Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра Программной Инженерии

Лабораторная работа 11-12

По дисциплине «Основы алгоритмизации и программирования»

На тему «Разработка проекта с использованием бинарного дерева»

Выполнил:

Студент 1 курса 3 группы

Шатерник Г.И.

Преподаватель: асс. Андронова М.В.

2023, Минск

Вариант 15

11: Вершина бинарного дерева содержит ключ, строку и два указателя на потомков. Написать функцию определения количества узлов правой ветви дерева.

12:Добавить к проекту  функции смешанного и нисходящего обхода дерева с выводом  на консоль, проверки сбалансированности дерева и функцию в соответствии с вариантом из таблицы, представленной в лабораторной работе № 11, изменив ее так, чтобы функция соответствовала проекту данной лабораторной работы.

|  |
| --- |
| Программный код(11-12.cpp)  #include<iostream>  #include<conio.h>  using namespace std;  struct Node  {  int data;  Node\* left, \* right;  };  Node\* tree = nullptr;  void insert(int a, Node\*\* t)  {  if ((\*t) == NULL)  {  (\*t) = new Node;  (\*t)->data = a;  (\*t)->right = (\*t)->left = NULL;  return;  }  if (a > (\*t)->data)  insert(a, &(\*t)->left);  else  insert(a, &(\*t)->right);  }  void print(Node\* t, int u)  {  if (t == NULL) return;  else  {  print(t->left, ++u);  for (int i = 0; i < u; ++i)  cout << " ";  cout << t->data << endl;  u--;  }  print(t->right, ++u);  }  int d = 0;  int wrtreelast1(Node\* t)  {  if (t == NULL) return d;  if (t->data != NULL)  {  d++;  }  wrtreelast1(t->left);  return d;  }  int find(Node\* t, int el)  {  bool oo;  if (t == NULL) return 0;  else  {  find(t->right, el);  if (t->data == el)  {  oo = true;  }  }  }  bool find2(Node\* t, int el)  {  bool oo = false;  find(t, el);  return oo;  }  Node\* deltree(Node\* t)  {  if (t == NULL) return NULL;  deltree(t->left);  deltree(t->right);  delete(t);  t = NULL;  return NULL;  }  void wrtree(Node\* t)  {  if (t == NULL) return;  cout << t->data << " ";  wrtree(t->left);  wrtree(t->right);  }  void wrtree1(Node\* t)  {  if (t == NULL) return;  wrtree(t->left);  cout << t->data << " ";  wrtree(t->right);  }  void wrtree2(Node\* t)  {  if (t == NULL) return;  wrtree(t->left);  wrtree(t->right);  cout << t->data << " ";  }  int Depth(Node\* t)  {  if (t != NULL)  {  int A = Depth(t->left);  int B = Depth(t->right);  if (A > B) return A + 1;  else return B + 1;  }  else return 0;  }  bool IsBalanced(Node\* t)  {  int l, r;  if (t == NULL) return true;  else  {  if (IsBalanced(t->left) && IsBalanced(t->right))  {  l = Depth(t->left);  r = Depth(t->right);  if (l - r == 1 || r - l == 1 || l == r)  return true;  else  return false;  }  else  return false;  }  }  int main()  {  int k;  setlocale(LC\_CTYPE, "Russian");  int count, temp;  cout << "Введите количество элементов ";  cin >> count;  for (int i = 0; i < count; ++i)  {  cout << "Введите элемент ";  cin >> temp;  if (i == 1)  {  k = temp;  }  insert(temp, &tree);  }  cout << "Дерево\n";  print(tree, 0);  if (IsBalanced(tree) != 1)  cout << "Это сбалансированное дерево" << endl;  else cout << "Это НЕ сбалансированное дерево" << endl;  cout << "Количества узлов правой ветви дерева: " << wrtreelast1(tree) - 1;  int f;  cout << endl << endl;  cout << "\n Введите элемент, который нужно найти ";  cin >> f;  if (find2(tree, f) != 1)  cout << "Такой элемент есть в дереве\n ";  else cout << "Такого элемента в дереве нет\n";  cout << "прямой обход: \n";  wrtree(tree);  cout << "\n симметричный обход: \n";  wrtree1(tree);  cout << "\n обратный обход: \n";  wrtree2(tree);  } |
| Вывод: |