|  |  |
| --- | --- |
| + | **Отчёт по лабораторной работе** №23  по курсу 1 фундаментальная информатика  студента группы М8О-105Б-21 Бондаревой Елены Евгеньвны, № по списку 1  Контакты www, e-mail, icq, skype : lena\_bondareva\_03@mail.ru  Работа выполнена: «4» мая 2022г.  Преподаватель: каф.806 В.К.Титов  Входной контроль знаний с оценкой  Отчёт сдан, 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. Основные функции работы с деревьями реализовать в виде универсальных процедур или функций. После того, как дерево создано, его обработка должна производиться в режиме текстового меню со след. действиями:

1) добавление нового узла

2) текстовая визуализация дерева

3) удаление узла

4) вычисление значения некоторой функции от дерева в соответствие с заданным действием.

1. **Задание:**

Проверить монотонность возрастания ширины уровня дерева.

1. **Оборудование** (*лабораторное*):

ЭВМ -, процессор -, имя узла сети-с ОП -МБ

НМД -ГБ. Терминал - адрес -. Принтер -.

Другие устройства -.

*Оборудование ПЭВМ студента, если использовалось:*

Процессор Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz , ОП 6 ГБ, НМД 240 ГБ. Монитор IPS 1920x1080

Другие устройства -.

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

Операционная система семейства -, наименование - версия -

Интерпретатор команд - версия -

Система программирования -версия -

Редактор текстов - версия -

Утилиты операционной системы -

Прикладные системы и программы -

Местонахождения и имена файлов программ и данных-

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства UNIX/GNU , наименование Ubuntu версия x86\_64

Интерпретатор команд bash

Редактор текстов emax

Утилиты операционной системы: head, du, grep, sum, tee, file, find, diff, tail, od, wc, cut, tar, touch, paste, uniq, gzip, sort, cmp, bzip2.

Прикладные системы и программы VTM-diagram

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)

Необходимо составить программу на языке Си для построения и обработки дерева общего вида или упорядоченного двоичного дерева. Для начала создадим menu, благодаря которому будем выбирать необходимое действие, которое будет осуществлять операцию. Отдельно создадим функции под: печать дерева, вставка нового элемента, удаление элемента, генерация(создание дерева из n элементов), очистить дерево, проверить монотонность возрастания ширины уровня дерева. Пользователь в любом порядке может проводить действия. Если дерево пустое, то пользователю будет выведено на экран сообщение: tree is empty. Функция check width выводит на экран YES, если ширина уровня дерева монотонно возрастает, NO- иначе.

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

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

struct node

{

char data;

struct node\* left;

struct node\* right;

};

typedef struct node tnode;

tnode\* add(tnode\* p, char data) {

if (p == NULL) {

p = (tnode\*)malloc(sizeof(tnode));

p->data = data;

p->left = p->right = NULL;

}

else if (p->data > data)

p->left = add(p->left, data);

else if (p->data < data)

p->right = add(p->right, data);

return p;

}

void printtree(tnode\* p, int level) {

if (p) {

printtree(p->right, level + 1);

for (int i = 0; i < level; i++)

printf(" ");

printf("\\\_\_%c\n", p->data);

printtree(p->left, level + 1);

}

}

void destroy(tnode\* p) {

if (p != NULL) {

destroy(p->left);

destroy(p->right);

free(p);

}

}

tnode\* remove\_item(tnode\* p, char val) {

if (p == NULL)

return p;

if (val == p->data) {

tnode\* tmp;

if (p->right == NULL)

tmp = p->left;

else {

tnode\* ptr = p->right;

if (ptr->left == NULL) {

ptr->left = p->left;

tmp = ptr;

}

else {

tnode\* pmin = ptr->left;

while (pmin->left != NULL) {

ptr = pmin;

pmin = ptr->left;

}

ptr->left = pmin->right;

pmin->left = p->left;

pmin->right = p->right;

tmp = pmin;

}

}

free(p);

return tmp;

}

else if (val < p->data)

p->left = remove\_item(p->left, val);

else

p->right = remove\_item(p->right, val);

return p;

