МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Алгоритмы и структуры данных»

Тема: Иерархические списки

Студентка гр. 7383 _____ Чемова К.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

СОДЕРЖАНИЕ

Цель работы

Ознакомится с иерархическим списком на практике, написать программу с реализацией иерархического списка на языке программирования C++.

Формулировка варианта 21:

Арифметическое, вычисление, постфиксная форма.

В задаче вычисления на входе дополнительно задаётся список значений переменных

((x_1 c_1) (x_2 c_2) … (x_k c_k)),

где x_i – переменная, а c_i – её значение (константа).

Реализация задачи

Для работы со списком были реализованы структуры `struct s_expr` и `struct two_ptr`:

```
struct s_expr;
struct two_ptr{
    s_expr *hd;
    s_expr* tl;
};
struct s_expr{
    bool tag;
    union{
        base atom;
        two_ptr pair;
    }node;
};
typedef s_expr* lisp;
```

Для работы со списком были написаны следующие функции `lisp cons, lisp make_char_atom, int read_s_expr, void read_lisp, bool write_lisp, bool write, void destroy`:

`lisp cons` – создает список;

`lisp make_char_atom` – создает атом;

`int read_s_expr`, `void read_lisp` – считывают список;

`bool write_lisp`, `bool write` – печатают список;

`void destroy` – удаляет список.

Для обработки выражения были написаны функуции `Pairs pairs, int count, string comp_pair`:

`Pairs pairs` – создает пару переменная-константа;

`int count`– считает количество символов, которое нужно заменить;

`string comp_pair` – заменяет в исходной строке буквы на цифры.

Обработка всех данных происходит в функции `void process`, чтобы избежать повторения кода.

Тестирование

Программа была собрана в компиляторе G++ в OS Linux Ubuntu 16.04 при помощи g++. В других системах тестирование не проводилось. Результаты тестирования приведены в приложении А.

Выводы

В ходе лабораторной работы была освоена работа с иерархическим списком, а также были получены практические навыки работы со стеком для решения задач на языке C++.

Была написана программа для вычисления заданной постфиксной формы.

ПРИЛОЖЕНИЕ А.

ТЕСТОВЫЕ СЛУЧАИ

В табл. 1 приведены примеры работы программы.

Таблица 1 – Тестовые случаи

Входные данные	Результат
10 5 + 2 * 66 3 / -	8
z x + c * ((z 1)(x 2)(c 3))	9
z 5 / c * 8 + ((z 5)(c 3))	11

ПРИЛОЖЕНИЕ Б.

