**Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки**

**Лабораторна робота № 2.2**

з дисципліни  
«Алгоритми і структури даних»

Виконав Перевірила:

Студент групи ІП-03 Сергієнко А. А.  
Пашковський Євгеній Сергійович  
номер у списку групи: 17

Київ 2021

**Завдання**

1. Створити список з ***n*** (***n***>0) елементів (***n*** вводиться з клавіатури), якщо інша кількість елементів не вказана у конкретному завданні.

2. Тип ключів (інформаційних полів) задано за варіантом.

3. Значення елементів списку взяти самостійно такими, щоб можна було продемонструвати коректність роботи алгоритму програми. Введення значень елементів списку можна виконати довільним способом (випадкові числа, формування значень за формулою, введення з файлу чи з клавіатури).

4. Вид списку (черга, стек, дек, прямий однозв’язний лінійний список,

обернений однозв’язний лінійний список, двозв’язний лінійний список,

однозв’язний кільцевий список, двозв’язний кільцевий список) вибрати

самостійно з метою найбільш доцільного рішення поставленої за варіантом

задачі.

5. Виконати над створеним списком дії, вказані за варіантом, та коректне

звільнення пам’яті списку.

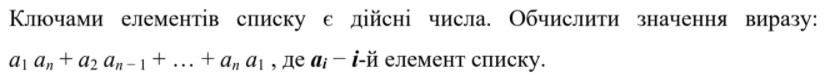
6. **При виконанні заданих дій, виводі значень елементів та звільненні**

**пам’яті списку вважати, що довжина списку (кількість елементів n чи 2n)**

***невідома* на момент виконання цих дій.**

7. Повторювані частини алгоритму необхідно оформити у вигляді процедур або функцій (для створення, обробки, виведення та звільнення пам’яті списків) з передачею списку за допомогою параметра(ів).

Варіант 17:



**Текст програми**

#include <stdio.h>

#include <stdlib.h>

typedef struct linked\_list

{

double data;

struct linked\_list \* next\_p;

struct linked\_list \* prev\_p;

} l\_list;

l\_list \* l\_list\_init(double item)

{

l\_list \*l\_p;

l\_p = malloc(sizeof(struct linked\_list));

l\_p->data = item;

l\_p->next\_p = l\_p;

l\_p->prev\_p = NULL;

return l\_p;

}

l\_list \* get\_last\_node(l\_list \*l\_p)

{

l\_list \*this\_p = l\_p;

while(this\_p->prev\_p)

{

this\_p = this\_p->prev\_p;

}

return this\_p;

}

l\_list \* add\_node(l\_list \*l\_p, double item)

{

l\_list \*node\_p;

node\_p = malloc(sizeof(struct linked\_list));

l\_p->next\_p = node\_p;

node\_p->data = item;

node\_p->next\_p = get\_last\_node(l\_p);

node\_p->prev\_p = l\_p;

return node\_p;

}

//remove node from begining

l\_list \* unshift\_node(l\_list \*l\_p)

{

l\_list \*new\_l\_p = l\_p->prev\_p;

new\_l\_p->next\_p = l\_p->next\_p;

free(l\_p);

return new\_l\_p;

}

//remove node from end

l\_list \* pop\_node(l\_list \*l\_p)

{

if (l\_p->next\_p != l\_p)

{

l\_list \*last\_node = l\_p->next\_p;

last\_node->next\_p->prev\_p = NULL;

l\_p->next\_p = last\_node->next\_p;

free(last\_node);

return l\_p;

} else {

free(l\_p);

return NULL;

}

}

int main()

{

system("cls");

unsigned int n = 0;

while(n <= 0 || n % 2 != 0)

{

printf("n (n %% 2 = 0; n > 0): ");

scanf("%u", &n);

}

l\_list \*list\_p;

for (unsigned int i = n; i >= 1; i--)

{

double a;

printf("a%u: ", i);

scanf("%lf", &a);

if (i == n)

{

//initialization

list\_p = l\_list\_init(a);

} else {

//fulfilling

list\_p = add\_node(list\_p, a);

}

}

//operations

double result = 0;

while (list\_p) {

result += 2 \* (list\_p->data) \* (list\_p->next\_p->data);

list\_p = unshift\_node(list\_p);

list\_p = pop\_node(list\_p);

