Министерство образования Республики Беларусь

Учреждения образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №6

По дисциплине ОАИП за 2 семестр

«динамические структуры данных»

Выполнил:

Студент группы ИИ-21

1-го курса

Корпач Д.Р.

Проверил :

Гирель Т.Н.

Брест 2022

**Задание 1:** Создайте программой символьное двоичное дерево. Опишите логическую функцию, проверяющую, есть ли в непустом дереве хотя бы два одинаковых символа. В программе используйте подпрограммы.

#include<iostream>

using namespace std;

int tabs=0,cnt, val;

struct Branch{

  char Data;

  Branch \*LeftBranch;

  Branch \*RightBranch;

};

void Add(int aData, Branch \*&aBranch){//создание бинарного дерева

    if (!aBranch){                   //и запонение его веток

        aBranch = new Branch;

        aBranch->Data = aData;

        aBranch->LeftBranch = 0;

        aBranch->RightBranch = 0;

        return;

    }

    if (aBranch->Data>aData){

        Add(aData, aBranch->RightBranch);

    }

    else {

        Add(aData, aBranch->LeftBranch);

    }

}

void print(Branch \*aBranch){//вывод бинарного дерева

    if (!aBranch) return;

    tabs++;

    print(aBranch->LeftBranch);

    for (int i = 0; i < tabs; i++){

        cout<<" | ";

    }

    cout << aBranch->Data <<endl;

    print(aBranch->RightBranch);

    tabs--;

    return;

}

bool findDouble(Branch \*aBranch){

    if(aBranch){

        findDouble(aBranch->LeftBranch);

        if(cnt && (aBranch->Data == val)){

            ++cnt;

            if((cnt > 1) && (0 == aBranch->RightBranch)){

                for(int i = 0; i < cnt; ++i){

                    return true;

                }

                cnt = 1;

            }

        }

        else{

            if(cnt > 1){

                for(int i = 0; i < cnt; ++i){

                    return true;

                }

            }

            val = aBranch->Data;

            cnt = 1;

        }

        findDouble(aBranch->RightBranch);

    }

}

void FreeTree(Branch \*aBranch){//высвобождение памяти от бин дерева

    if (!aBranch) return;

    FreeTree(aBranch->LeftBranch);

    FreeTree(aBranch->RightBranch);

    delete aBranch;

    return;

}

int main(){

    Branch \*Root = 0;

    int n, \*even\_number;

    char cifr;

    bool double\_element=false;

    cout<<"input count cifr: ";cin >>n;

    for (int i = 0;i< n; i++){

        cin>>cifr;

        Add(cifr, Root);

    }

    print(Root);

    double\_element=findDouble(Root);

    if(double\_element==true){

        cout<<"bin tree have double element"<<endl;

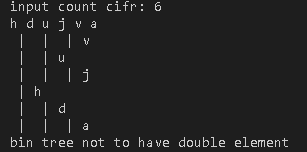
    }

    else{cout<<"bin tree not to have double element"<<endl;}

    FreeTree(Root);

    return 0;

}



**Задание 2:** В файле находится текст программы на Си. Используя стек, проверить правильность вложений циклов в этой программе.

#include<cstdlib>

#include<iostream>

#include<string>

#include<stack>

#include<fstream>

#include<vector>

using namespace std;

bool isOpenBracket(char symbol) {

  return symbol == '(' || symbol == '{' || symbol == '[';

}

bool isClosingBracket(char symbol) {

  return symbol == ')' || symbol == '}' || symbol == ']';

}

char getOpenBracketPair(char bracket) {

  return (bracket == '}') ? '{' : (bracket == ')') ? '(' :

    (bracket == ']') ? '[' : '\0';

}

bool areParenthesesGood(const std::string &source) {

  stack<char> stack;

  for (size\_t i = 0; i < source.size(); ++i) {

    if (isOpenBracket(source[i])) {

      stack.push(source[i]);

    } else {

      if (isClosingBracket(source[i])) {

        char openBracket = getOpenBracketPair(source[i]);

        if (!stack.empty() && stack.top() == openBracket) {

          stack.pop();

        } else {

          return false;

        }

      }

    }

  }

  return stack.empty();

}

 int main() {

    string expression = "";

   string mas[250];

    ifstream f;

    f.open("programC++.txt");

    int i = 0;

    while (!f.eof()) {

        f >> mas[i];

        expression+=mas[i];

        i++;

    }

  cout<<expression<<endl;

  bool check;

  check=areParenthesesGood(expression);

  if(check==1){

     cout<<"code is true"<<endl;

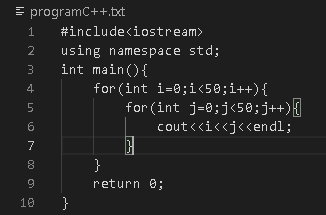
  }

  else{cout<<"code is false"<<endl;}

  f.close();

  return 0;

}





**Задание 3-4:** Написать программу, содержащую процедуры формирования и просмотра списка со строковой информационной частью. Включите в нее функцию, которая оставляет в информационной части только первое слово находящегося там предложения.

Написать программу, содержащую функцию, которая определяет, входит ли список М1 в список М2. Предполагается, что списки существуют.

