|  |  |
| --- | --- |
|  | **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**: Cоставить 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

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/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;   
}   
pavel@lenovo  ~/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");   
}   
pavel@lenovo  ~/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   
pavel@lenovo  ~/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;   
}   
pavel@lenovo  ~/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   
pavel@lenovo  ~/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);   
 }   
}   
pavel@lenovo  ~/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   
pavel@lenovo  ~/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

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/lb24   master  ./prog    
(1-3)\*2/1+89/1   
       89   
+   
               2   
       \*   
                       3   
               -   
                       1   
(1-3)\*2+89   
pavel@lenovo  ~/Programs/C/lb24   master  ./prog   
2/1+3   
       3   
+   
       2   
2+3   
pavel@lenovo  ~/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

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 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