МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

Тема: Программирование алгоритмов с бинарными деревьями

Студентка гр. 7383 _____ Иолшина В.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

Содержание

Цель работы.

Познакомиться с такой нелинейной структурой данных, как бинарное дерево, способами ее представления и реализации, получить навыки решения задач и обработки бинарных деревьев.

Формулировка задачи: рассматриваются бинарные деревья с элементами типа char. Заданы перечисления узлов некоторого дерева b в порядке КЛП и ЛКП. Требуется:

а) восстановить дерево b и вывести его изображение;

б) перечислить узлы дерева b в порядке ЛПК.

Реализация задачи.

Бинарное дерево в данной работе реализовано с помощью массива структур. Используются следующие функции:

- bool isNull(binTree b, int i); //проверка на нулевой узел
- char RootBT(int i, binTree b); //взятие корня поддерева
- int Left(int i, binTree b); //взятие индекса левого узла
- int Right(int i, binTree b); //взятие индекса правого узла
- int ConsBT(char x, int lst, int rst, binTree b, int i); //добавление узла в список на базе вектора
- int createBT(base* arrayKLP, base* arrayLKP, int &i, binTree b); //создание дерева
- void printBT(binTree b, int i, int n); //печать дерева
- void LPK(binTree b, int i); //обход ЛПК

В функции main выводится приглашение выбрать способ ввода входных данных или закончить работу. В случае выбора ввода данных из файла, программа считывает строку из файла text.txt и записывает её в поток ввода, после этого закрывает файл. В случае выбора ввода информации с консоли, функция main считывает строку и записывает её в этот же поток ввода. Если файл пустой, то main выводит ошибку и начинает выполнение программы заново. Происходит вызов функции check_inp, которая проверяет, являются ли узлы в перечислениях КЛП и ЛКП одинаковыми, а также являются ли

перечисления одинаковыми по длине. Если все было введено верно, происходит вызов функции createBT. Функция createBT строит дерево с помощью функции ConsBT. Далее при помощи функции printBT дерево выводится на консоль. После этого происходит вызов функции LPK, которая обходит дерево в порядке ЛПК и перечисляет узлы.

Тестирование.

Процесс тестирования.

Программа собрана в операционной системе Ubuntu 16.04.2 LTS", с использованием компилятора g++ (Ubuntu 5.4.0-6ubuntu1~16.04.5). В других ОС и компиляторах тестирование не проводилось.

Результаты тестирования

Тестовые случаи представлены в Приложении А.

Во время тестирования было обнаружено, что считывание входных данных происходит неправильно, если между перечислениями КЛП и ЛКП больше одного пробела. Для устранения этого была добавлена проверка на пробелы.

Выводы.

В ходе выполнения лабораторной работы была изучена такая нелинейная структура данных, как бинарное дерево, способы её представления и реализации, получены навыки решения задач и обработки бинарных деревьев на языке C++. Была реализована программа восстанавливающая дерево по заданным перечислениям узлов в порядке КЛП и ЛПК и выводящая его изображение, а также перечисляющая узлы дерева в порядке ЛПК .

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ

bt.h :

```
#pragma once
typedef char base;

struct Node
{
  base info = '\0';
  int lt; //индекс левого узла
  int rt; //индекс правого узла
};
typedef Node binTree[100];
bool isNull(binTree b, int i); //проверка на нулевой узел
char RootBT(int i, binTree b); //взятие корня поддерева
int Left(int i, binTree b); //взятие индекса левого узла
int Right(int i, binTree b); //взятие индекса правого узла
int ConsBT(char x, int lst, int rst, binTree b, int i); //добавление узла в список на базе вектора
int createBT(base* arrayKLP, base* arrayLKP, int &i, binTree b);
void printBT(binTree b, int i, int n); //печать дерева
void LPK(binTree b, int i); //обход ЛПК
```

