danych (nazwa){  
    dane <- z pliku (nazwa)  
    jeżeli (dane nie sa otwarte)  
        wypisz blad otwarcia pliku  
    dopóki(dane sa wczytywane ) wykonuj
```

```

        dane >> tmp
        wektor.dodaj (tmp)
    dane.zamknij()
    zwróć dane
}
funkcja zapisanie_danych(nazwa, powtarzajacy_element, czas){
    Wypisz do pliku "Powtarzajacy element to: ", powtarzajacy_element
    Wypisz do pliku "Czas wykonywania programu: ", czas "sekundy"
}
funkcja główna
{
start = start_mierzenia_czasu()
dane = pobranie_danych("nazwa_pliku_z_danymi")
    powtarzajacy_element(dane)
    koniec = koniec_mierzenia_czasu()
    czas = koniec-dane
    zapisanie_danych("nazwa_pliku_do_zapisu", powtarzajacy_element(dane), czas)
    zwróć 0
}

```

(główny algorytm)

```

funkcja powtarzajacy_element(wektor) {
    dla (i = 1 do rozmiar wektora) wykonuj
    dopóki (j <= rozmiar wektora) wykonuj
        jeżeli (i==j) - przerwij
        jeżeli (wektor[i]==wektor[j])
            zwróć wektor[i]
}

```

Wnioski

Główny algorytm w złożoności kwadratowej działa poprawnie. Niestety nie dałem rady mniejszej złożoności próbowałem z sortowaniem bąbelkowym w ten sposób ale to też jest złożoność kwadratowa i oprócz funkcji sortujących potrzebna jest dodatkowa funkcja z znalezieniem powtarzającego się elementu. Kod tego rozwiązania:

```

vector<int> wektor = { 1, 2, 3, 3, 4 };
for (int i = 0; i < wektor.size() - 1; i++)
{
    for (int j = 0; j < wektor.size() - 1; j++)
    {
        if (wektor[j] > wektor[j + 1])
        {
            swap(wektor, wektor[j + 1]);
        }
    }
}

```

```
        }  
    }  
  
    for(int i=0; i<wektor.dlugosc+1; i++)  
    if wektor[i] == wektor [i+1]{ return wektor[i] }
```

Program również liczy czas trwania algorytmu i podaje go w sekundach
Zrobiłem też funkcję wczytującą dane z pliku do wektora i drugą zapisującą dane do pliku z wynikiem
głównej funkcji - powtarzającym się elemencie i z czasem trwania algorytmu. Program jest
kompletny nie mam pomysłu jak go można było zrobić lepiej