Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

Зображення, що містить текст, Шрифт, коло, логотип

Автоматично згенерований опис

**Звіт**

про виконання

**Лабораторних та практичних робіт № 6**

***з дисципліни:*** «Мови та парадигми програмування»

***з розділу***: « Динамічні структури, Алгоритми обробки Дерев.»

***Виконав:***

студент групи ШІ – 12

Пушак Владислав Русланович

Львів 2023

# **Тема роботи:**

Динамічні структури, Алгоритми обробки Дерев.

# **Мета роботи:**

# Ознайомлення та вивчення використання динамічних структур даних, таких як Черга, Стек, Списки та Дерево, а також освоєння алгоритмів обробки даних в дереві. Розгляд особливостей застосування кожної структури та їх використання для оптимальної ефективності в конкретних завданнях.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Динамічні структури (Черга, Стек, Списки, Дерево).
* Тема №2: Алгоритми обробки Дерев.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічні структури (Черга, Стек, Списки, Дерево).
  + Джерела Інформації
    - Присутність на лекціях, опрацювання надоного матеріалу.
    - <https://www.geeksforgeeks.org/data-structures/>
    - <https://www.boardinfinity.com/blog/guide-to-5-data-structures-in-c/>
  + Що опрацьовано:
    - Ознайомився з різними структурами даних стеки, чергою, зв’язними списками та деревами.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 11.29.2023
  + Звершення опрацювання теми: 12.4.2023
* Тема №2: Алгоритми обробки Дерев.
  + Джерела Інформації:
    - Присутність на лекціях, опрацювання надоного матеріалу.
    - <https://www.programiz.com/dsa/trees>
  + Що опрацьовано:
    - Ознайомився з алгоритмами обробки дерев, властивості бінарного дерева, сортування ним та бінарним пошуком.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 11.29.2023
  + Звершення опрацювання теми: 12.4.2023

# **Виконання роботи:**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №\_1 VNS Lab 10 Task 1

* Варіант завдання: 11

Зображення, що містить текст, Шрифт, знімок екрана, білий

Автоматично згенерований опис

Завдання №\_2 Algotester Lab 5

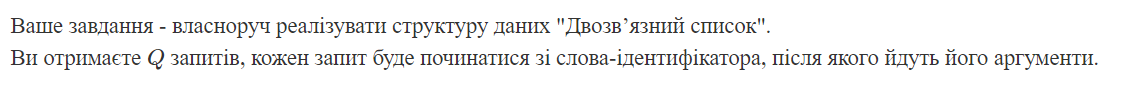
* Варіант завдання: 2

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Завдання №\_3 Algotester Lab 7-8

* Варіант завдання: 2



Завдання №\_4 Class Practice Task

* Реверс списку (Reverse list).
* Порівняння списків.
* Додавання великих чисел.

## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма №\_1 VNS Lab 10 Task 1

* Планований час на реалізацію 20 хв.