}

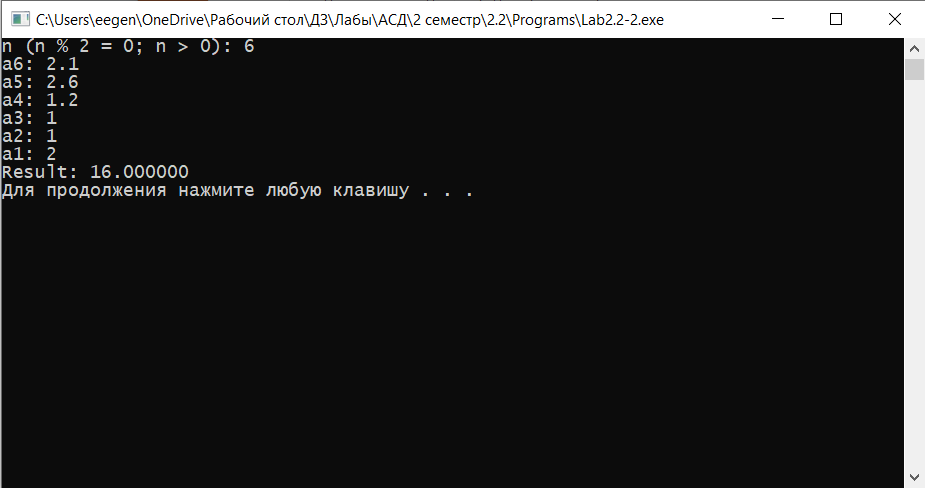
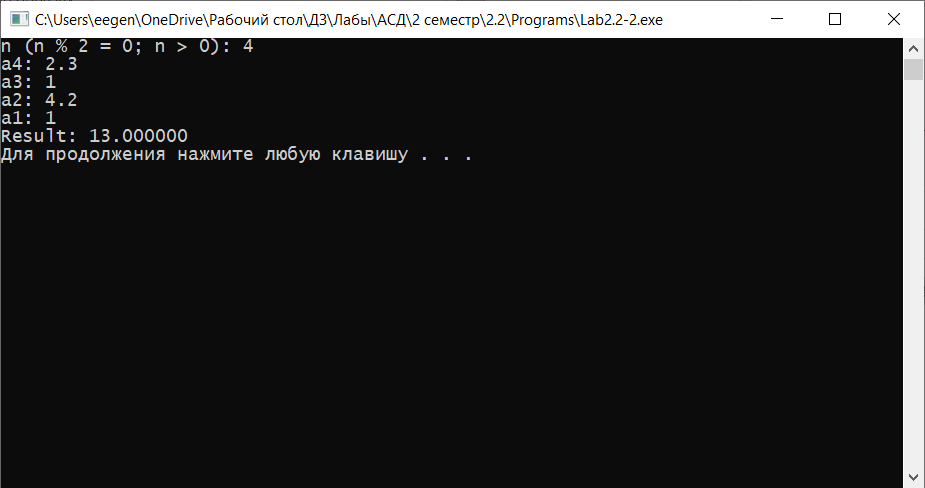
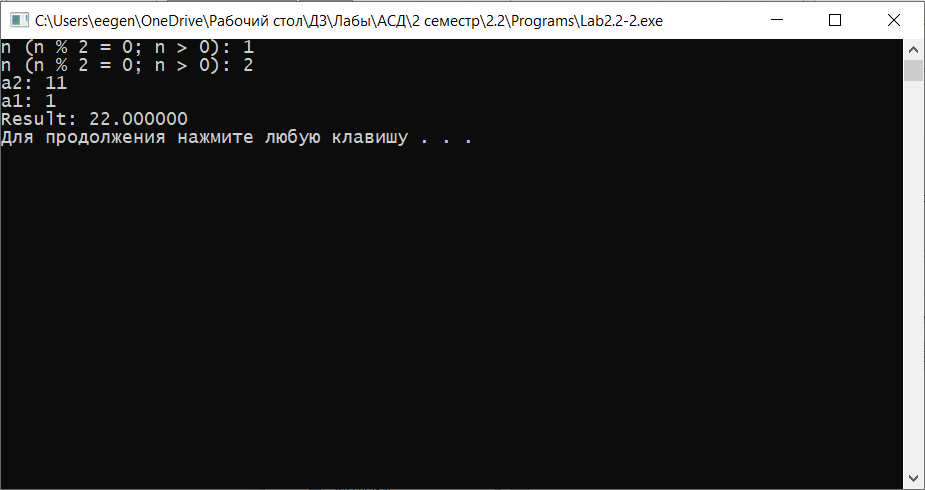
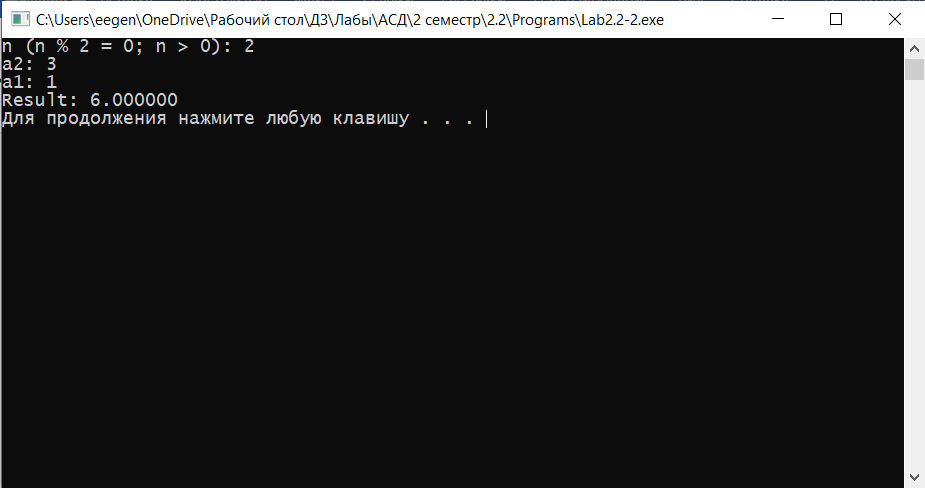
