Отчет по лабораторной работе № 23 по курсу «Практикум на ЭВМ»

Студент группы М8О-105Б-21 Минеева Светлана Алексеевна, № по списку 14

Контакты e-mail: svetlana.mineewa2003@yandex.ru

Работа выполнена: «30» апреля 2022 г.

Преподаватель: Вячеслав Константинович Титов каф. 805

Отчет сдан «30» апреля 2022 г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Динамические структуры данных. Обработка деревьев.

2. **Цель работы:** Составить программу на языке Си для построения и обработки дерева общего вида или упорядоченного двоичного дерева, содержащего узлы типа char.

3. **Задание** (*вариант №* 24)**:** Определить глубину дерева.

4. **Оборудование** (лабораторное):
ЭВМ _____, процессор _____, имя узла сети _____ с ОП _____ Мб,
НМД _____ Мб. Терминал _____ адрес _____. Принтер _____
Другие устройства _____
_____

Оборудование ПЭВМ студента, если использовалось:
Процессор 2,9 GHz 2-ядерный процессор Intel Core i5 с ОП 8 Гб, НМД 500 Гб. Монитор 13,3-дюймовый (2560 x 1600).
Другие устройства _____
_____

5. **Программное обеспечение (лабораторное):**
Операционная система семейства _____, наименование _____ версия _____
интерпретатор команд _____ версия _____
Система программирования _____ версия _____
Редактор текстов _____ версия _____
Утилиты операционной системы _____
_____
Прикладные системы и программы _____
Местонахождение и имена файлов программ и данных_____
_____

Программное обеспечение ЭВМ студента, если использовалось:
Операционная система семейства UNIX, наименование Terminal версия 2.10
интерпретатор команд bash версия 3.2.
Система программирования _____версия _____
Редактор текстов emacs версия 27.2-2
Утилиты операционной системы cat, ls, gcc и другие
Прикладные системы и программы _____
_____
Местонахождение и имена файлов программ и данных на домашнем компьютере _____

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

Алгоритм работы:

При запуске программы выводится меню, из которого мы понимаем - какая цифра какую выполняемую функцию обозначает. Далее при вводе какой-либо из этих цифр выполняется определенная операция и её результат выводится на экран. Рассмотрим каждую из операций:

1) «0» - завершает работу программы.
2) «1» - создание произвольного дерева общего вида с элементами типа char.
3) «2» - вывод дерева на экран. Проверяем, что дерево непустое. Если непустое, то, проходя по дереву, выводим его на экран.
4) «3» - добавить корневой элемент в дерево. На вход получаем имя корневого элемента, который нужно добавить, и добавляем его как корневой элемент в дерево.
5) «4» - добавить элемент-сын в дерево. На вход получаем элемент, для которого нужно добавить сына, и имя элемента-сына. Проходим по дереву и, найдя нужный элемент, добавляем для него элемент-сын (левое ответвление).
6) «5» - добавить элемент-брат в дерево. На вход получаем элемент, для которого нужно добавить брата, и имя элемента-брата. Проходим по дереву и, найдя нужный элемент, добавляем для него элемент-брат (на том же уровне, сверху над этим элементом).
7) «6» - найти отца для определенного элемента. На вход получаем элемент, для которого нужно найти отца. Проходим по дереву и, найдя введенный элемент, поднимаемся по дереву вверх на один уровень и выводим этот элемент.
8) «7» - найти старшего брата для определенного элемента. На вход получаем элемент, для которого нужно найти старшего брата. Проходим по дереву и, найдя введенный элемент, выводим элемент того же уровня, находящийся под введенным элементом.
9) «8» - удалить определенный элемент. Получаем на вход элемент, который нужно удалить. Проходя по дереву, находим этот элемент, удаляем его и все вершины, исходящие из него.
10) «9» - удалить дерево.
11) «10» - подсчитать глубину дерева. Определяем глубину по самому отдалённому сыну дерева, глубиной становится количество уровней от начала до уровня этого сына, включая этот уровень.

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

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
typedef char tdata;
int i,r,q,count,*h;
tdata m;
struct node;
typedef node * link;
struct node{
  tdata data;
  link son;
  link brother;};

void printtree(link t){
    static int l=0;
    if(t){
       printtree(t->brother);
       for(i=0;i<l;i++)printf("    ");
       printf("\\__%c\n",t->data);
      l++; printtree(t->son); l--;}}

link search(link tree, tdata c){
  if(tree){
    if(tree->data==c) return tree;
    link t=search(tree->son,c);
    if(!t) t=search(tree->brother,c);
    return t;}
```

