Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра систем штучного інтелекту

Звіт

про виконання лабораторних та практичних робіт блоку № 4

На тему: «Одновимірні масиви. Двовимірні Масиви. Вказівники та Посилання. Динамічні масиви. Структури даних. Вкладені структури. Алгоритми обробки та робота з масивами та структурами.»

з дисципліни: «Основи програмування»

до:

ВНС Лабораторної Роботи №4
ВНС Лабораторної Роботи №5
Алготестер Лабораторної Роботи №2
Алготестер Лабораторної Роботи №3
Практичних Робіт до блоку №4

Виконав:

Студент групи ШІ-12
Кривичко Назар

Львів 2024

Тема роботи:

Одновимірні масиви. Двовимірні Масиви. Вказівники та Посилання. Динамічні масиви. Структури даних. Вкладені структури. Алгоритми обробки та робота з масивами та структурами.»

Мета роботи:

Дослідження одновимірних і двовимірних масивів для зберігання і впорядкування даних, що забезпечує швидкий доступ і обробку великих обсягів інформації.

Дослідження вказівників та посилань для розуміння адресації пам’яті та оптимізації використання ресурсів, що дозволяє ефективніше працювати з динамічними структурами даних.

Дослідження динамічних масивів для створення програм із змінною кількістю елементів, що підвищує гнучкість і адаптивність коду.

Дослідження структур даних та вкладених структур для організації складних об'єктів, що забезпечує кращу структуру і читабельність програмного коду.

Дослідження алгоритмів обробки масивів і структур для реалізації ефективної обробки даних, що сприяє написанню оптимізованих і масштабованих програм.

Теоретичні відомості:

У даній роботі розглядаються основні принципи роботи з масивами та структурами даних, зокрема одновимірні й двовимірні масиви для організації і зберігання великих обсягів даних. Особливу увагу приділено вказівникам і посиланням як засобам управління пам'яттю та ефективного доступу до даних. Розглянуто динамічні масиви, які забезпечують гнучке управління розміром даних під час виконання програми. Досліджено основи структур даних і вкладених структур для створення складних, логічно організованих об’єктів. Описано алгоритми обробки масивів і структур, що дозволяють ефективно

виконувати операції пошуку, сортування і модифікації даних, покращуючи оптимізацію коду.

Джерела :

книга - Stephen Prata - “ *C++ Primer Plus* ”
книга - *Aditya Y.Bhargava - “ Grokking algorithms ”*

Завдання № 3

Requirements :

VNS Lab 4

Time:

Expected: 30 min

Spent: 2h

```
int main(void)
{
    const int N = 100;
    int array[N];
    size_t userN;
    std::cout << "Enter array length: ";
    cin >> userN;
    if(userN > N)
    {
        std::cout << "Ooops... too much memory allocated, try again with less size" << std::endl;
        exit(-1);
    }

    fill_random(array,userN);
    print_array(array,userN);
    add_after_even(array,userN);
    print_array(array,userN);

    return 0;
}
```

```cpp
void destroy_first_zero(int* arr, const size_t& size)
{
    for (size_t i = 0; i < size; i++)
    {
        if(arr[i] == 0)
        {
            for (size_t j = i; j < size-1; j++)
            {
                arr[j] = arr[j+1];
            }
            arr[size-1] = -1; // for garbage ( deleted ) value
            break;
        }
    }
}

void add_after_even(int* arr, size_t& size)
{
    // Skiping first because it doesnt have past element
    for (size_t i = 1; i < size; i++) {
        if (arr[i] % 2 == 0) {
            for (size_t j = size; j > i + 1; --j) {
                arr[j] = arr[j - 1];
            }
            arr[i + 1] = arr[i-1] + 2;
            size++;
            i++;
        }
    }

}
```

```cpp
template<typename T>
void print_array(T* array, const size_t& size, const char& delim = ' ')
{
    for (size_t i = 0; i < size; i++)
    {
        std::cout << array[i] << delim;
    }
    std::cout << "\b\b"; // erasing last delimiter
    std::cout << std::endl;
}


template<typename T>
void fill_random(T* array,const size_t& size)
{
    srand((unsigned) time(NULL)); // for time dependency
    for (size_t i = 0; i < size; i++)
    {
        *(array + i) = rand() % 10 + 1  ;
    }
}
```

```
Enter array length: 30
9 7 5 7 1 10 8 10 10 5 1 9 4 3 9 3 3 4 5 1 3 9 8 9 7 8 9 8 1 4
9 7 5 7 1 10 3 8 5 10 7 10 9 5 1 9 4 11 3 9 3 3 4 5 5 1 3 9 8 11 9 7 8 9 9 8 11 1 4 3
```