Програма №\_2 Algotester Lab 5

* Блок-схема

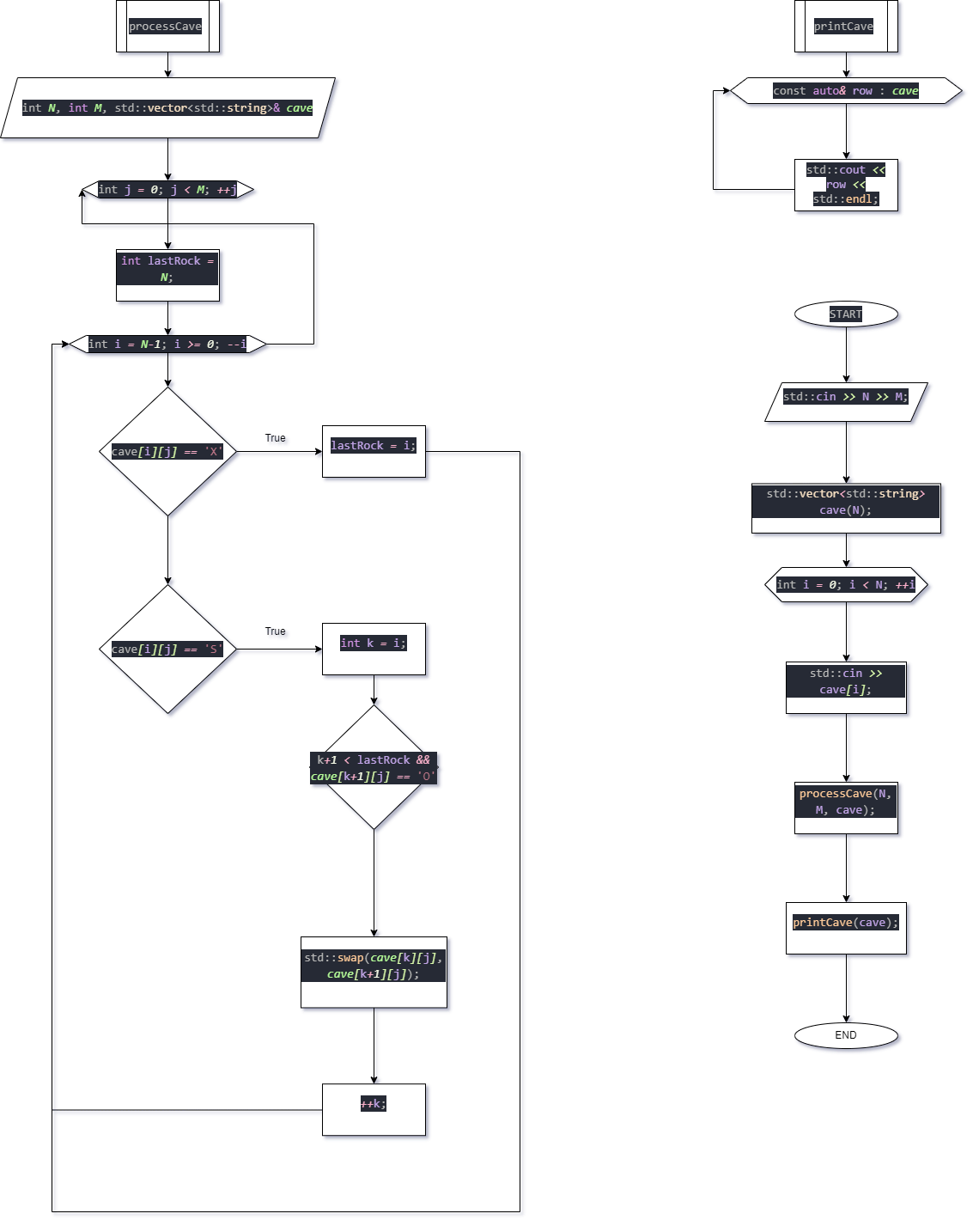


Рисунок Algotester 5 v 2

* Планований час на реалізацію 25 хв.

Програма №\_3 Algotester Lab 7-8

* Планований час на реалізацію 25 хв.

Програма №\_4 Class Practice Task 1

* Планований час на реалізацію 15 хв.

## **3. Код програм з посиланням на зовнішні ресурси:**

Завдання №\_1 VNS Lab 10 Task 1  
<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1121/files#diff-4cb3c1d68d1eef2145586b9d49e8b2bf8b665f5d792083bd462fe6b1ee49820d>

*#include* <iostream>

*#include* <fstream>

*#include* <string>

*#include* <memory>

*using* *namespace* std;

class Node {

public:

    string data;

    shared\_ptr*<*Node*>* prev;

    shared\_ptr*<*Node*>* next;

*Node*(const string& *value*) : data(*value*), prev(nullptr), next(nullptr) {}

};

class LinkedList {

private:

    shared\_ptr*<*Node*>* head;

public:

*LinkedList*() : head(nullptr) {}

    void *addElement*(const string& *data*) {

        auto newNode *=* make\_shared<Node>(*data*);

*if* (*!*head) {

            head *=* newNode;

        } *else* {

            auto temp *=* head;

*while* (temp*->*next) {

                temp *=* temp*->*next;

            }

            temp*->*next *=* newNode;

            newNode*->*prev *=* temp;

        }

    }

    void *printList*() const {

*if* (*!*head) {

            cout *<<* "List is empty" *<<* endl;

        } *else* {

            auto temp *=* head;