bt.cpp :

```
#include <fstream>
#include <iostream>
#include <cstdlib>
#include <cstring>
#include "bt.h"
using namespace std;
int count = 1;

bool isNull(binTree b, int i)
{
  return b[i].info == '\0';
}

char RootBT(int i, binTree b)
{
```

```
	if (b == NULL)
	{
		cerr << "Error: RootBT(null) \n";
		exit(1);
	}
	else
		return b[i].info;
}

int Left(int i, binTree b)
{
	if (b == NULL)
	{
		cerr << "Error: Left(null) \n";
		exit(1);
	}
	else
		return b[i].lt;
}

int Right(int i, binTree b)
{
	if (b == NULL)
	{
		cerr << "Error: Right(null) \n";
		exit(1);
	}
	else
		return b[i].rt;
}

int ConsBT(char x, int lst, int rst, binTree b, int i)
{
  if (b != NULL)
	{
		b[i].info = x;
		b[i].lt = lst;
		b[i].rt = rst;
		return i;
```

```
		}
		else
		{
			cerr << "Memory not enough\n";
			exit(1);
		}
}

int createBT(base* arrayKLP, base* arrayLKP, int &i, binTree b)
{
  int l, r;
  char c,c1;
  if(i<strlen(arrayKLP) && strlen(arrayLKP)!=0)
  {
    c=arrayKLP[i];
    c1=arrayLKP[strlen(arrayLKP)-1-i];
    for(int j=0;j<strlen(arrayLKP);j++)
    {
      if(arrayKLP[i]==arrayLKP[j])
      {
        char* gap_left_str=(char*)calloc(20,sizeof(char));
        strncpy(gap_left_str,arrayLKP,j);
        char* gap_right_str=(char*)calloc(20,sizeof(char));
        strncpy(gap_right_str,arrayLKP+j+1,strlen(arrayLKP)-j-1);
        i++;
        l=createBT(arrayKLP, gap_left_str, i, b);
        r=createBT(arrayKLP, gap_right_str, i, b);
        return ConsBT(c, l, r, b, count++);
      }
    }
  }
  else
      return 0;
}

void printBT(binTree b, int i, int n)
{
  if (i!=0)
  {
```

```
    if (!isNull(b, Right(i, b)))
      printBT(b, Right(i, b), n+1);
    else cout << endl;
    if(n==0)
      cout << RootBT(i, b)<<endl;
    for(int j=0; j<=n; j++)
      cout << "  ";
    if(n!=0)
      cout << RootBT(i, b)<<endl;
    if(!isNull(b, Left(i, b)))
      printBT(b, Left(i, b), n+1);
  }
  else
      return;
}

void LPK(binTree b, int i)
{
  if(b!=NULL && i!=0)
  {
    LPK(b, Left(i, b));
    LPK(b, Right(i, b));
    cout << RootBT(i, b);
  }
  else
    return;
}
```

Main.cpp :

```
#include <iostream>
#include <fstream>
#include "bt.h"
#include <sstream>
#include <ctype.h>
#include <cstring>

using namespace std;

bool check_inp(base* arrayKLP, base* arrayLKP)
```

```
{
  int k=0;
  for(int i=0; i<strlen(arrayKLP); i++)
  {
    for(int j=0; j<strlen(arrayLKP); j++)
    {
      if(arrayLKP[j] == arrayKLP[i])
      k++;
    }
  }
  if(k == strlen(arrayKLP))
    return true;
  else
  {
    cerr << "Перечисления содержат разные узлы" << endl;
    exit(1);
  }

}

int main()
{
  bool b = 0;
  char c;
  base arrayKLP[100];
  base arrayLKP[100];
  int run = 0;
    cout << "Введите 1, если хотите ввести выражение с клавиатуры.\n"
            "Введите 2, если еспользовать выражение из файла test.txt.\n"
            "Введите 3, если хотите закончить работу." << endl;
    cin >> run;
    switch(run)
    {
      case 1:
        {
          cout << "Введите КЛП и ЛКП: \n";
          cin.get();
          base c;
          do
```

```
    c=getchar();
    while(c == ' ');
    int j=0;
    while(!isspace(c))
    {
      arrayKLP[j]=c;
      j++;
      c=getchar();
    }
    do
    c=getchar();
    while(c == ' ');
    j=0;
    while(!isspace(c))
    {
      arrayLKP[j]=c;
      j++;
      c=getchar();
    }
    check_inp(arrayKLP, arrayLKP);
    b=1;
    break;
  }
case 2:
  {
    ifstream outfile;
    string str;
    outfile.open("test.txt");
    if (!outfile)
    {
      cout << "Входной файл не открыт!\n";
      b = 0;
      break;
    }
    outfile >> arrayKLP;
    outfile >> arrayLKP;
    check_inp(arrayKLP, arrayLKP);
    outfile.close();
    b=1;
```

```
        break;
      }
    case 3:
      {
        b=0;
        break;
      }
    default:
      {
        cout << "Введите верное число\n";
        b=0;
        break;
      }
  }
  if(b)
  {
    binTree b;
    int i=0, res=0, n=0;
    res = createBT(arrayKLP, arrayLKP, i, b);
    cout << "Результат работы программы: \n";
    printBT(b, i, n);
    cout << endl;
    LPK(b, i);
    cout << endl << endl;
  }
return 0;
}
```

ПРИЛОЖЕНИЕ Б.
ТЕСТОВЫЕ СЛУЧАИ

Ввод	Вывод	Верно
abcgdef cgbdeaf	Результат работы программы f a e d b c g gcedbfa	Да
apelri aelpri	Результат работы программы i r p l e a leirpa	Да
adejkflbghcimn djkelfaghbmnic	Реузльтат работы программы c i n m b h g a f l e k j d kjlfedhgnmicba	Да
5314972 1345792	Результат работы программы 2 9 7 5 4 3 1 1437295	Да

smadntzx admnstxz	Результат работы программы z x t s n m d a danmxzts	Да
Oijkwenf ohfdsaohnk	Перечисления содержат разные узлы	Да