|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Практическая работа № 3 | | |
| по дисциплине «Информационные технологии и основы программирования» | | |
| СТРУКТУРЫ ДАННЫХ: ДЕРЕВЬЯ | | |
|  | | |
|  | Бригада №3 | Ерощенко Артем, Кауфман Яна, Ковалева МАрия |
| Группа ПМИ-33 |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Преподаватель | Дворецкая виктория константиновна |
|  |  |
| Новосибирск,2024 | | |

**1.** **Цель работы:** изучить основные алгоритмы работы с деревьями; получить практические навыки разработки и использования этих структур и алгоритмов для решения задач.

**2. Задание:** Используя очередь или стек (считать уже описанными их типы и операции над ними), опишите на Си тип *бинарное дерево*(считая, что все элементы имеют некоторый простой тип) и реализуйте в виде процедуры или функции одну из перечисленных ниже операций над бинарным деревом.

d

NULL

NULL NULL

NULL NULL

NULL

NULL NULL

Используйте наиболее подходящий для решения задачи обход дерева (в глубину: префискный/инфиксный/постфиксный, в ширину):

Найти значение самого левого листа непустого дерева;

**3. Анализ задачи:**

**Входные данные:** пользовательский ввод с клавиатуры для выбора действия (u, l, r, 1, 2, 3, 4) и данных для добавления в узел.

**Выходные данные**: отображение структуры дерева после каждого действия пользователя и вывод определенной программы.

**Метод решения:**

Создание структуры дерева.

Основной цикл ввода пользователя:

* Отображать пользователю доступные действия (u, l, r, 1, 2, 3, 4).
* Считать ввод пользователя и выполнить соответствующее действие:

'u': Перейти к родительскому узлу, если возможно.

'l': Перейти к левому потомку текущего узла, если существует.

'r': Перейти к правому потомку текущего узла, если существует.

'1': Добавить новый узел слева от текущего узла.

'2': Добавить новый узел справа от текущего узла.

'3': Вывести значение самого левого узла дерева.

'4': Завершить выполнение программы.

Функции:

insert: Добавить узел слева или справа от текущего узла.

findLeftMost: Найти самый левый узел в дереве.

createTree: Инициализировать структуру дерева.

**4. Текст программы:**

#include <iostream>

#include <math.h>

#include <stack>

#include <utility>

#include <string>

#include <limits.h>

typedef struct Node {

int data;

Node \*left;

Node \*right;

Node \*parent;

} Node;

Node \*createTree(){

Node \*tree = new Node;

tree->left = NULL;

tree->right = NULL;

tree->parent = NULL;

return tree;

}

Node \*createTree(int data){

Node \*tree = createTree();

tree->data = data;

return tree;

}

Node \*createTree(int data, Node \*left, Node \*right){

Node \*tree = createTree();

tree->data = data;

tree->left = left;

tree->right = right;

return tree;

}

void insert(Node \*node, char where, int data){

Node \*buff = createTree(data);

buff->parent = node;

if(where == 'l'){

node->left = buff;

} else if(where == 'r'){

node->right = buff;

}

}

Node \*findLeftMost(Node \*tree){

while(tree->left != NULL){

tree = tree->left;

}

return tree;

}

void printTree(Node \*tree, Node \*current = NULL){

std::stack<std::pair<Node \*, int>> s;

s.push({tree, 0});

std::pair<Node \*, int> buff;

while(!s.empty()){

buff = s.top();

s.pop();

std::cout << std::string(std::max(buff.second - 1, 0), '|') << ((buff.second != 0) ? "∟": "") << ((buff.first == current) ? ">": "") << buff.first->data << '\n';

if(buff.first->left != NULL) s.push({buff.first->left, buff.second + 1});

if(buff.first->right != NULL) s.push({buff.first->right, buff.second + 1});

}

}

int main(){

Node \*tree = createTree(0);

Node \*current = tree;

char inp\_buff;

int num;

while(1){

std::cout << "Enter u - go up, l - go left, r - go right, 1 - add to left, 2 - add to right, 3 - print leftmost node, 4 - exit\n";

std::cin >> inp\_buff;

if(inp\_buff == 'u'){

if(current->parent != NULL){

current = current->parent;

} else {

std::cout << "You are already up)\n";

}

} else if(inp\_buff == 'l'){

if(current->left != NULL){

current = current->left;

} else {

std::cout << "There is no left node(\n";

}

} else if(inp\_buff == 'r'){

if(current->right != NULL){

current = current->right;

} else {

std::cout << "There is no right node(\n";

}

} else if(inp\_buff == '1'){

if(current->left == NULL){

std::cout << "Enter the data to store: ";

std::cin >> num;

insert(current, 'l', num);

} else {

std::cout << "Left node already exist)\n";

}

} else if(inp\_buff == '2'){

if(current->right == NULL){

std::cout << "Enter the data to store: ";

std::cin >> num;

insert(current, 'r', num);

} else {

std::cout << "Right node already exist)\n";

}

} else if(inp\_buff == '3'){

std::cout << "LeftMost Node: " << findLeftMost(tree)->data << '\n';

}else if(inp\_buff == '4'){

break;

}else {

std::cout << "Wrong mode!\n";

}

printTree(tree, current);

}

return 0;

}

**5. Тестирование программы:**

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Выходные данные | Примечание |
| 1 | 1,2,l,1,4,2,5,l,1,8,2,9,u,r,1,3,2,7 |  | Построение дерева |
| 2 | 3 | LeftMost Node: 8 | Вывод самого левого листа непустого дерева |

**6. Вывод**

Задача решена верно, что подтверждается результатами теста.