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 "9" апреля 2020г., итоговая оценка

Подпись преподавателя

1. **Тема**: Деревья арифметических выражений

2. **Цель работы**: Составить программу выполнения заданных перобразований арифметических выражений с применением деревьев

3. **Задание** (*вариант №* 16): Убрать из частных все делители равные 1

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

5. **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

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

```
[redacted] ~/Programs/C/lb24 master cat main.c
#include <stdio.h>
#include <stdlib.h>
#include "stack.h"
#include "arvir.h"
#include "tree.h"

void laba(struct tree **root)
{
 if((*root)!=NULL)
 {
   int data;
   int type;
   peek_Tree(root,&data,&type);
   if((type==0)&&((data==47)||(data==42)))
   {
     if(((*root)->right->type==1)&&((*root)->right->data==1))
     {
       struct tree *q=(*root);
       free((*root)->right);
       (*root)=(*root)->left;
       free(q);
     }
     else if(((*root)->left->type==1)&&((*root)->left->data==1)&&(data==42))
     {
       struct tree *q=(*root);
       free((*root)->left);
       (*root)=(*root)->right;
       free(q);
     }
   }
   laba(&((*root)->right));
   laba(&((*root)->left));
 }
}

void tree_to_inf(struct tree **root,struct arvir **top,int last)
{
 if((*root)!= NULL)
 {
   int data;
   int type;
   int pres=-1;
   int sw=0;
   peek_Tree(root,&data,&type);
   if(type == 1) pres=0;
   else if((type==0)&&((data==42)||(data==47))) pres=2;
   else if((type==0)&&((data==43)||(data==45))) pres=1;
   if((pres<last)&&(pres!=0)) sw=1;
   if(sw==1) push_l(top,40,0);
   tree_to_inf(&((*root)->left),top,pres);
   struct tree *r=(*root)->right;
   pop_Tree(root,&data,&type);
   push_l(top,data,type);
   tree_to_inf(&r,top,pres);
   if(sw==1) push_l(top,41,0);
 }
}


void post_to_tree(struct arvir **first,struct tree **top)
{
 int data;
 int type;
 pop_l(first,&data,&type);
 push_tree(top,data,type);
 if(type == 0)
 {
   post_to_tree(first,&((*top)->right));
   post_to_tree(first,&((*top)->left));
 }
}
```

```
void inf_to_post(struct arvir *pre,struct arvir **pos)
{
 int ur=0;
 int s=size_ar(&pre);
 struct stack_i *ch=NULL;
 for(int i=0;i<s;i++)
 {
   int data;
   int type;
   pop_f(&pre,&data,&type);
   //-----------------------1-------------------------------------------
   if(type==1)
   {
     push_l(pos,data,1);
   }
   else
   {
     //----------------------2a-------------------------------------------
     if((peek_i(&ch)==40)||(ch==NULL))
     {
       if((data==42)||(data==47)) ur=1;
       if((data==43)||(data==45)) ur=0;
       push_i(&ch,data);
     }
     //----------------------2b-------------------------------------------
     else if((ur==0)&&((data==42)||(data==47)||(data==43)||(data==45)))
     {
       if((data==42)||(data==47)) ur=1;
       if((data==43)||(data==45)) ur=0;
       push_i(&ch,data);
     }
     else if((ur==1)&&((data==42)||(data==47))) push_i(&ch,data);
     //----------------------2c-------------------------------------------
     else if((ur==1)&&((data==43)||(data==45)))
     {
       while((peek_i(&ch)!=40)&&(size_i(&ch)!=0))
       {
         push_l(pos,pop_i(&ch),0);
       }
       ur=0;
       push_i(&ch,data);
     }
     //----------------------3-------------------------------------------
     else if(data==40) push_i(&ch,data);
     //----------------------4-------------------------------------------
     else if(data==41)
     {
       while(peek_i(&ch)!=40)
       {
         push_l(pos,pop_i(&ch),0);
       }
       pop_i(&ch);
       if((peek_i(&ch)==42)||(peek_i(&ch)==47)) ur=1;
