**SPRAWOZDANIE**

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

1. Treść zadania

Zaimplementuj algorytm realizujący kolejno: Wypełnienie listy kolejnymi wartościami 1..100 Wyświetlenie zawartości listy Obliczenie i wyświetlenie średniej z wartości umieszczonych w liście Usunięcie z listy wartości nieparzystychWyświetlenie zawartości listy Obliczenie i wyświetlenie średniej z wartości umieszczonych w liście

* 1. Metoda realizacji

Generujemy liste 100 elementowa i wyswietlamy wyniki na ekranie

* 1. Założenia / ograniczenia dotyczące danych:
     1. Dane wejściowe *(*
     2. Dane wyjściowe *(opis)*

Kolejne wyrazy ciągu każdy w nowej linii - wyprowadzane na ekran.

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

1.Wypełnienie listy kolejnymi wartościami 1..100

2.Wyświetlenie zawartości listy

3.Obliczenie i wyświetlenie średniej z wartości umieszczonych w liście 4.Usunięcie z listy wartości nieparzystych

5.Wyświetlenie zawartości listy

6.Obliczenie i wyświetlenie średniej z wartości umieszczonych w liście

* 1. Kod źródłowy

#include <stdio.h>

#include <stdlib.h>

struct Node {

int value;

struct Node \*next;

};

struct LinkedList {

struct Node \*head;

};

// Funkcja do dodawania nowego elementu na początek listy

void add\_node(struct LinkedList \*list, int value) {

// Alokuj pamięć dla nowego elementu

struct Node \*new\_node = (struct Node\*)malloc(sizeof(struct Node));

// Przypisz wartość nowemu elementowi

new\_node->value = value;

// Ustaw wskaźnik next nowego elementu na obecny head listy

new\_node->next = list->head;

// Ustaw nowy element jako head listy

list->head = new\_node;

}

// Funkcja do obliczania średniej wartości elementów listy

float average\_list(struct LinkedList \*list) {

float sum = 0;

int count = 0;

// Ustaw obecny element jako head listy

struct Node \*current = list->head;

// Iteruj przez wszystkie elementy listy

while (current != NULL) {

sum += current->value;

count++;

current = current->next;

}

return sum / count;

}

void print\_list(struct LinkedList \*list) {

struct Node \*current = list->head;

while (current != NULL) {

printf("%d ", current->value);

current = current->next;

}

}

void delete\_odd\_nodes(struct LinkedList \*list) {

struct Node \*current = list->head;

struct Node \*previous = NULL;

while (current != NULL) {

if (current->value % 2 != 0) {

if (previous != NULL) {

previous->next = current->next;

} else {

list->head = current->next;

}

struct Node \*temp = current;

current = current->next;

free(temp);

} else {

previous = current;

current = current->next;

}

}

}

int main() {

struct LinkedList list;

list.head = NULL;

int i;

for(i=1; i<101;i++)

{ add\_node(&list, i);}

printf("Lista: \n");

print\_list(&list);

printf("\nSrednia to %f \n",average\_list(&list));

delete\_odd\_nodes(&list);

printf("Lista: \n");

print\_list(&list);

printf("\nSrednia to %f \n",average\_list(&list));

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

n+n+n+1/2n=3,5n = ***O(n)***