}

int height(tnode\* p) {

if (p == NULL)

return 0;

else {

int lHeight = height(p->left);

int rHeight = height(p->right);

return (lHeight > rHeight) ? (lHeight + 1)

: (rHeight + 1);

}

}

int getWidth(tnode\* root, int level) {

if (root == NULL)

return 0;

if (level == 1)

return 1;

else if (level > 1)

return getWidth(root->left, level - 1)

+ getWidth(root->right, level - 1);

}

int check\_width(tnode\* root) {

int width = 1;

int h = height(root);

for (int i = 2; i <= h; i++) {

int current\_width = getWidth(root, i);

if (current\_width < width)

return 0;

width = current\_width;

}

return 1;

}

int main()

{

srand(time(0));

int k;

tnode\* root = NULL;

do

{

printf("\n MENU\n 0 - exit\n 1 - add random elements"

"\n 2 - print tree\n 3 - add item"

"\n 4 - delete item\n 5 - check width"

"\n 6 - clear tree\n ==>");

scanf("%d", &k);

while (getchar() != '\n');

char data;

if (k == 1) {

int n;

printf("n = ");

scanf("%d", &n);

for (int i = 0; i < n; i++) {

data = rand() % 26 + 'a';

root = add(root, data);

}

}

else if (k == 2) {

if (root != NULL) {

printtree(root, 0);

}

else

printf("tree is empty\n");

}

else if (k == 3) {

printf("Input symbol: ");

scanf("%c", &data);

root = add(root, data);

}

else if (k == 4) {

if (root != NULL) {

printf("Input symbol: ");

scanf("%c", &data);

root = remove\_item(root, data);

}

else

printf("tree is empty\n");

}

else if (k == 5) {

if (check\_width(root))

printf("YES\n");

else

printf("NO\n");

}

else if (k == 6) {

destroy(root);

root = NULL;

}

}

while (k != 0);

destroy(root);

}

*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)

elena@elena-Aspire-A315-53G:~$ cat tit.txt

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~ ~

~ Лабораторная работа № 23 ~

~ Динамические структуры данных. Обработка деревьев ~

~ ~

~ Бондарева Елена ~

~ М8О-105Б-21 ~

~ ~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

elena@elena-Aspire-A315-53G:~$ cat 23.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

struct node

{

char data;

struct node\* left;

struct node\* right;

};

typedef struct node tnode;

tnode\* add(tnode\* p, char data) {

if (p == NULL) {

p = (tnode\*)malloc(sizeof(tnode));

p->data = data;

p->left = p->right = NULL;

}

else if (p->data > data)

p->left = add(p->left, data);

else if (p->data < data)

p->right = add(p->right, data);

return p;

}

void printtree(tnode\* p, int level) {

if (p) {

printtree(p->right, level + 1);

for (int i = 0; i < level; i++)

printf(" ");

printf("\\\_\_%c\n", p->data);

printtree(p->left, level + 1);

}

}

void destroy(tnode\* p) {

if (p != NULL) {

destroy(p->left);

destroy(p->right);

free(p);

}

}

tnode\* remove\_item(tnode\* p, char val) {

if (p == NULL)

return p;

if (val == p->data) {

tnode\* tmp;

if (p->right == NULL)

tmp = p->left;

else {

tnode\* ptr = p->right;

if (ptr->left == NULL) {

ptr->left = p->left;

tmp = ptr;

}

else {

tnode\* pmin = ptr->left;

while (pmin->left != NULL) {

ptr = pmin;

pmin = ptr->left;

}

ptr->left = pmin->right;

pmin->left = p->left;

pmin->right = p->right;

tmp = pmin;

}

}

free(p);

return tmp;

}

else if (val < p->data)

p->left = remove\_item(p->left, val);

else

p->right = remove\_item(p->right, val);

return p;

}

int height(tnode\* p) {

if (p == NULL)

return 0;

else {

int lHeight = height(p->left);

int rHeight = height(p->right);

return (lHeight > rHeight) ? (lHeight + 1)

