**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Algorytmy i struktury danych | **Zadanie** | 2\_01 |
| **Autor** | Kamil Borkowski | **Grupa** | WCY22IY1S1 |
| **Temat** | Lista jednokierunkowa zaimplementowana wskaźnikowo | | |

1. Treść

Zaimplementuj algorytm realizujący kolejno:

Wypełnianie listy kolejnymi wartościami 1…100

Wyświetlanie zawartości listy

Obliczanie i wyświetlanie średniej z wartości umieszczonych w liście

Usuniecie z listy wartości nieparzystych

Wyświetlenie zawartości listy

Obliczenie i wyświetlenie średniej z wartości umieszczonych w liście

* 1. Metoda realizacji

Wypełniamy w pętli kolejno wartości od 1 do 100 i obliczamy z nich średnią, następnie usuwamy w pętli elementy o wartościach nieparzystych i liczymy średnia ponownie.

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

brak

* + 1. Dane wyjściowe *(opis)*

Wartości zgodne z opisem działania – wyświetlone na ekranie

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

Deklaracja struktury listy

Utworzenie funkcji void uzupełniającą listę poprzez tworzenie nowych wskaźników

Utworzenie funkcji void wypisującą wszystkie elementy listy

Utworzenie funkcji float obliczającą średnią z wartości elementów listy

Utworzenie funkcji void usuwającą elementy o wartościach xmod2 !=0

Utworzenie funkcji void usuwającą pierwszy element tablicy który nie został usunięty podczas działania pierwszej funkcji usuwającej elementy listy

Ponowne wywołanie funkcji wypisującej i średniej

* 1. Kod źródłowy

#include<stdio.h>

#include<stdlib.h>

//deklaracja struktury tablicy

typedef struct list{

int wartosc;

struct list\* next;

}list;

//funkcja wyswietlajaca elementy listy

void wyswietl(list\* head)

{

list\* pom = head;

if(pom == NULL)return;

while(pom != NULL)

{

printf("%d ", pom->wartosc);

pom = pom->next;

}

printf("\n");

}

//funkcja uzupelniajaca liste

void dodaj\_element(list\*\* head, int wartosc)

{

list\* tail = \*head;

while (tail->next != NULL) {

tail = tail->next;

};

list\* new = (list\*)malloc(sizeof(list));

new->wartosc = wartosc;

new->next = NULL;

tail->next = new;

}

//funkcja obliczajaca srednia z elementow tablicy

void srednia(list\* head)

{

float j = 0.0;

float wartosc = 0.0;

float srednia;

list\* pom = head;

if(pom == NULL)

return;

while(pom != NULL){

wartosc = wartosc + pom->wartosc;

j++;

pom = pom->next;

}

srednia = wartosc / j;

printf("Srednia: %f\n", srednia);

}

//funkcja usuwajaca pierwszy element listy(wartosc nieparzysta)

void usuwanie\_pierwszego\_elementu(list \*\*head)

{

list \*pom;

pom = \*head;

if(pom){

\*head = pom->next;

free(pom);

}

}

//funkcja usuwajaca element listy

void usun\_element(list\*\* head, int wartosc)

{

list\* current = \*head;

list\* poprzedni = current;

current = current->next;

while(current != NULL){

if(current->wartosc == wartosc){

poprzedni->next = current->next;

free(current);

return;

}

poprzedni = current;

current = current->next;

}

}

int main()

{

list\* lista = (list\*)malloc(sizeof(list));

lista->wartosc=1;

lista->next=NULL;

printf("Wartosci:\n");

//petla dodajaca 99 elementow do listy

int i=2;

for(;i<=100;i++)

{

dodaj\_element(&lista, i);

}

wyswietl(lista);

srednia(lista);

usuwanie\_pierwszego\_elementu(&lista);

//petla usuwajaca nieparzyste elementy (poza pierwszym, juz usunietym)

i=1;

for(;i<=100;i++)

{

if(i%2==1)usun\_element(&lista, i);

}

wyswietl(lista);

srednia(lista);

return 0;

}

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

brak

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

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

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

Złożoność obliczeniowa programu wynosi: O(n), ponieważ najbardziej złożone obliczeniowo jest wykonanie niezagnieżdżonych pętli.