```
   return 0;}

link searchfather(link tree, tdata c){
 link t;
 if(tree){
   if(tree->son&&tree->son->data==c) return tree;
   t=tree->son;
   while(t) { if(t->brother&&t->brother->data==c) return tree; t=t->brother;}
      t=0;  t=searchfather(tree->son,c);
      if(!t) t=searchfather(tree->brother,c);
      return t;} return 0;}

link searchbrother(link tree, tdata c){
   if(tree){
   if(tree->brother&&tree->brother->data==c) return tree;
   link t=searchbrother(tree->brother,c);
   if(!t) t=searchbrother(tree->son,c);
   return t;} return 0;}

void insertbrother(link &t,tdata v){
   if(!t) {t=new node ; t->data=v;
              t->son=0; t->brother=0;}
   else
  insertbrother(t->brother,v);}

void insertson(link &t,tdata v){
   if(!(t->son)) {t->son=new node ; t->son->data=v;
              t->son->son=0; t->son->brother=0;}
   else
  insertbrother(t->son,v);}

void inserttree(link &t,tdata v){
   if(!t) {t=new node ; t->data=v;
              t->son=0; t->brother=0;}
     else{  r=rand()%2;
              if(r) inserttree(t->son,v);
               else inserttree(t->brother,v);}}

void action(link tree, int *h, int count){
   if (tree!=0){
      if (count > *h) *h = count;
      action(tree->son, h, count + 1);
      action(tree->brother, h, count); }}

void mix(tdata *m, int n){
      int r; tdata s;
      for(i=0;i<n;i++){
         r=rand()%n; s=m[i]; m[i]=m[r]; m[r]=s;}}

int main(){
   long a; srand(time(&a));
   int k,n,count; tdata *m, f, s;
   link tree=0,t;
```

```
printf("MENU:\n0. Exit"
            "\n1. Generation random tree"
            "\n2. Print tree"
            "\n3. Insert root"
            "\n4. Insert son"
            "\n5. Insert brother"
            "\n6. Find father"
        "\n7. Find older brother"
        "\n8. Delete node"
        "\n9. Delete tree"
        "\n10. Tree depth\n");
for(;;){
   printf("\nInput number ==>"); scanf("%d",&k);
   if(k==0)break;
   if(k==1){ printf("\nInput number of nodes ==>"); scanf("%d",&n);
          m=new tdata[n];
          for(i=0;i<n;i++)  m[i]=100+i; mix(m,n);
                       for(i=0;i<n;i++) inserttree(tree,m[i]);}
   else
      if(k==2){ if(!tree) printf("\nTree is empty\n");
            else printtree(tree);}
   else
      if(k==3){ printf("\nInput value of root ==>"); scanf(" %c",&f);
                  t=new node ; t->data=f;
                  t->son=0; t->brother=0;
                  if(!tree) tree=t;
                  else { t->son=tree; tree=t; }}
   else
      if(k==4){ printf("\nInput value of father ==>"); scanf(" %c",&f);
                  t=search(tree,f);
                  if(!t) printf("\nFather not found\n");
                  else{ printf("\nInput value of son ==>"); scanf(" %c",&s);
                        insertson(t,s);}}
   else
      if(k==5){ printf("\nInput value of node ==>"); scanf(" %c",&f);
                  t=search(tree,f);
                  if(!t) printf("\nNode not found\n");
                  else{ printf("\nInput value of brother ==>"); scanf(" %c",&s);
                              insertbrother(t,s);}}
   else
      if(k==6){ printf("\nInput value of node ==>"); scanf(" %c",&f);
            t=searchfather(tree,f);
                  if(!t) printf("\nNode not found\n");
                  else printf("\nValue of father = %c\n",t->data);}
   else
      if(k==7){ printf("\nInput value of node ==>"); scanf(" %c",&f);
            t=searchbrother(tree,f);
                  if(!t) printf("\nNode not found\n");
                  else  printf("\nValue of older brother = %c\n",t->data);}
   else
      if(k==8){ printf("\nInput value of node ==>"); scanf(" %c",&f);
                  if(tree->data==f)if(tree->brother==0){tree=0; continue;}
                              else tree=tree->brother;
```

```
                else {t=searchbrother(tree,f);
                        if(t) t->brother=t->brother->brother;
                        else { t=searchfather(tree,f);
                                if(t) if(t->son->brother) t->son=t->son->brother;
                                        else t->son=0;
                                else printf("\nNode not found\n");}}}
    else
       if(k==9) tree=0;
    else
       if(k==10){ count=0;int h=0; action(tree,&h,count);
                  printf("\nTree depth = %d\n",h+1);}
    else break;}}
```

Так как при работе программы деревья создаются рандомно, то тесты заранее не предусмотрены, проверить верность выводимых данных для рандомно созданного дерева можно лишь непосредственно при работе программы, поэтому я рассматриваю здесь правильность тестов уже после создания протокола. Тесты:

```
1)
\__a
    \__d
    \__e
        \__h
        \__g
        \__f
            \__k

Глубина дерева = 4

2)
\__p
    \__e
\__f
    \__j
\__h
    \__g
        \__d
    \__i
        \__k
            \__l
                \__q
            \__r
                \__n
                \__o
                    \__m

Глубина дерева = 6

3)
\__j
\__g
\__k
    \__e
    \__h
        \__i
            \__f
            \__d

Глубина дерева = 4
```

4)

```
\__d
\__h
    \__f
    \__g
        \__e
```

Глубина дерева = 3

Пункты 1-7 отчета составляются строго до начала лабораторной работы.

8. Распечатка протокола (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем).

