**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Algorytmy i struktury danych | **Zadanie** | 2.1 |
| **Autor** | Wojciech Matras | **Grupa** | WCY21KY1S1 |
| **Temat** | Lista jednokierunkowa \* | | |

1. Treść zadania

Zaimplementuj program realizujący w formie listy dwukierunkowej następujące funkcje wizytownika :

- dodaj wizytówkę,

- szukaj wizytówki,

- wypisz wizytówki (A... Z),

- wypisz wizytówki (Z... A),

- usuń wizytówkę,

– zapisz rekordy w pliku

- powtórz sterowanie

* 1. Metoda realizacji

Dane wejsciowe to tylko liczby wprowadzone z klawiatoru do listy dwukierunkowej. Program zwraca liste na ekranie i w pliku . txt

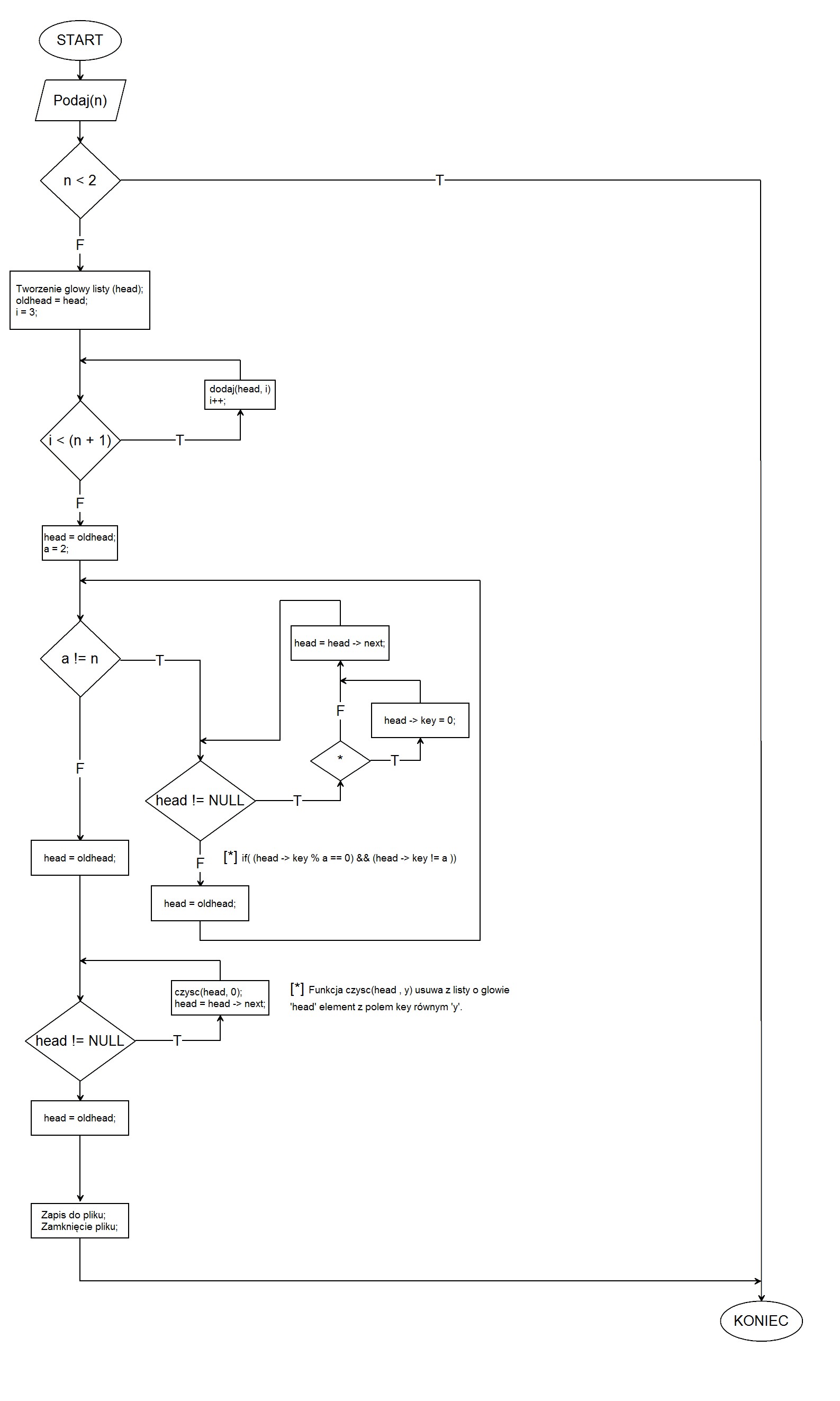
* 1. Założenia / ograniczenia dotyczące danych:
     1. Dane wejściowe:

n podawane z klawiatury

* + 1. Dane wyjściowe:

wy.TXT- plik tekstowy o nazwie wy.TXT, w którym każda liczba pierwsza zapisana jest w nowej linii.