       if((peek_i(&ch)==43)||(peek_i(&ch)==45)) ur=0;
     }
   }
 }
 s=size_i(&ch);
 for(int i=0;i<s;i++)
 {
   push_l(pos,pop_i(&ch),0);
 }
}

int main()
{
 char ch[100];
 struct arvir *arinf=NULL;
 struct arvir *arpost=NULL;
 struct tree *tr=NULL;
 scanf("%s",ch);
 strtoarvir(ch,100,&arinf);
 inf_to_post(arinf,&arpost);
 post_to_tree(&arpost,&tr);
```

```
 laba(&tr);
 print_Tree(&tr,0);
 tree_to_inf(&tr,&arpost,0);
 prarvir(&arpost);
 return 0;
}
[redacted] ~/Programs/C/lb24  master cat arvir.c
#include <stdio.h>
#include <stdlib.h>
#include "arvir.h"

void push_f(struct arvir **top,int d,int type)
{
 struct arvir *q;
 q=malloc(sizeof(struct arvir));
 q->data=d;
 q->type=type;
 if((*top)==NULL)
 {
   q->next=NULL;
   q->previous=NULL;
   q->last=q;
   (*top)=q;
 }
 else
 {
   q->next=(*top);
   q->previous=NULL;
   q->last=(*top)->last;
   (*top)=q;
 }
}


void push_l(struct arvir **top,int d,int type)
{
 struct arvir *q;
 q=malloc(sizeof(struct arvir));
 q->data=d;
 q->type=type;
 if((*top)==NULL)
 {
   q->next=NULL;
   q->previous=NULL;
   q->last=q;
   (*top)=q;
 }
 else
 {
   q->previous=(*top)->last;
   q->next=NULL;
   q->last=q;
   (*top)->last->next=q;
   (*top)->last=q;
 }
}

void pop_f(struct arvir **top,int *d,int *type)
{
 *d=(*top)->data;
 *type=(*top)->type;
 if((*top)->next==NULL)
 {
   free((*top));
   (*top)=NULL;
 }
 else
 {
   struct arvir *q;
   q=(*top);
   (*top)=(*top)->next;
   (*top)->previous=NULL;
   (*top)->last=q->last;
   free(q);
 }
```

```
}

void pop_l(struct arvir **top,int *d,int *type)
{
 struct arvir *l;
 l=(*top)->last;
 *d=l->data;
 *type=l->type;
 if(l->previous==NULL)
 {
   free((*top));
   (*top)=NULL;
 }
 else
 {
   struct arvir *q;
   q=l;
   l=q->previous;
   l->next=NULL;
   l->last=l;
   (*top)->last=l;
   free(q);
 }
}

int size_ar(struct arvir **top)
{
 int size=0;
 struct arvir *q=(*top);
 while(q!=NULL)
 {
   q=q->next;
   size++;
 }
 return size;
}

void peek_f(struct arvir **top,int *d,int *type)
{
 *d=(*top)->data;
 *type=(*top)->type;
}

void peek_l(struct arvir **top,int *d,int *type)
{
 struct arvir *l;
 l=(*top)->last;
 *d=l->data;
 *type=l->type;
}

void strtoarvir(char *array,int maxsize,struct arvir **top)
{
 int i=0;
 int a=0;
 int flag=0;
 while(1)
 {
   if((array[i]>=48)&&(array[i]<=57))
   {
     if(flag==0)
     {
       flag=1;
       a+=array[i]-48;
     }
     else
     {
       a*=10;
       a+=array[i]-48;
     }
   }
   else
   {
     if(flag==1)
     {
```

```
        flag=0;
        push_l(top,a,1);
        a=0;
        if(array[i]=='\0') break;
        push_l(top,(int)array[i],0);
      }
      else
      {
        if(array[i]=='\0') break;
        push_l(top,(int)array[i],0);
      }
    }
    if(array[i]=='\0') break;
    i++;
  }
}


void prarvir(struct arvir **top)
{
  struct arvir *q=(*top);
  int data;
  int type;
  for(int i=0;i<size_ar(top);i++)
  {
    peek_f(&q,&data,&type);
    if(type==0) printf("%c",(char)data);
    if(type==1) printf("%d", data);
    q=q->next;
  }
  printf("\n");
}
[redacted] ~/Programs/C/lb24  master  cat arvir.h
#ifndef _ARVIR_
#define _ARVIR_

struct arvir{
  int type;
  int data;
  struct arvir *next;
  struct arvir *previous;
  struct arvir *last;
};


void push_f(struct arvir **top,int d,int type);
void push_l(struct arvir **top,int d,int type);

void pop_f(struct arvir **top,int *d,int *type);
void pop_l(struct arvir **top,int *d,int *type);

int size_ar(struct arvir **top);

void peek_f(struct arvir **top,int *d,int *type);
void peek_l(struct arvir **top,int *d,int *type);

void strtoarvir(char *array,int maxsize,struct arvir **top);

void prarvir(struct arvir **top);

#endif
[redacted] ~/Programs/C/lb24  master  cat stack.c
