Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра Обчислювальна техніка та програмування

# Альбом звітів з

# практичних робіт по курсу «Алгоритми та структури даних»

Виконав студент групи КІТ–120А,

Макаренко Владислав Олександрович

Перевірила

Бречко В.О.

Харків 2021

**Лабораторна робота № 2. Рекурсивні та ітераційні**

**Алгоритми**

**Мета:** набути навички та практичний досвід у розробці рекурсивних програм.

**1 Вимоги**

**1.1 Розробник**

* Макаренко Владислав Олександрович
* Студент 1-го курсу
* Групи КІТ-120а

**1.2 Загальне завдання**

* Розробити рекурсивний та ітераційний алгоритми розв’язання індивідуального завдання.
* Визначити та порівняти час виконання відповідних функцій, зробити висновки.

**1.3 Індивідуальні завдання**

* Визначити значення числа *a* в ступені *n.* Значення *a* та *n* ввести з клавіатури.

**2 Описи програм**

**2.1 Програма з ітераційним алгоритмом**

**2.1.1 Код програми**

#include <stdlib.h>  
#include <stdio.h>  
#include <time.h>  
  
int power( int *x*, int *n*); // Повертає х у ступені n (n>=0)  
  
int main()  
{  
 int n; //степінь  
 int a; //число  
 clock\_t begin = clock();  
 printf("Введіть число: ");  
 scanf("%i", &a);  
 printf("Введіть степінь: ");  
 scanf("%i", &n);  
  
 if (n < 0){  
 printf("Неприпустимий аргумент функції");  
 }else{  
 int result = power(a, n);  
 printf("%i в степені %i дорівнює %i \n", a, n, result);  
 }  
 clock\_t end = clock();  
 double time\_spent = (double)(end - begin)/ CLOCKS\_PER\_SEC;  
 printf("Час, затрачений на виконання функції: %.10f", time\_spent);  
 return 0;  
}  
  
int power( int *x*, int *n*)  
{  
 if (*n* > 0 ){  
 int result = *x*;  
 int buf = *x*;  
 for(int i = 0; i < *n* - 1; i++){  
 result \*= buf;  
 }  
 return result;  
 }else{  
 return 1 ;// n == 0  
 }  
}

**2.1.2 Результати виконання програми**

Вводимо число *44* і за алгоритмом коду дізнаємося значення цього числа в степені *4* та отримуємо час виконання роботи алгоритму за допомогою функції *clock()* в секундах (див. рис. 1).

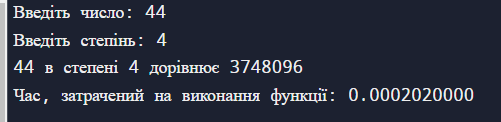


Рисунок 1 – Результати ітераційного алгоритму

**2.2 Програма з рекурсивним алгоритмом**

**2.2.1 Код програми**

#include <stdlib.h>  
#include <stdio.h>  
#include <time.h>  
  
  
int power( int *x*, int *n*); // Повертає х у ступені n (n>=0)  
  
int main()  
{  
 int n; //степінь  
 int a; //число  
 clock\_t begin = clock();  
 printf("Введіть число: ");  
 scanf("%i", &a);  
 printf("Введіть степінь: ");  
 scanf("%i", &n);  
 if (n < 0){  
 printf("Неприпустимий аргумент функції");  
 }else{  
 int result = power(a, n);  
 printf("%i в степені %i дорівнює %i \n", a, n, result);  
 }  
 clock\_t end = clock();  
 double time\_spent = (double)(end - begin)/ CLOCKS\_PER\_SEC;  
 printf("Час, затрачений на виконання функції: %.10f", time\_spent);  
 return 0;  
}  
  
int power( int *x*, int *n*)  
{  
 if (*n* > 0 ){  
 return power(*x*, *n*-1) \* *x*;  
 }else{  
 return 1 ;// n == 0  
 }  
}

**2.2.2 Результати виконання програми**

Вводимо число *44* і за алгоритмом коду дізнаємося значення цього числа в степені *4* та отримуємо час виконання роботи алгоритму за допомогою функції *clock()* в секундах (див. рис. 2).

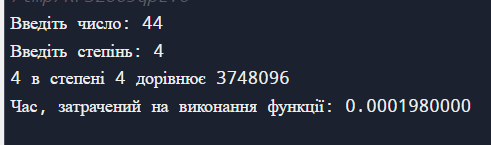


Рисунок 2 – Результати рекурсивного алгоритму

**3 Порівняння часу програм**

Порівняємо час виконання наших алгоритмів: ітераційного та рекурсивного методів за допомогою функції *clock()* (див. рис.3).

|  |  |  |
| --- | --- | --- |
|  | **<** |  |

Рисунок 3 – Порівняння часу

Як бачимо з (рис.3) на рекурсивний метод витрачається трохи менше часу ніж на ітераційний.

**Висновок:** на цій лабораторній роботі ми набули навички та практичний досвід у розробці рекурсивних програм та ітераційних, визначили та порівняли час обох методів.