1. Realizacja
   1. Algorytm *(lista kroków lub schemat blokowy)*



* 1. Kod źródłowy

//WCY21KY1S1 WOJCIECH MATRAS

#include<iostream>

#include<fstream> //bedziemy dzialac na plikach

using namespace std;

typedef struct Node //tworzymy liste jednokierunkowa z int'em

{

int data;

struct Node\* next;

} Node;

void sito(bool \*tab, unsigned int n)

{

for (int i=2; i\*i<=n; i++) //przeszukujemy kolejnych kandydatów na pierwsze

{ //wystarczy sprawdzić do pierwiastka z n

// i<=sqrt(n) - podnosząc do kwadratu mamy

// i\*i <= n

if(tab[i] == 0) //jesli liczba jest pierwsza(ma wartosc 0)

for (int j = i\*i ; j<=n; j+=i) //to wykreslamy jej wielokrotnosci

tab[j] = 1; //ustawiając wartosć na 1

}

}

void insert(Node\*\* pointerToHead, int x) //funkcja dodajaca kolejna liczbe pierwsza do kolejki

{ //po wykonaniu funkcji sita w main'ie dostajemy tablice n-elementowa bool, jesli w n-elemencie tab[n]=1 znaczy to

Node\* temp=(Node\*)malloc(sizeof(struct Node)); //ze w tym miejscu jest nowa liczba pierwsza

temp->data = x;

temp->next=\*pointerToHead;

if(\*pointerToHead != NULL) temp->next=\*pointerToHead;

\*pointerToHead = temp;

}

void drukuj(Node\* head) //funkcja drukujaca liste

{ int i =0;

printf("\n");

while(head != NULL) //dopoki head != NULL drukujemy wartosci listy

{

printf("%d ", head->data);

i++; // naliczamy i by program nie drukowal w jednej lini wszystkich wyrazow

if (i%7==0){ // co 7 wyraz przechodzimy na nastepna linie

printf("\n");

}

head = head->next;

}

}

void zapisdoPliku(Node \*head) { //funkcja zapisujaca liste do Pliku.txt

fstream new\_file; //tworzymy Plik

new\_file.open("new\_file\_write.txt",ios::out); //otwieramy Plik

if(!new\_file)

{

printf("\n");

cout<<"File creation failed";

}

else

{

while(head != NULL) //dopoki head != NULL wypisujemy kolejne elementy do Pliku

{

new\_file<<head->data<<" "; //wpisujemy do Pliku

head = head->next;

}

new\_file.close(); //Zamykamy nowy Plik

}

}

int main()

{

Node\* head ; //tworzymy wskaznik na glowe listy

head=NULL;

int n; //n to liczba do ktorej szukamy liczb pierwszych w zakresie od [2,n]

cout<<"Podaj zakres górny przedziału: ";

cin>>n; //wczytujemy n

bool m[n+1]; //tworzymy tablice n elementowa bool czyli z wartosciamy 0 lub 1

for(int i=2; i<=n; i++) //zerowanie tablicy

m[i] = 0;

sito(m, n); //wywolanie funkcji sita

cout<<"Kolejne liczby pierwsze z przedziału [2.."<<n<<"]: ";

for(int i=2;i<=n;i++){

if(m[i]==0){ //jesli wartosc tablicy w dany miejscu pozostal rowny 0, wpisujemy ja do Listy

insert(&head,i);

}

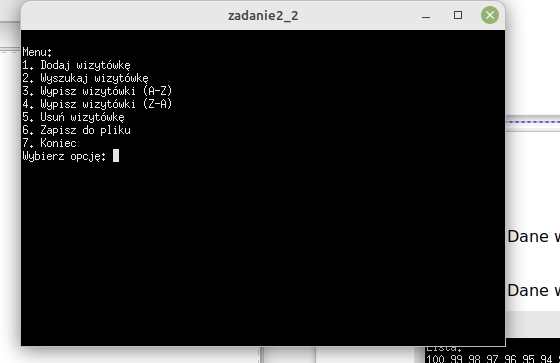
}

drukuj(head);

zapisdoPliku(head);

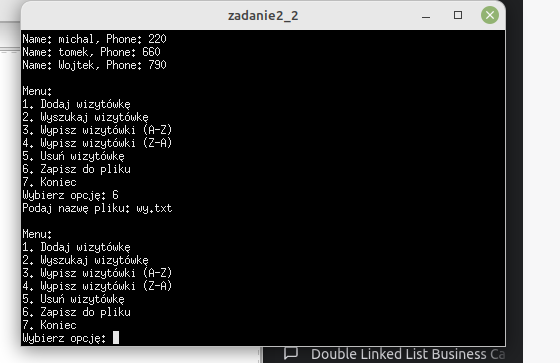
return 0;

}

* 1. Dane wejściowe *(ekran uruchomienia programu)*

Brak

* 1. Dane wyjściowe *(ekran uruchomienia programu)*



Wnioski *(złożoność asymptotyczna algorytmu przy użyciu notacji* ***O lub innej i inne wnioski)***

Złożoność obliczeniowa algorytmu

Załączony program został stworzony w celu znajdowania liczb pierwszych

w zadanym przedziale od 2 do n. Praca programu polegała na tym, by w

danym przedziale eliminować kolejno wielokrotności liczb, jednocześnie

nie usuwając liczb przez które dzieliliśmy. W ten sposób początkowo

eliminowano liczby podzielne przez 2, następnie przez 3, przez 5 i

ostatecznie otrzymywano zbiór liczb pierwszych dla danego przedziału.

Niestety nie jest to idealna realizacja algorytmu sita Eratostenesa.

Załączony program pracuje wyraźnie wolniej już dla n > 10 000.

zalezy od naszych preferencji w interfejsie jaka będzie zlozonosc jednak wszystkie funkcje poza jedna maja zlozonosc

O(n)=n

poza Wypisz wizytowki Z-A

gdzie

O(n)=2n