#include <stdio.h>
#include <stdlib.h>
#include "stack.h"


void push_i(struct stack_i **top, int d)
{
  struct stack_i *q;
  q=malloc(sizeof(struct stack_i));
  q->data=d;
  if((*top)==NULL)
  {
    q->next=NULL;
```

```
   (*top)=q;
 }
 else
 {
   q->next=(*top);
   (*top)=q;
 }
}

int pop_i(struct stack_i **top)
{
 int d=(*top)->data;
 struct stack_i *q=(*top);
 (*top)=(*top)->next;
 free(q);
 return d;
}

int size_i(struct stack_i **top)
{
 struct stack_i *q;
 q=(*top);
 int size=0;
 while(q!=NULL)
 {
   size++;
   q=(q->next);
 }
 return size;
}

int peek_i(struct stack_i **top)
{
 if (*top!=NULL) return (*top)->data;
 else return -1;
}
[redacted] ~/Programs/C/lb24  master  cat stack.h
#ifndef _STACK_
#define _STACK_

struct stack_i{
 int data;
 struct stack_i *next;
} ;


void push_i(struct stack_i **top, int d);
int pop_i(struct stack_i **top);
int size_i(struct stack_i **top);
int peek_i(struct stack_i **top);

#endif
[redacted] ~/Programs/C/lb24  master  cat tree.c
#include <stdio.h>
#include <stdlib.h>
#include "tree.h"
#include "arvir.h"

void peek_Tree(struct tree **root,int *data,int *type)
{
 if((*root)!=NULL)
 {
   *data=(*root)->data;
   *type=(*root)->type;
 }
}

void pop_Tree(struct tree **root,int *data,int *type)
{
 if((*root)!=NULL)
 {
   *data=(*root)->data;
   *type=(*root)->type;
   free(*root);
 }
```

```
}

void push_tree(struct tree **top,int data,int type)
{
 if((*top)==NULL)
 {
   (*top)=malloc(sizeof(struct tree));
   (*top)->right=NULL;
   (*top)->left=NULL;
   (*top)->data=data;
   (*top)->type=type;
 }
 else (*top)->data=data;
}

void print_Tree(struct tree **root,int level)
{
   if((*root)!=NULL)
   {
     int data;
     int type;
     print_Tree(&((*root)->right),level+1);
     pr_padding('\t',level);
     peek_Tree(root,&data,&type);
     if(type==1) printf("%d\n",data);
     if(type==0) printf("%c\n",(char)data);
     print_Tree(&((*root)->left),level+1);
   }
}

void pr_padding(char ch,int level)
{
 for(int i=0;i<(level);i++)
 {
   printf("%c",ch);
 }
}
[redacted] ~/Programs/C/lb24  master cat tree.h
#ifndef _TREE_
#define _TREE_

struct tree{
 int data;
 int type;
 struct tree *right;
 struct tree *left;
};

void peek_Tree(struct tree **root,int *data,int *type);
void print_Tree(struct tree **root,int level);
void push_tree(struct tree **top,int data,int type);
void pr_padding(char ch,int level);
void pop_Tree(struct tree **root,int *data,int *type);

#endif
[redacted] ~/Programs/C/lb24  master cat Makefile
CC=gcc

CFLAGS=-c -Wall

all: lb25-26

lb25-26: main.o stack.o arvir.o tree.o
	$(CC) main.o stack.o arvir.o tree.o -o prog

main.o: main.c
	$(CC) $(CFLAGS) main.c

stack.o: stack.c
	$(CC) $(CFLAGS) stack.c

arvir.o: arvir.c
	$(CC) $(CFLAGS) arvir.c

tree.o: tree.c
```

```
	$(CC) $(CFLAGS) tree.c

clean:
	rm -rf *.o prog
```

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

Тесты:

```
[redacted] ~/Programs/C/lb24 master ./prog
(1-3)*2/1+89/1
    89
+
        2
    *
            3
        -
            1
(1-3)*2+89
[redacted] ~/Programs/C/lb24 master ./prog
2/1+3
    3
+
    2
2+3
[redacted] ~/Programs/C/lb24 master ./prog
1/2+3
    3
+
        2
    /
        1
1/2+3
```

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

Допущен к выполнению работы. Подпись преподавателя _____

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

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

10. Замечание автора по существу работы _____

_____

11. Выводы _____Найчился работать с деревьями арифметических выражений и реализовал программу для работы с ними на Си._____

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

_____

Подпись студента _____