|  |  |
| --- | --- |
|  | **Отчёт по лабораторной работе** №5 по курсу 1 Прикладная мат. и инф.  студента группы 08-108 Мохлякова Павла., № по списку 16. .  Адреса www, e-mail, jabber, skype. pmokhliakov@gmail.com  Работа выполнена: “ “ 2001г.  Преподаватель: каф.806. Поповкин Александр .  Входной контроль знаний с оценкой  Отчёт сдан “16“ апреля 2020г., итоговая оценка  Подпись преподавателя |

1. **Тема**: Динамические структуры данных. Обработка деревьев
2. **Цель работы**: Составить пограмму на языке СИ для простроения и обработки деревьев
3. **Задание** (*вариант №* 16 ): Проверить, является ла двоичное дерево симметричным
4. **Оборудование** (*лабораторное*):

ЭВМ PC , процессор i7-3770 , имя узла сети alisa с ОП 16384 МБ

НМД 400 ГБ. Терминал GNOME адрес 192.168.2.255 . Принтер

Другие устройства

*Оборудование ПЭВМ студента, если использовалось:*

Процессор Intel Core i5-3470 , ОП 8192 МБ, НМД 120 ГБ. Монитор Acer IPS 23’

Другие устройства

1. **Программное обеспечение** (*лабораторное*):

Операционная система семейства Linux , наименование Ubuntu версия 18.04.03

Интерпретатор команд bash версия 4.4.19

Система программирования gcc версия

Редактор текстов nano версия

Утилиты операционной системы make

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства Linux , наименование Manjaro версия 5.4.28

Интерпретатор команд zsh версия 5.8

Система программирования gcc версия

Редактор текстов atom версия

Утилиты операционной системы make

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

1. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальное описание с пред- и постусловиями)

⚙ pavel@lenovo  ~/Programs/C/lb23   master  cat main.c   
#include <stdio.h>   
#include "tree.h"   
  
int main()   
{   
 struct tree \*t=NULL;   
 printf("Заполнение дерева:\n");   
 while(1)   
 {   
   int v;   
   printf("Добавить новый элимент (да-1,нет-0):");   
   scanf("%d",&v);   
   if(v==0) break;   
   else   
   {   
     float j;   
     printf("Введите новый элимент:");   
     scanf("%f",&j);   
     add\_t(&t,j,NULL);   
     pr\_t(&t,0);   
   }   
 }   
 pr\_t(&t,0);   
 if(laba(&t)) printf("Дерево симметричное\n");   
 else printf("Дерево не симметричное\n");   
  
 return 0;   
}   
⚙ pavel@lenovo  ~/Programs/C/lb23   master  cat Makefile    
CC=gcc   
  
CFLAGS=-c -Wall   
  
all: lb23   
  
lb23: main.o tree.o   
       $(CC) main.o tree.o -o prog   
  
main.o: main.c   
       $(CC) $(CFLAGS) main.c   
  
tree.o: tree.c   
       $(CC) $(CFLAGS) tree.c   
  
clean:   
       rm -rf \*.o prog   
⚙ pavel@lenovo  ~/Programs/C/lb23   master  cat tree.h     
#ifndef \_TREE\_   
#define \_TREE\_   
  
struct tree{   
 float data;   
 struct tree \*parent;   
 struct tree \*right;   
 struct tree \*left;   
};   
  
  
void add\_t(struct tree \*\*root,float data,struct tree \*parent);   
void pr\_t(struct tree \*\*root,int level);   
void pad(int level);   
void dl\_t(struct tree \*\*root,float data,struct tree \*parent);   
int laba(struct tree \*\*root);   
#endif   
⚙ pavel@lenovo  ~/Programs/C/lb23   master  cat tree.c   
#include <stdlib.h>   
#include <stdio.h>   
#include "tree.h"   
  
void add\_t(struct tree \*\*root,float data,struct tree \*parent)   
{   
 if((\*root)==NULL)   
 {   
   (\*root)=malloc(sizeof(struct tree));   
   (\*root)->right=NULL;   
   (\*root)->left=NULL;   
   (\*root)->data=data;   
   (\*root)->parent=parent;   
   if(parent!=NULL)   
   {   
     if(data<parent->data) parent->left=(\*root);   
     else parent->right=(\*root);   
   }   
 }   
 else   
 {   
   if(data<(\*root)->data) add\_t(&((\*root)->left),data,(\*root));   
   else add\_t(&((\*root)->right),data,(\*root));   
 }   
}   
  
void pr\_t(struct tree \*\*root,int level)   
{   
 if((\*root)!=NULL)   
 {   
   pr\_t(&((\*root)->right),level+1);   
   pad(level);   
   printf("%g\n",(\*root)->data);   
   pr\_t(&((\*root)->left),level+1);   
  