```
Last login: Sat Apr 30 22:13:41 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Pro-MacBook:~ macbookpro$ cat zag.txt
**********************************
*   Минеева Светлана Алексеевна   *
*          М8О-105Б-21            *
*    Лабораторная работа №23      *
**********************************
MacBook-Pro-MacBook:~ macbookpro$ ls |tail -4
PycharmProjects
lab23.cpp
pslog_20200707_123036.log
zag.txt
MacBook-Pro-MacBook:~ macbookpro$ cat lab23.cpp
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
typedef char tdata;
int i,r,q,count,*h;
tdata m;
struct node;
typedef node * link;
struct node{
  tdata data;
  link son;
  link brother;};

void printtree(link t){
    static int l=0;
    if(t){
       printtree(t->brother);
       for(i=0;i<l;i++)printf("    ");
       printf("\\__%c\n",t->data);
      l++; printtree(t->son); l--;}}

link search(link tree, tdata c){
  if(tree){
    if(tree->data==c) return tree;
    link t=search(tree->son,c);
    if(!t) t=search(tree->brother,c);
    return t;}
    return 0;}

link searchfather(link tree, tdata c){
  link t;
  if(tree){
    if(tree->son&&tree->son->data==c) return tree;
    t=tree->son;
    while(t) { if(t->brother&&t->brother->data==c) return tree; t=t->brother;}
        t=0;  t=searchfather(tree->son,c);
        if(!t) t=searchfather(tree->brother,c);
        return t;} return 0;}

link searchbrother(link tree, tdata c){
    if(tree){
    if(tree->brother&&tree->brother->data==c) return tree;
```

```
    link t=searchbrother(tree->brother,c);
    if(!t) t=searchbrother(tree->son,c);
    return t;} return 0;}

void insertbrother(link &t,tdata v){
    if(!t) {t=new node ; t->data=v;
        t->son=0; t->brother=0;}
    else
   insertbrother(t->brother,v);}

void insertson(link &t,tdata v){
    if(!(t->son)) {t->son=new node ; t->son->data=v;
        t->son->son=0; t->son->brother=0;}
    else
   insertbrother(t->son,v);}

void inserttree(link &t,tdata v){
    if(!t) {t=new node ; t->data=v;
        t->son=0; t->brother=0;}
      else{  r=rand()%2;
        if(r) inserttree(t->son,v);
         else inserttree(t->brother,v);}}

void action(link tree, int *h, int count){
    if (tree!=0){
        if (count > *h) *h = count;
        action(tree->son, h, count + 1);
        action(tree->brother, h, count); }}

void mix(tdata *m, int n){
        int r; tdata s;
        for(i=0;i<n;i++){
            r=rand()%n; s=m[i]; m[i]=m[r]; m[r]=s;}}

int main(){
    long a; srand(time(&a));
    int k,n,count; tdata *m, f, s;
    link tree=0,t;
    printf("MENU:\n0. Exit"
           "\n1. Generation random tree"
           "\n2. Print tree"
           "\n3. Insert root"
           "\n4. Insert son"
           "\n5. Insert brother"
           "\n6. Find father"
               "\n7. Find older brother"
               "\n8. Delete node"
               "\n9. Delete tree"
               "\n10. Tree depth\n");
    for(;;){
        printf("\nInput number ==>"); scanf("%d",&k);
        if(k==0)break;
        if(k==1){ printf("\nInput number of nodes ==>"); scanf("%d",&n);
                  m=new tdata[n];
                  for(i=0;i<n;i++)  m[i]=100+i; mix(m,n);
                   for(i=0;i<n;i++) inserttree(tree,m[i]);}
        else
            if(k==2){ if(!tree) printf("\nTree is empty\n");
            else printtree(tree);}
        else
            if(k==3){ printf("\nInput value of root ==>"); scanf(" %c",&f);
```