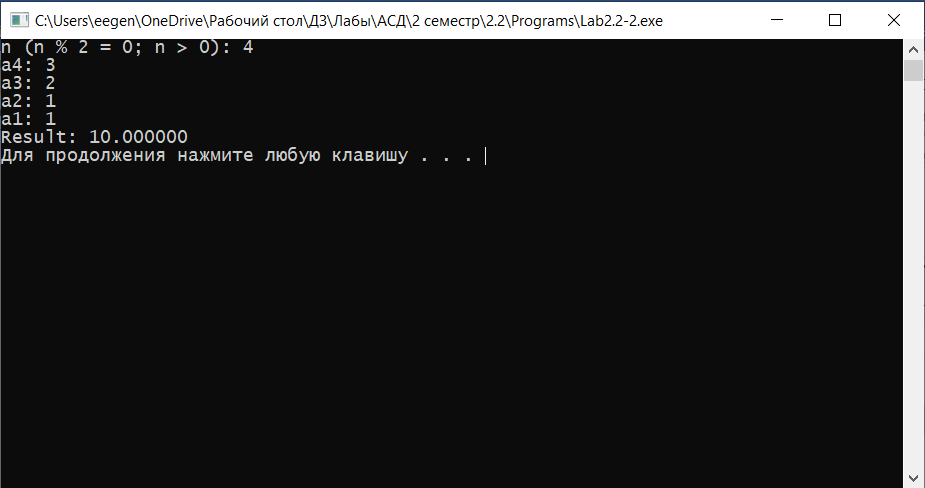
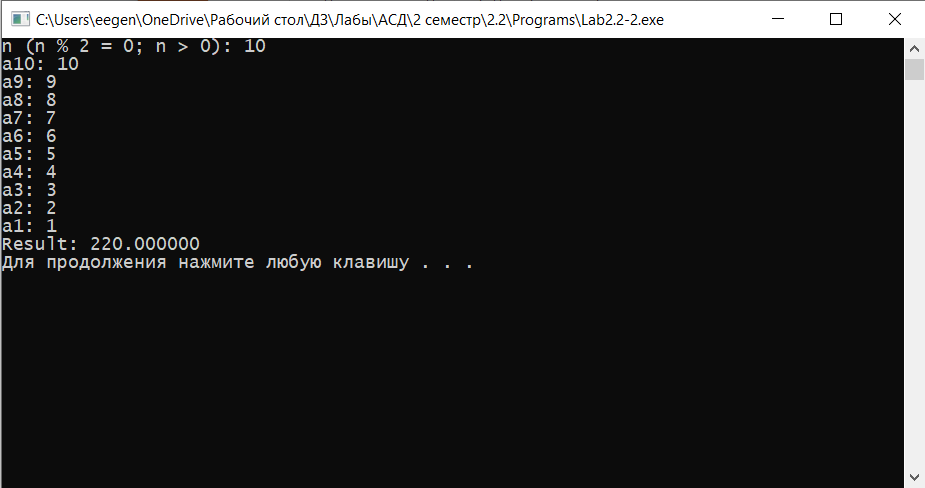
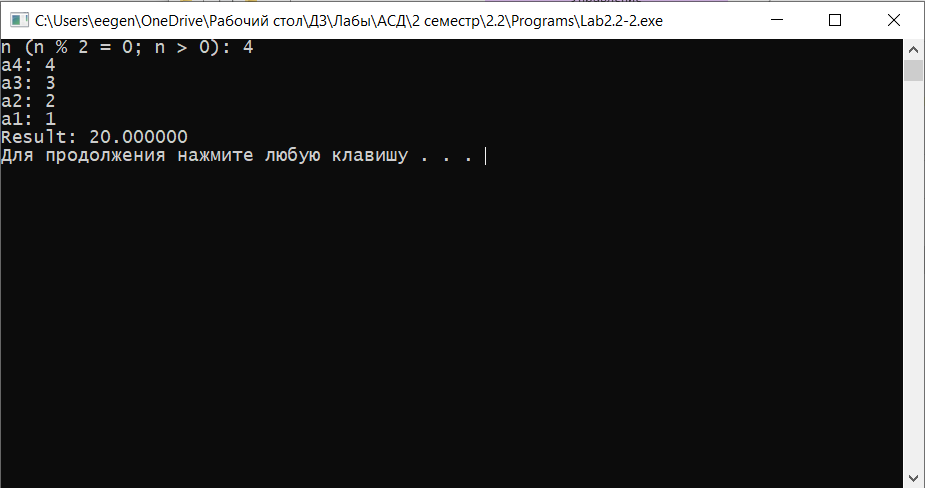
printf("Result: %lf\n", result);