: (rHeight + 1);

}

}

int getWidth(tnode\* root, int level) {

if (root == NULL)

return 0;

if (level == 1)

return 1;

else if (level > 1)

return getWidth(root->left, level - 1)

+ getWidth(root->right, level - 1);

}

int check\_width(tnode\* root) {

int width = 1;

int h = height(root);

for (int i = 2; i <= h; i++) {

int current\_width = getWidth(root, i);

if (current\_width < width)

return 0;

width = current\_width;

}

return 1;

}

int main()

{

srand(time(0));

int k;

tnode\* root = NULL;

do

{

printf("\n MENU\n 0 - exit\n 1 - add random elements"

"\n 2 - print tree\n 3 - add item"

"\n 4 - delete item\n 5 - check width"

"\n 6 - clear tree\n ==>");

scanf("%d", &k);

while (getchar() != '\n');

char data;

if (k == 1) {

int n;

printf("n = ");

scanf("%d", &n);

for (int i = 0; i < n; i++) {

data = rand() % 26 + 'a';

root = add(root, data);

}

}

else if (k == 2) {

if (root != NULL) {

printtree(root, 0);

}

else

printf("tree is empty\n");

}

else if (k == 3) {

printf("Input symbol: ");

scanf("%c", &data);

root = add(root, data);

}

else if (k == 4) {

if (root != NULL) {

printf("Input symbol: ");

scanf("%c", &data);

root = remove\_item(root, data);

}

else

printf("tree is empty\n");

}

else if (k == 5) {

if (check\_width(root))

printf("YES\n");

else

printf("NO\n");

}

else if (k == 6) {

destroy(root);

root = NULL;

}

}

while (k != 0);

destroy(root);

}

elena@elena-Aspire-A315-53G:~$ g++ 23.cpp

elena@elena-Aspire-A315-53G:~$ ./a.out

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>1

n = 5

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>2

\\_\_z

\\_\_w

\\_\_v

\\_\_r

\\_\_i

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>3

Input symbol: o

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>2

\\_\_z

\\_\_w

\\_\_v

\\_\_r

\\_\_o

\\_\_i

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>3

Input symbol: k

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>2

\\_\_z

\\_\_w

\\_\_v

\\_\_r

\\_\_o

\\_\_k

\\_\_i

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>5

YES

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>6

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>2

tree is empty

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>1

n = 6

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>2

\\_\_y

\\_\_t

\\_\_n

\\_\_j

\\_\_f

\\_\_e

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>5

YES

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>6

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>2

tree is empty

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>1

n = 7

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>2

\\_\_u

\\_\_s

\\_\_p

\\_\_j

\\_\_h

\\_\_g

\\_\_e

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>3

Input symbol: a

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>2

\\_\_u

\\_\_s

\\_\_p

\\_\_j

\\_\_h

\\_\_g

\\_\_e

\\_\_a

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>4

Input symbol: p

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>2

\\_\_u

\\_\_s

\\_\_j

\\_\_h

\\_\_g

\\_\_e

\\_\_a

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>4

Input symbol: j

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>2

\\_\_u

\\_\_s

\\_\_h

\\_\_g

\\_\_e

\\_\_a

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>5

YES

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>6

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>2

tree is empty

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>1

n = 4

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>2

\\_\_x

\\_\_u

\\_\_m

\\_\_g

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>5

NO

MENU

0 - exit

1 - add random elements

2 - print tree

3 - add item

4 - delete item

5 - check width

6 - clear tree

==>0

**9. 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 | дом | 3.05.  2022 | 8:15 | В одной из функций поставила лишний символ «}». | Внимательно писать код. |  |

**10**. Замечание автора по существу работы

**11.** Выводы:

В результате выполнения работы я составила программу на языке Си для построения и обработки дерева общего вида или упорядоченного двоичного дерева, содержащего узлы типа char, а также для выполнения следующих действий: добавление нового узла, текстовая визуализация дерева, удаление узла, проверка монотонности возрастания ширины уровня дерева.

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

Подпись студента