Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Кафедра ІПЗ

**Лабораторна робота № 5**

**«Структури: стек, черга, дерева»**

Виконав

ст. гр. ІП-22-1

Курчій С.В.

Перевірив

доцент Процюк В.Р.

Івано-Франківськ

2023

**Мета:**сформувати практичні навички організації таких поширених структур даних як стеки, черги і дерева, а також їх використання при вирішенні завдань

1. **Стек (Stack):**
   * Стек - це структура даних, яка працює за принципом "LastIn, FirstOut" (LIFO).
   * Елементи додаються та видаляються тільки з одного кінця стеку, який називається вершиною.
   * Операції на стеці: **push** (додавання елемента на вершину) та **pop** (видалення елемента з вершини).
2. **Черга (Queue):**
   * Черга - це структура даних, яка працює за принципом "FirstIn, FirstOut" (FIFO).
   * Елементи додаються в кінець черги, а видаляються з початку.
   * Операції на черзі: **enqueue** (додавання елемента в кінець) та **dequeue** (видалення елемента з початку).
3. **Дерева:**
   * Дерева - це ієрархічна структура даних, що складається з вузлів, з'єднаних гілками (ребрами).
   * Кожен вузол може мати декілька дочірніх вузлів, утворюючи піддерева.
   * Головний (верхній) вузол називається коренем, а вузли без дочірніх елементів - листями.
   * Різні види дерев включають бінарні дерева, двічні дерева пошуку, AVL-дерева і т.д

**Умова: Варіант 12.**

1. Заданий рядок тексту, який в тому числі містить множину дужок: "(", "{". "[" що відкриваються і закриваються. Рядок вважається коректним в тому випадку, якщо всім дужкам, що відкриваються, є відповідні дужки, що закриваються. Наприклад, рядок "abc (as) [] {аa [z]}" вважається коректним, а рядок "[[09] ({]})" – не є таким. Перевірити коректність рядка з використанням стека
2. Дано дві непусті черги, які містять однакову кількість елементів. Об'єднати черги в одну зі збереженням упорядкованості елементів.
3. Написати функцію, яка дозволяє визначити число входжень елемента х в бінарне дерево.

**Розв’язок**

1)

#include<iostream>

boolisCorrectString(conststd::string&str)

{

    constintn = str.length();

    charstack[n];

    inttop = -1; // Індекс вершини стека

    for (inti = 0; i<n; ++i)

    {

        charch = str[i];

        if (ch == '(' || ch == '{' || ch == '[')

        {

            stack[++top] = ch; // Додаємо в стек

        }

        elseif (ch == ')' || ch == '}' || ch == ']')

        {

            if (top == -1)

            {

                returnfalse; // Закриваюча дужка без відповідної відкриваючої

            }

            charopenBracket = stack[top--];

            if ((ch == ')'&&openBracket != '(') ||

                (ch == '}'&&openBracket != '{') ||

                (ch == ']'&&openBracket != '['))

            {

                returnfalse; // Невідповідні відкриваюча і закриваюча дужки

            }

        }

    }

    returntop == -1; // Всі відкриваючі дужки мають відповідні закриваючі

}

intmain()

{

    std::stringinput;

    std::cout<<"Enter a lineoftext: ";

    std::getline(std::cin, input);

    if (isCorrectString(input))

    {

        std::cout<<"Thestringisvalid.\n";

    }

    else

    {

        std::cout<<"Thestringisnotvalid.\n";

    }

    return0;

}

3)

#include<iostream>

#include<queue>

voidmergeQueues(std::queue<int>&q1, std::queue<int>&q2)

{

    std::queue<int>result;

    while (!q1.empty() && !q2.empty())

    {

        if (q1.front() <q2.front())

        {

            result.push(q1.front());

            q1.pop();

        }

        else

        {

            result.push(q2.front());

            q2.pop();

        }

    }

    while (!q1.empty())

    {

        result.push(q1.front());

        q1.pop();

    }

    while (!q2.empty())

    {

        result.push(q2.front());

        q2.pop();

    }

    // Переносимо результат об'єднання у вихідну чергу q1

    q1 = result;

}

intmain()

{

    std::queue<int>q1, q2;

    // Ініціалізація черг q1 та q2

    q1.push(2);

    q1.push(5);

    q1.push(8);

    q2.push(1);

    q2.push(3);

    q2.push(7);

    std::cout<<"Quque q1: ";

    std::queue<int>tempQ1 = q1;

    while (!tempQ1.empty())

    {

        std::cout<<tempQ1.front() <<"";

        tempQ1.pop();

    }

    std::cout<<std::endl;

    std::cout<<"Quque q2: ";

    std::queue<int>tempQ2 = q2;

    while (!tempQ2.empty())

    {

        std::cout<<tempQ2.front() <<"";

        tempQ2.pop();

    }

    std::cout<<std::endl;

    mergeQueues(q1, q2);

    // Виведення об'єднаної черги q1

    std::cout<<"Combinedqueue: ";

    while (!q1.empty())

    {

        std::cout<<q1.front() <<"";

        q1.pop();

    }

    return0;

}

5)

#include<iostream>

structTreeNode

{

    intdata;

    TreeNode \*left;

    TreeNode \*right;

    TreeNode(intval) : data(val), left(nullptr), right(nullptr) {}

};

// Функція для підрахунку кількості входжень заданого значення у бінарне дерево

intcountOccurrences(TreeNode \*root, intx)

{

    if (root == nullptr)

    {

        return0;

    }

    intcount = 0;

// Перевірка, чи дані поточного вузла дорівнюють заданому значенню

    if (root->data == x)

    {

        count++;

    }

// Рекурсивно підрахувати входження у лівому та правому піддеревах

    count += countOccurrences(root->left, x);

    count += countOccurrences(root->right, x);

    returncount;

}

intmain()

{

    // Ініціалізація бінарного дерева

    TreeNode \*root = newTreeNode(5);

    root->left = newTreeNode(3);

    root->right = newTreeNode(8);

    root->left->left = newTreeNode(2);

    root->left->right = newTreeNode(8);

    root->right->left = newTreeNode(7);

    root->right->right = newTreeNode(5);

    intx; // Шуканий елемент

    std::cout<<"Entertheitemyouarelookingfor: ";

    std::cin>>x;

    intoccurrences = countOccurrences(root, x);

    std::cout<<"Element"<<x<<"entersthetree"<<occurrences<<"times.\n";

    // Очищення пам'яті від бінарного дерева

    deleteroot->right->right;

    deleteroot->right->left;

    deleteroot->right;

    deleteroot->left->right;

    deleteroot->left->left;

    deleteroot->left;

    deleteroot;

    return0;

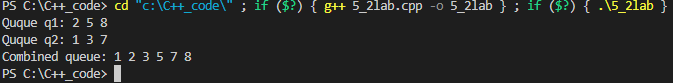
}

**Результат:  
1)**





**2)**

****

**3)**

****

****

**Висновок:**на цій лабораторній роботі я сформував практичні навички організації таких поширених структур даних як стеки, черги і дерева, а також їх використав при вирішенні завдань