Завдання № 4

Requirements :

VNS Lab 5

Time:

Expected: 30 min

Spent: 2h

```cpp
#include<iostream>


// 2. Написати функцію для обміну рядків двовимірного масиву з її допомогою
// відсортувати масив по елементах третього стовпця.


void swap_row(int matrix[4][4], int row1, int row2)
{
    if(row1 > 4 || row1 < 1 || row2 > 4 || row2 < 1)  return;


    row1--; row2--;
    int temp[4];
    for (int j = 0; j < 4; j++) {
        temp[j] = matrix[row1][j];
    }

    for (int j = 0; j < 4; j++) {
        matrix[row1][j] = matrix[row2][j];
        matrix[row2][j] = temp[j];
    }
}


void sort_matrix_by_3nd_column(int matrix[4][4])
{
    //Considering bubble sort for 3nd column element
    for (size_t i = 0; i < 4-1; i++)
    {
        if(matrix[i][2] > matrix[i+1][2])
        {
            swap_row(matrix, i+1,i+2);
            i--;
        }
    }
    swap_row(matrix,1,2);
}

void print_matrix(int matrix[][4],const size_t& rows, const size_t& cols)
{
    for (size_t i = 0; i < rows; i++)
    {
        for (size_t j = 0; j < cols; j++)
        {
            std::cout << matrix[i][j] << " ";
        }
        std::cout << std::endl;
    }
    std::cout << std::endl;
}

int main(void)
{
    int matrix[4][4] = { {1,6,3,9}, {9,0,5,4} , {4,5,1,8}, {1,5,8,2} };
    print_matrix(matrix,4,4);
    sort_matrix_by_3nd_column(matrix);
    print_matrix(matrix,4,4);

    return 0;
}
```

```
1 6 3 9
9 0 5 4
4 5 1 8
1 5 8 2

4 5 1 8
1 6 3 9
9 0 5 4
1 5 8 2
```

Завдання № 5

Requirements :

Algotester Lab 2

Time:

Expected: 30 min

Spent: 1h

```cpp
#include <iostream>
using namespace std;

void delete_elements(int*& arr, int& size, int a, int b, int c) {
    int newSize = 0;

    for (int i = 0; i < size; i++) {
        if (arr[i] != a && arr[i] != b && arr[i] != c) {
            newSize++;
        }
    }

    int* newArr = new int[newSize];
    int newIndex = 0;

    for (int i = 0; i < size; i++) {
        if (arr[i] != a && arr[i] != b && arr[i] != c) {
            newArr[newIndex++] = arr[i];
        }
    }

    delete[] arr;
    arr = newArr;
    size = newSize;
}

int* sum_neighbors(const int* arr, int size) {
    if (size <= 1) return nullptr;

    int* sumNeibArray = new int[size - 1];
    for (int i = 1; i < size; i++) {
        sumNeibArray[i - 1] = arr[i - 1] + arr[i];
    }
    return sumNeibArray;
}

void print_array(const int* arr, int size) {
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main() {
    int N;
    cin >> N;

    int* arr = new int[N];
    for (int i = 0; i < N; i++) {
        cin >> arr[i];
    }

    int a, b, c;
    cin >> a >> b >> c;

    delete_elements(arr, N, a, b, c);

    int* sumNeibArray = sum_neighbors(arr, N);

    if (sumNeibArray == nullptr) {
        cout << 0 << endl;
    } else {
        cout << N - 1 << endl;
        print_array(sumNeibArray, N - 1);
        delete[] sumNeibArray;
    }

    delete[] arr;
    return 0;
}
```

```
5
5 4 3 2 1
4 5 6
2
5 3
```

Завдання № 6

Requirements :

Algotester Lab 3

Time:

Expected: 30 min

Spent: 1h

```cpp
#include <iostream>
#include <string>

std::string compress_string(const std::string& s) {
    std::string compressed;
    int count = 1;

    for (size_t i = 1; i <= s.size(); ++i) {
        if (i < s.size() && s[i] == s[i - 1]) {
            ++count;
        } else {
            compressed.push_back(s[i - 1]);
            if (count > 1) {
                compressed += std::to_string(count);
            }
            count = 1;
        }
    }

    return compressed;
}

int main() {
    std::string s;
    std::cin >> s;

    std::string compressed = compress_string(s);
    std::cout << compressed << std::endl;

    return 0;
}
```

```
AAABBCCCEE
A3B2C3E2
```

Завдання № 7

Requirements :

Class Practise Task

Time:

Expected: 30 min

Spent: 1h

```cpp
#include<iostream>
#include<cmath>
// level
// eve
// v

bool isPalindrome(const std::string& str, int start, int end)
{
    if (start >= end) {
        return true;
    }
    if (str[start] != str[end]) {
        return false;
    }
    return isPalindrome(str, start + 1, end - 1);
}

// 34
// log10(120) = (int)2.xxxx + 1;

int numDigits(unsigned int n)
{
    return n == 0 ? 1 : (int)log10(n) + 1;
}

bool isPalindromeH(int num, int digits)
{
    if(num < 10) return true;

    int firstDigit = num / (int)pow(10, digits - 1);
    int lastDigit = num % 10;
    // 8090 ->
    // firstDigit = num / (int)10*3 -> 8
    // lastDigit = 8090 % 10 -> 809.[0] -< 0

    if (firstDigit != lastDigit) return false;

    num = (num % (int)pow(10,digits-1)) / 10;
    return isPalindromeH(num,digits-2);
}

bool isPalindrome(unsigned int num)
{
    return isPalindromeH(num,numDigits(num));
}

int main()
{
    std::string palindrome = "level", notPalindrome = "strike";
    unsigned int numP = 13031,numNP = 1902;

    std::cout << std::boolalpha;
    std::cout << "Is Palindrome: " << palindrome << " -> " << isPalindrome(palindrome,0,palindrome.size()-1) ? "Yes" : "No";
    std::cout << std::endl;
    std::cout << "Is Palindrome: " << notPalindrome << " -> " << isPalindrome(notPalindrome,0,notPalindrome.size()-1) ? "Yes" : "No";
    std::cout << std::endl;
    std::cout << "Is Palindrome: " << numP << " -> " << isPalindrome(numP) ? "Yes" : "No";
    std::cout << std::endl;
    std::cout << "Is Palindrome: " << numNP << " -> " << isPalindrome(numNP) ? "Yes" : "No";
    return 0;
}
```

```
Is Palindrome: level -> true
Is Palindrome: strike -> false
Is Palindrome: 13031 -> true
Is Palindrome: 1902 -> false
```

Завдання № 8

Requirements :

Self Algotester Task

Time:

Expected: 30 min

Spent: 1h

```cpp
#include <iostream>
#include <stack>
#include <vector>
#include <algorithm>

int main() {
    std::string s;
    std::cin >> s;

    std::stack<int> open_parens;
    std::vector<std::pair<int, int>> pairs;

    for (int i = 0; i < s.length(); i++) {
        if (s[i] == '(') {
            open_parens.push(i);
        } else if (s[i] == ')') {
            int open_pos = open_parens.top();
            open_parens.pop();
            pairs.push_back({open_pos + 1, i + 1});
        }
    }

    // std::reverse(pairs.begin(), pairs.end());

    std::cout << pairs.size() << "\n";
    for (const auto &p : pairs) {
        std::cout << p.first << " " << p.second << "\n";
    }

    return 0;
}
```

```
4+((7-4)+(4+4))*(7-4)
4
4 8
10 14
3 15
17 21
```

Pull Request: Link

Висновок:

Я навчився працювати з масивами, що дозволяє мені ефективно організовувати та зберігати дані. Я зрозумів, як ініціалізувати, заповнювати та звертатися до елементів масиву за індексами. Опанування алгоритмів для обробки масивів, таких як пошук і сортування, суттєво покращило мої навички програмування. Також я дізнався, як оптимізувати використання пам’яті при роботі з масивами. Знання про одновимірні масиви стали основою для подальшого вивчення більш складних структур даних.