*while* (temp) {

                cout *<<* temp*->*data *<<* " ";

                temp *=* temp*->*next;

            }

            cout *<<* endl;

        }

    }

    void *deleteElement*(const string& *key*) {

        auto current *=* head;

*while* (current) {

*if* (current*->*data *==* *key*) {

*if* (current*->*prev) {

                    current*->*prev*->*next *=* current*->*next;

                } *else* {

                    head *=* current*->*next;

                }

*if* (current*->*next) {

                    current*->*next*->*prev *=* current*->*prev;

                }

*return*;

            }

            current *=* current*->*next;

        }

        cout *<<* "Element with a key '" *<<* *key* *<<* "' not found" *<<* endl;

    }

    void *addElementAtPosition*(const string& *data*, int *position*) {

        auto newNode *=* make\_shared<Node>(*data*);

*if* (*!*head *||* *position* *<=* *1*) {

            newNode*->*next *=* head;

*if* (head) {

                head*->*prev *=* newNode;

            }

            head *=* newNode;

        } *else* {

            auto temp *=* head;

*for* (int i *=* *1*; i *<* *position* *-* *1* *&&* temp; i*++*) {

                temp *=* temp*->*next;

            }

*if* (*!*temp) {

                cout *<<* "Error: Invalid insertion position" *<<* endl;

            } *else* {

                newNode*->*next *=* temp*->*next;

                newNode*->*prev *=* temp;

*if* (temp*->*next) {

                    temp*->*next*->*prev *=* newNode;

                }

                temp*->*next *=* newNode;

            }

        }

    }

    void *destroyList*() {

*while* (head) {

            auto temp *=* head;

            head *=* head*->*next;

        }

    }

    void *saveListToFile*(const string& *filename*) const {

        ofstream file(*filename*);

*if* (*!*file.*is\_open*()) {

            cout *<<* "Error with opening file for writing in" *<<* endl;

*return*;

        }

        auto temp *=* head;

*while* (temp) {

            file *<<* temp*->*data *<<* endl;

            temp *=* temp*->*next;

        }

        file.*close*();

    }

    static LinkedList *restoreListFromFile*(const string& *filename*) {

        LinkedList newList;

        ifstream file(*filename*);

*if* (*!*file.*is\_open*()) {

            cout *<<* "Error with opening file for reading" *<<* endl;

*return* newList;

        }

        string buffer;

*while* (getline(file, buffer)) {

            newList.*addElement*(buffer);

        }

        file.*close*();

*return* newList;

    }

};

int main() {

    LinkedList list;

    list.*addElement*("red");

    list.*addElement*("green");

    list.*addElement*("blue");

    cout *<<* "Created list: ";

    list.*printList*();

    list.*deleteElement*("green");

    cout *<<* "List after deleting the element with the key 'green': ";

    list.*printList*();

    list.*addElementAtPosition*("qwerty", *2*);

    cout *<<* "List after adding the 'qwerty' element at position 2: ";

    list.*printList*();

    string filename *=* "linkedlist.txt";

    list.*saveListToFile*(filename);

    list.*destroyList*();

    cout *<<* "List after destruction: ";

    list.*printList*();

    list *=* LinkedList::*restoreListFromFile*(filename);

    cout *<<* "Restored list: ";

    list.*printList*();

    list.*destroyList*();

    cout *<<* "List after final destruction: ";

    list.*printList*();

*return* *0*;

}

Код VNS lab 10

Завдання №\_2 Algotester Lab 5  
<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1121/files#diff-154a6a6423bc3b7ab189975524eba33dde70027b49198bf30f40d1967fecbe7f>

*#include* <iostream>

*#include* <vector>

*#include* <algorithm>

void processCave(int *N*, int *M*, std::vector<std::string>& *cave*) {

*for* (int j *=* *0*; j *<* *M*; *++*j) {

        int lastRock *=* *N*;

*for* (int i *=* *N-1*; i *>=* *0*; *--*i) {

*if* (*cave[*i*][*j*]* *==* 'X') {

                lastRock *=* i;

            } *else* *if* (*cave[*i*][*j*]* *==* 'S') {

                int k *=* i;