```
                    t=new node ; t->data=f;
                    t->son=0; t->brother=0;
                    if(!tree) tree=t;
                    else { t->son=tree; tree=t; }}
        else
            if(k==4){ printf("\nInput value of father ==>"); scanf(" %c",&f);
                    t=search(tree,f);
                    if(!t) printf("\nFather not found\n");
                    else{ printf("\nInput value of son ==>"); scanf(" %c",&s);
                     insertson(t,s);}}
        else
            if(k==5){ printf("\nInput value of node ==>"); scanf(" %c",&f);
                    t=search(tree,f);
                    if(!t) printf("\nNode not found\n");
                    else{ printf("\nInput value of brother ==>"); scanf("
%c",&s);
                          insertbrother(t,s);}}
        else
            if(k==6){ printf("\nInput value of node ==>"); scanf(" %c",&f);
                    t=searchfather(tree,f);
                    if(!t) printf("\nNode not found\n");
                    else printf("\nValue of father = %c\n",t->data);}
        else
            if(k==7){ printf("\nInput value of node ==>"); scanf(" %c",&f);
                    t=searchbrother(tree,f);
                    if(!t) printf("\nNode not found\n");
                    else  printf("\nValue of older brother = %c\n",t->data);}
        else
            if(k==8){ printf("\nInput value of node ==>"); scanf(" %c",&f);
                    if(tree->data==f)if(tree->brother==0){tree=0; continue;}
                     else tree=tree->brother;
                    else {t=searchbrother(tree,f);
                          if(t) t->brother=t->brother->brother;
                          else { t=searchfather(tree,f);
                           if(t) if(t->son->brother) t->son=t->son->brother;
                            else t->son=0;
                           else printf("\nNode not found\n");}}}
        else
            if(k==9) tree=0;
        else
            if(k==10){ count=0;int h=0; action(tree,&h,count);
                       printf("\nTree depth = %d\n",h+1);}
        else break;}}
MacBook-Pro-MacBook:~ macbookpro$ g++ -Wall -o lab23 lab23.cpp
MacBook-Pro-MacBook:~ macbookpro$ ls |tail -4
lab23
lab23.cpp
pslog_20200707_123036.log
zag.txt
MacBook-Pro-MacBook:~ macbookpro$ ./lab23
MENU:
0. Exit
1. Generation random tree
2. Print tree
3. Insert root
4. Insert son
5. Insert brother
6. Find father
7. Find older brother
8. Delete node
9. Delete tree
```

10. Tree depth

```
Input number ==>1

Input number of nodes ==>5

Input number ==>2
\__d
\__e
    \__h
    \__g
    \__f

Input number ==>3

Input value of root ==>a

Input number ==>2
\__a
    \__d
    \__e
        \__h
        \__g
        \__f

Input number ==>4

Input value of father ==>f

Input value of son ==>l

Input number ==>2
\__a
    \__d
    \__e
        \__h
        \__g
        \__f
            \__l

Input number ==>5

Input value of node ==>l

Input value of brother ==>k

Input number ==>2
\__a
    \__d
    \__e
        \__h
        \__g
        \__f
            \__k
            \__l

Input number ==>6

Input value of node ==>k

Value of father = f
```

```
Input number ==>7

Input value of node ==>g

Value of older brother = f

Input number ==>8

Input value of node ==>l

Input number ==>2
\__a
    \__d
    \__e
        \__h
        \__g
        \__f
            \__k

Input number ==>10

Tree depth = 4

Input number ==>9

Input number ==>2

Tree is empty

Input number ==>1

Input number of nodes ==>15

Input number ==>2
\__p
    \__e
\__f
    \__j
\__h
    \__g
        \__d
    \__i
        \__k
            \__l
                \__q
            \__r
                \__n
                \__o
                    \__m

Input number ==>10

Tree depth = 6

Input number ==>9

Input number ==>1

Input number of nodes ==>8

Input number ==>2
```

```
\__j
\__g
\__k
    \__e
    \__h
        \__i
            \__f
            \__d

Input number ==>10

Tree depth = 4

Input number ==>9

Input number ==>1

Input number of nodes ==>5

Input number ==>2
\__d
\__h
    \__f
    \__g
        \__e

Input number ==>10

Tree depth = 3

Input number ==>0
MacBook-Pro-MacBook:~ macbookpro$
```

9. Дневник отладки должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) 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	Дом.	30.04.22	21:50	Неверная печать дерева	Ошибка исправлена; подробнее изучены деревья общего вида	В функции printtree нужно увеличивать переменную l в условии if(t)

10. Замечания автора по существу работы

Эта лабораторная работа очень полезна, она отлично развивает мышление и учит программированию на языке Си.

11. Выводы

Я составила программу на языке Си для построения и обработки дерева общего вида или упорядоченного двоичного дерева, содержащего узлы типа char.

Недочёты при выполнении задания могут быть устранены следующим образом: больше практиковаться в написании программ на языке программирования Си.

Подпись студента Минеева С.А