 }   
 else   
 {   
   pad(level);   
   printf("-\n");   
 }   
}   
  
void pad(int level)   
{   
 for(int i=0;i<level;i++)   
 {   
   printf("\t");   
 }   
}   
  
void dl\_t(struct tree \*\*root,float data,struct tree \*parent)   
{   
 if((\*root)!=NULL)   
 {   
   if(data<(\*root)->data) dl\_t(&((\*root)->left),data,(\*root));   
   if(data>(\*root)->data) dl\_t(&((\*root)->right),data,(\*root));   
   if(data==(\*root)->data)   
   {   
     if(((\*root)->right==NULL)&&((\*root)->left==NULL))   
     {   
       if(parent->right->data==data)   
       {   
         struct tree \*q=(\*root)->parent;   
         free((\*root));   
         q->right=NULL;   
       }   
       else   
       {   
         struct tree \*q=(\*root)->parent;   
         free((\*root));   
         q->left=NULL;   
       }   
     }   
     else if(((\*root)->right!=NULL)&&((\*root)->left!=NULL))   
     {   
       struct tree \*last=(\*root);   
       last=last->right;   
       if(last->left!=NULL)   
       {   
         while(last->left!=NULL)   
         {   
           last=last->left;   
         }   
         (\*root)->data=last->data;   
         if(last->right==NULL)   
         {   
           last->parent->left=NULL;   
           free(last);   
         }   
         else   
         {   
           last->parent->left=last->right;   
           free(last);   
         }   
       }   
       else   
       {   
         if(last->right==NULL)   
         {   
           last->parent->right=NULL;   
           free(last);   
         }   
         else   
         {   
           last->parent->right=last->right;   
           free(last);   
         }   
       }   
     }   
     else if(((\*root)->right!=NULL)&&((\*root)->left==NULL))   
     {   
       struct tree \*last=(\*root);   
       struct tree \*pr=last->parent;   
       if(last==pr->right) pr->right=last->right;   
       else pr->left=last->right;   
       free(last);   
     }   
     else if(((\*root)->right==NULL)&&((\*root)->left!=NULL))   
     {   
       struct tree \*last=(\*root);   
       struct tree \*pr=last->parent;   
       if(last==pr->right) pr->right=last->left;   
       else pr->left=last->left;   
       free(last);   
     }   
   }   
 }   
}   
  
int laba(struct tree \*\*root)   
{   
 if((\*root)==NULL || ((\*root)->right==NULL && (\*root)->left==NULL)) return 1;   
 return 0;   
}

1. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты, либо соображения по тестированию].

Тесты:

⚙ pavel@lenovo  ~/Programs/C/lb23   master  ./prog   
Заполнение дерева:   
Добавить новый элимент (да-1,нет-0):1   
Введите новый элимент5   
       -   
5   
       -   
Добавить новый элимент (да-1,нет-0):0          
       -   
5   
       -   
Дерево симметричное   
⚙ pavel@lenovo  ~/Programs/C/lb23   master  ./prog   
Заполнение дерева:   
Добавить новый элимент (да-1,нет-0):1   
Введите новый элимент50   
       -   
50   
       -   
Добавить новый элимент (да-1,нет-0):1   
Введите новый элимент25    
       -   
50   
               -   
       25   
               -   
Добавить новый элимент (да-1,нет-0):1   
Введите новый элимент100   
               -   
       100   
               -   
50   
               -   
       25   
               -   
Добавить новый элимент (да-1,нет-0):0   
               -   
       100   
               -   
50   
               -   
       25   
               -   
Дерево не симметричное   
⚙ pavel@lenovo  ~/Programs/C/lb23   master  ./prog   
Заполнение дерева:   
Добавить новый элимент (да-1,нет-0):1   
Введите новый элимент50   
       -   
50   
       -   
Добавить новый элимент (да-1,нет-0):100   
Введите новый элимент100   
               -   
       100   
               -   
50   
       -   
Добавить новый элимент (да-1,нет-0):0   
               -   
       100   
               -   
50   
       -   
Дерево не симметричное

⚙ pavel@lenovo  ~/Programs/C/lb23   master  ./prog   
Заполнение дерева:   
Добавить новый элимент (да-1,нет-0):1   
Введите новый элимент50   
       -   
50   
       -   
Добавить новый элимент (да-1,нет-0):1   
Введите новый элимент25   
       -   
50   
               -   
       25   
               -   
Добавить новый элимент (да-1,нет-0):0   
       -   
50   
               -   
       25   
               -   
Дерево не симметричное

*Пункты 1-7 отчёта составляются* ***строго до*** *начала лабораторной работы.*

Допущен к выполнению работы. Подпись преподавателя

1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем)
2. **Дневник отладки** должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб.  или  дом. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  |  |  |  |  |  |  |

1. Замечание автора по существу работы
2. Выводы Научился работать с деревьями поискаю, реализовал функции для обработки деревьев.

Недочеты, допущенные при выполнении задания, могут быть устранены следующим образом

Подпись студента