*while* (k*+1* *<* lastRock *&&* *cave[*k*+1][*j*]* *==* 'O') {

                    std::swap(*cave[*k*][*j*]*, *cave[*k*+1][*j*]*);

*++*k;

                }

            }

        }

    }

}

void printCave(const std::vector<std::string>& *cave*) {

*for* (const auto*&* row : *cave*) {

        std::cout *<<* row *<<* std::endl;

    }

}

int main() {

    int N, M;

    std::cin *>>* N *>>* M;

    std::vector*<*std::string*>* cave(N);

*for* (int i *=* *0*; i *<* N; *++*i) {

        std::cin *>>* cave*[*i*]*;

    }

    processCave(N, M, cave);

    printCave(cave);

*return* *0*;

}

Код Algo 5v2

Завдання №\_3 Algotester Lab 7-8

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1121/files#diff-90bfa8d1107b038d8ebd8f1e2e39e48011bc644ced51a9fae6d4bb999477f3b5>

*#include* <iostream>

*#include* <vector>

*using* *namespace* std;

struct Node {

    int data;

    Node*\** next;

    Node*\** prev;

*Node*(int *val*) : data(*val*), next(nullptr), prev(nullptr) {}

};

class DoublyLinkedList {

private:

    Node*\** head;

    Node*\** tail;

    int size;

public:

*DoublyLinkedList*() : head(nullptr), tail(nullptr), size(*0*) {}

    void *insert*(int *index*, int *N*, vector<int>& *values*) {

*for* (int i *=* *0*; i *<* *N*; *++*i) {

            Node*\** newNode *=* *new* Node(*values[*i*]*);

*if* (size *==* *0*) {

                head *=* tail *=* newNode;

            } *else* *if* (*index* *==* *0*) {

                newNode->next *=* head;

                head->prev *=* newNode;

                head *=* newNode;

            } *else* *if* (*index* *>=* size) {

                tail->next *=* newNode;

                newNode->prev *=* tail;

                tail *=* newNode;

            } *else* {

                Node*\** current *=* head;

*for* (int j *=* *0*; j *<* *index*; *++*j) {

                    current *=* current->next;

                }

                newNode->prev *=* current->prev;

                newNode->next *=* current;

                current->prev->next *=* newNode;

                current->prev *=* newNode;

            }

*++*size;

*++index*;

        }

    }

    void *erase*(int *index*, int *n*) {

*while* (*n* *>* *0* *&&* *index* *<* size) {

            Node*\** current *=* head;

*for* (int i *=* *0*; i *<* *index*; *++*i) {

                current *=* current->next;

            }

            Node*\** nextNode *=* current->next;

*if* (current->prev) {

                current->prev->next *=* nextNode;

            } *else* {

                head *=* nextNode;

            }

*if* (nextNode) {

                nextNode->prev *=* current->prev;

            } *else* {

                tail *=* current->prev;

            }

*delete* current;

*--*size;

*--n*;

        }

    }

    void *getSize*() const {

        cout *<<* size *<<* endl;

    }

    void *get*(int *index*) const {

*if* (*index* *>=* *0* *&&* *index* *<* size) {

            Node*\** current *=* head;

*for* (int i *=* *0*; i *<* *index*; i*++*) {

                current *=* current->next;

            }

            cout *<<* current->data *<<* endl;

        }

    }

    void *set*(int *index*, int *value*) {

*if* (*index* *>=* *0* *&&* *index* *<* size) {

            Node*\** current *=* head;

*for* (int i *=* *0*; i *<* *index*; i*++*) {

                current *=* current->next;

            }

            current->data *=* *value*;

        }

    }

    friend ostream& *operator<<*(ostream& *os*, const DoublyLinkedList& *list*) {

        Node*\** current *=* *list*.head;

*while* (current) {

*os* *<<* current->data *<<* " ";

            current *=* current->next;

        }

*os* *<<* endl;

*return* *os*;

    }

*~DoublyLinkedList*() {

        Node*\** current *=* head;

*while* (current) {

            Node*\** nextNode *=* current->next;

*delete* current;

            current *=* nextNode;

        }

        size *=* *0*;

    }

};

int main() {

    int Q;

    cin *>>* Q;

    DoublyLinkedList list;