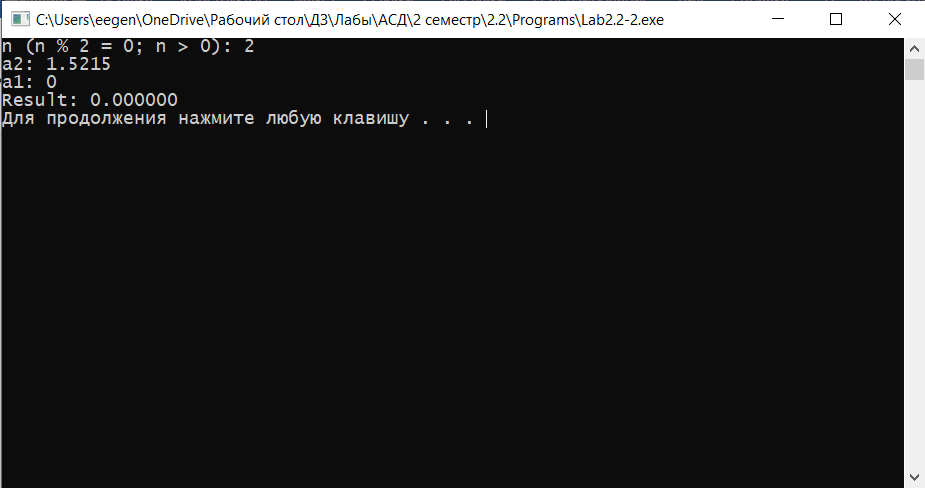
system("pause");

return 0;

}

**Тестування програми**

****

****