#include <iostream>

using namespace std;

struct Node{

    string val;

    Node\* next;

    Node(string \_val) : val(\_val), next(nullptr) {}

};

struct list {

    Node\* first;

    Node\* last;

    list() : first(nullptr), last(nullptr) {}

    bool is\_empty() {

        return first == nullptr;

    }

    void push\_back(string \_val) {

        Node\* p = new Node(\_val);

        if (is\_empty()) {

            first = p;

            last = p;

            return;

        }

        last->next = p;

        last = p;

    }

    void remove\_first() {

    if (is\_empty()) return;

    Node\* p = first;

    first = p->next;

    delete p;

    }

    void remove\_last() {

        if (is\_empty()) return;

        if (first == last) {

            remove\_first();

            return;

        }

        Node\* p = first;

        while (p->next != last) p = p->next;

        p->next = nullptr;

        delete last;

        last = p;

    }

    void remove(string \_val) {

        if (is\_empty()) return;

        if (first->val == \_val) {

            remove\_first();

            return;

        }

        else if (last->val == \_val) {

            remove\_last();

            return;

        }

        Node\* slow = first;

        Node\* fast = first->next;

        while (fast && fast->val != \_val) {

            fast = fast->next;

            slow = slow->next;

        }

        if (!fast) {

            cout << "This element does not exist" << endl;

            return;

        }

        slow->next = fast->next;

        delete fast;

    }

    Node\* operator[] (const int index) {

        if (is\_empty()) return nullptr;

        Node\* p = first;

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

            p = p->next;

            if (!p) return nullptr;

        }

        return p;

    }

    void print() {

        if (is\_empty()) return;

        Node\* p = first;

        while (p){

            cout << p->val << " ";

            p = p->next;

        }

        cout << endl;

    }

};

void doubl\_check(int n,int m,list l\_list1,list l\_list2){

        int count=0;

        for(int i=0;i<n;i++){

            for(int j=0;j<m;j++){

                    if(l\_list1[i]->val==l\_list2[j]->val){

                        count+=1;

                        }

                }

        }

        if(count==0){

                cout<<"list1 is not in list2"<<endl;

        }

        else{cout<<"list1 includes in luist2 count : "<<count<<endl;}

}

int main(){

    list l\_list1,l\_list2;

    cout<<"input how much element in list 1: "<<endl;

    string element;int n;cin>>n;

    for(int i=0;i<n;i++){

        cin>>element;

        l\_list1.push\_back(element);

    }

    cout<<"input how much element in list 2: "<<endl;

    int m; cin>>m;

    for(int i=0;i<m;i++){

        cin>>element;

        l\_list2.push\_back(element);

    }

    doubl\_check(n,m,l\_list1,l\_list2);

    cout<<endl;

    cout<<"output our list1: "; l\_list1.print();

    cout<<"delete all elemens but not first: ";

    for(int i=1,j=1;i<n;i++){

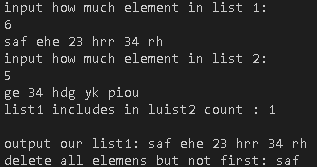
        l\_list1.remove(l\_list1[j]->val);

    }

    l\_list1.print();

    return 0;

}



**Задание 5:** Написать программу, содержащую процедуру, которая формирует список М, включив в него по одному разу элементы, которые входят хотя бы в один из списков М1 и М2.

#include <iostream>

using namespace std;

struct Node{

    string val;

    Node\* next;

    Node(string \_val) : val(\_val), next(nullptr) {}

};

struct list {

    Node\* first;

    Node\* last;

    list() : first(nullptr), last(nullptr) {}

    bool is\_empty() {

        return first == nullptr;

    }

    void push\_back(string \_val) {

        Node\* p = new Node(\_val);

        if (is\_empty()) {

            first = p;

            last = p;

            return;

        }

        last->next = p;

        last = p;

    }

    Node\* operator[] (const int index) {

        if (is\_empty()) return nullptr;

        Node\* p = first;

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

            p = p->next;

            if (!p) return nullptr;

        }

        return p;

    }

    void print() {

        if (is\_empty()) return;

        Node\* p = first;

        while (p){

            cout << p->val << " ";

            p = p->next;

        }

        cout << endl;

    }

    int doubl\_check;

};

void doubl\_check(int n,int m,list l\_listM1,list l\_listM2){

    list l\_listM;

        int count=0;

        for(int i=0;i<n;i++){

            for(int j=0;j<m;j++){

                    if(l\_listM1[i]->val==l\_listM2[j]->val){

                        count+=1;

                        l\_listM.push\_back(l\_listM1[i]->val);

                        }

                }

        }

        l\_listM.print();

}

int main(){

    list l\_listM1,l\_listM2;

    cout<<"input how much element in list 1: "<<endl;

    string element;int n;cin>>n;

    for(int i=0;i<n;i++){

        cin>>element;

        l\_listM1.push\_back(element);

    }

    cout<<"input how much element in list 2: "<<endl;

    int m; cin>>m;

    for(int i=0;i<m;i++){

        cin>>element;

        l\_listM2.push\_back(element);

    }

    cout<<"output list1: ";l\_listM1.print();

    cout<<"output list2: ";l\_listM2.print();

    cout<<"list M from list M1 and M2:  ";

    doubl\_check(n,m,l\_listM1,l\_listM2);

    return 0;

}