*for* (int i *=* *0*; i *<* Q; i*++*) {

        string command;

        cin *>>* command;

*if* (command *==* "insert") {

            int index, N;

            cin *>>* index *>>* N;

            vector*<*int*>* values(N);

*for* (int*&* value : values) {

                cin *>>* value;

            }

            list.*insert*(index, N, values);

        } *else* *if* (command *==* "erase") {

            int index, n;

            cin *>>* index *>>* n;

            list.*erase*(index, n);

        } *else* *if* (command *==* "print") {

            cout *<<* list;

        } *else* *if* (command *==* "size") {

            list.*getSize*();

        } *else* *if* (command *==* "get") {

            int index;

            cin *>>* index;

            list.*get*(index);

        } *else* *if* (command *==* "set") {

            int index, value;

            cin *>>* index *>>* value;

            list.*set*(index, value);

        }

    }

*return* *0*;

}

Код Algo 7-8 v 1

Завдання №\_4 Class Practice Task 1

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1121/files#diff-47d2d720378dc419ba01fca1ff67ff7b8154fce53b87822da3309edc7847902f>

*#include* <iostream>

*#include* <vector>

*using* *namespace* std;

struct Node {

    int data;

    Node*\** next;

*Node*(int *val*) : data(*val*), next(nullptr) {}

};

class LinkedList {

private:

    Node*\** head;

    int size;

public:

*LinkedList*() : head(nullptr), size(*0*) {}

    void *add*(int *val*) {

        Node*\** newNode *=* *new* Node(*val*);

*if* (size *==* *0*) {

            head *=* newNode;

        } *else* {

            Node*\** current *=* head;

*while* (current->next) {

                current *=* current->next;

            }

            current->next *=* newNode;

        }

*++*size;

    }

    bool *compare*(LinkedList& *list*) {

*if* (size *!=* *list*.size) {

*return* false;

        }

        Node*\** current1 *=* head;

        Node*\** current2 *=* *list*.head;

*while* (current1 *&&* current2) {

*if* (current1->data *!=* current2->data) {

*return* false;

            }

            current1 *=* current1->next;

            current2 *=* current2->next;

        }

*return* true;

    }

    void *print*() {

        Node*\** current *=* head;

*while* (current) {

            cout *<<* current->data *<<* " ";

            current *=* current->next;

        }

        cout *<<* endl;

    }

*~LinkedList*() {

        Node*\** current *=* head;

*while* (current) {

            Node*\** nextNode *=* current->next;

*delete* current;

            current *=* nextNode;

        }

        size *=* *0*;

    }

};

int main() {

    LinkedList list1;

    list1.*add*(*1*);

    list1.*add*(*2*);

    list1.*add*(*3*);

    list1.*add*(*4*);

    list1.*add*(*5*);

    LinkedList list2;

    list2.*add*(*6*);

    list2.*add*(*7*);

    list2.*add*(*8*);

    list2.*add*(*9*);

    list2.*add*(*10*);

    cout *<<* "List 1: ";

    list1.*print*();

    cout *<<* "List 2: ";

    list2.*print*();

*if* (list1.*compare*(list2)) {

        cout *<<* "Lists are identical" *<<* endl;

    } *else* {

        cout *<<* "Lists are different" *<<* endl;

    }

*return* *0*;

}

Код Class Practice

## **4. Результати виконання завдань, тестування та фактично затрачений час:**

Завдання №\_1 VNS Lab 10 Task 1

* Варіант завдання: 11

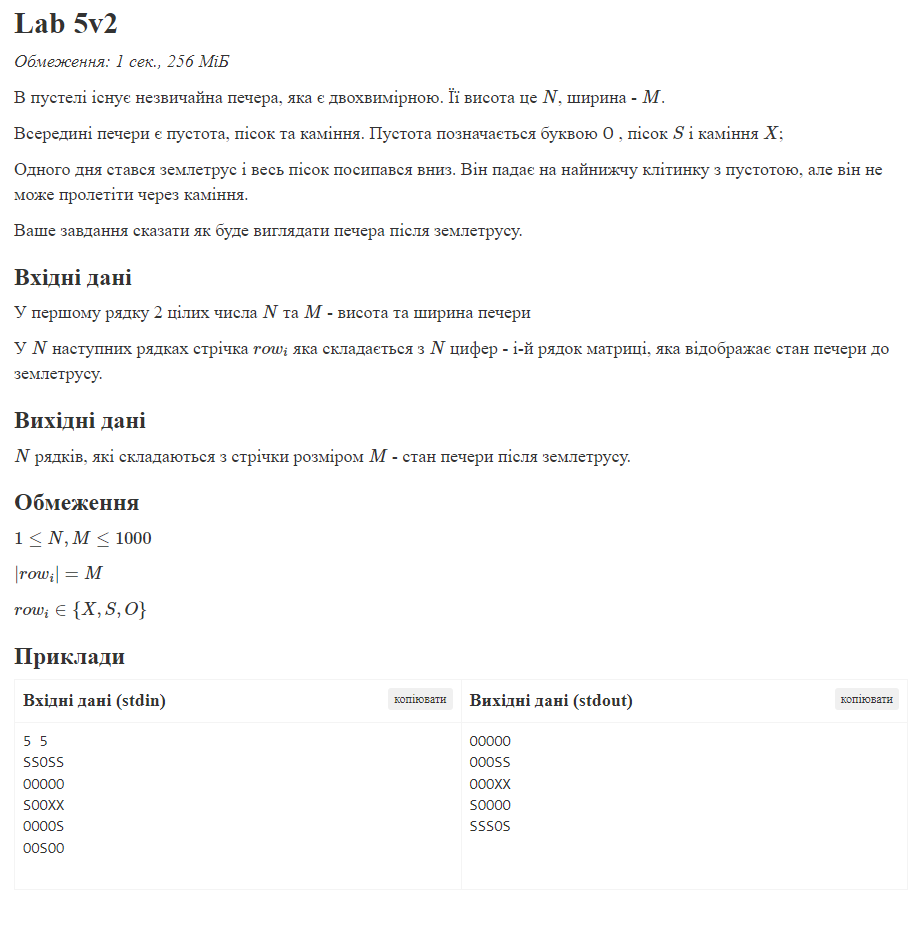
Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Результат

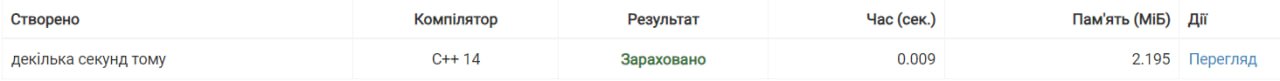
Час затрачений на виконання завдання : 35 хв.

Завдання №\_2 Algotester Lab 5



Зображення, що містить знімок екрана, Шрифт, текст, графічний дизайн

Автоматично згенерований опис



Результат

Час затрачений на виконання завдання : 25 хв.

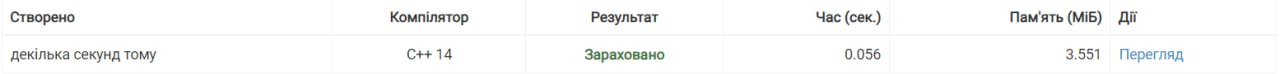
Завдання №\_3 Algotester Lab 7-8

*Зображення, що містить текст, знімок екрана, документ, Шрифт

Автоматично згенерований опис*

Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Автоматично згенерований опис



Результат

Час затрачений на виконання завдання : 30 хв.

Завдання №\_4 Class Practice Task

Зображення, що містить текст, Шрифт, знімок екрана, число

Автоматично згенерований опис

Результат

*.*

Час затрачений на виконання завдання : 25 хв.

## **5. Кооперація з командою:**

# **Висновки:**

Отримавши практичний досвід у використанні динамічних структур даних, я реалізував зв’язний список та впровадив алгоритми обробки дерев. Отримані навички вдосконалили моє розуміння алгоритмів і їхню практичну реалізацію в програмах. Процес вирішення завдань сприяв розвитку логічного мислення та допоміг оптимізувати код. Різноманітні практичні задачі, такі як обчислення суми великих чисел за допомогою зв’язних списків, розв'язання задачі про печеру, а також робота з алгоритмами обробки дерев, дозволили застосовувати теоретичні знання у реальних сценаріях програмування.