КОД ПРОГРАММЫ

main.cpp:

```
#include <iostream>
#include <cctype>
#include <cstdlib>
#include <cstring>
#include <fstream>
#include <cstring>

using namespace std;

struct STACK{
      float arr[100];
      short topIndex;
      bool flag;
};
typedef struct STACK stack;

void push(stack* Stack, float value){
      (Stack->topIndex)++;
      if(Stack->topIndex == 100){
            printf("Стек переполнен!\n");
            exit(1);
      }
      Stack->arr[Stack->topIndex] = value;
}
void mul(stack *Stack){
           Stack->arr[Stack->topIndex-1] *= Stack->arr[Stack-
>topIndex];
    Stack->topIndex--;
}
void add(stack *Stack){
           Stack->arr[Stack->topIndex-1] += Stack->arr[Stack-
>topIndex];
    Stack->topIndex--;
}
void sub(stack *Stack){
           Stack->arr[Stack->topIndex-1] -= Stack->arr[Stack-
>topIndex];
    Stack->topIndex--;
}
void div(stack *Stack){
  if (Stack->arr[Stack->topIndex] != 0)
    Stack->arr[Stack->topIndex-1] /= Stack->arr[Stack->topIndex];
  else cout << "Деление на 0." << endl;
  Stack->topIndex--;
```

```
}


enum key{
    op_found = -2,
    smth_wrong = -1,
};

typedef char base;
struct s_expr;
struct two_ptr{
    s_expr *hd;
    s_expr* tl;
};
struct s_expr{
    bool tag;
    union{
        base atom;
        two_ptr pair;
    }node;
};
typedef s_expr* lisp;

struct pairs {//пара переменная-константа
    char var;
    char con;
};
typedef pairs Pairs;

void errors(int err) {
    switch(err) {
        case 1:
            cerr<<"1Неверный ввод."<<endl;
            break;
        case 2:
            cerr<<"Деление на нуль."<<endl;
            break;
        case 3:
            cerr<<"Не хвататет значений, констант или знаков.
Проверьте введенные данные и повторите попытку."<<endl;
            break;
        default:
            break;
    }
}

lisp cons (const lisp h, const lisp t){
    lisp p;
    p = new s_expr;
    if ( p == NULL) {
        errors(4);
```

```
        return NULL;
    }else {
        p->tag = false;
        p->node.pair.hd = h;
        p->node.pair.tl = t;
        return p;
    }
}

lisp make_char_atom (const base x) {
    if(x=='+' || x=='-' || x=='*' || x=='/' || isalpha(x) ||
isdigit(x)){
        lisp s;
        s = new s_expr;
        s -> tag = true;
        s->node.atom = x;
        return s;
    }else{
        errors(3);
        return NULL;
    }
}

int read_s_expr(string array,int* ptr){
    int num=0;
    while(!isspace(array[(*ptr)]) && (*ptr)< array.length()){
        (*ptr)++;
    }
    (*ptr)++;
    if((*ptr) > 2 && isspace(array[(*ptr)-3])){
        if(isdigit(array[(*ptr)-2]) || isalpha(array[(*ptr)-2]))
            return array[(*ptr)-2];
        else if(array[(*ptr)-2] == '+' || array[(*ptr)-2]=='-' ||
                array[(*ptr)-2]=='*' || array[(*ptr)-2]=='/')
            return op_found;
        else
            return smth_wrong;
    }else if((*ptr) <= 2){
        if(isdigit(array[(*ptr)-2]) || isalpha(array[(*ptr)-2]))
            return array[(*ptr)-2];
        else if(array[(*ptr)-2] == '+' || array[(*ptr)-2]=='-' ||
                array[(*ptr)-2]=='*' || array[(*ptr)-2]=='/')
            return op_found;
        else
            return smth_wrong;
    }else
        return smth_wrong;
}

void read_lisp(string array,lisp* head,int* ptr){
    int num=0;
```

```
    num=read_s_expr(array,ptr);
    if(num!=smth_wrong && num!=op_found){
        lisp p11;
        p11=make_char_atom(array[(*ptr)-2]);
        int num2,num3;
        num2=read_s_expr(array,ptr);
        num3=read_s_expr(array,ptr);
        lisp p12;
        if(num3==smth_wrong || num2==smth_wrong){
            return;
        }
        if(num3!=smth_wrong && num3!=op_found){
            lisp p1,p2;
            (*ptr)=(*ptr)-4;
            read_lisp(array,&p1,ptr);
            read_lisp(array,&p2,ptr);
            p12=cons(p1,p2);
        }
        if(num3==op_found){
            if(num2!=op_found && num2!=smth_wrong){
                lisp p1,p2;
                p1=make_char_atom(array[(*ptr)-4]);
                p2=make_char_atom(array[(*ptr)-2]);
                p12=cons(p1,p2);
            }
        }
        (*head)=cons(p11,p12);
    }else if(num==op_found && (*ptr)!= 0){
        (*head)=make_char_atom(array[(*ptr)-2]);
    }else{
        return ;
    }
}

bool write_lisp(lisp head, int i,int j,bool* flag);

bool write(lisp head,int i,int j,bool* flag){
    if(head==NULL){
        cerr<< "2Неверный ввод."<<endl;
        return false;
    }else if(head->tag==true){
        cerr<<"Не хвататет значений, констант или знаков. Проверьте
введенные данные и повторите попытку."<<endl;
        return false;
    }else{
        i++;
        write_lisp(head->node.pair.hd,i,j,flag);
        j++;
        write_lisp(head->node.pair.tl,i,j,flag);
        return true;
    }
```

```
}

bool write_lisp(lisp head, int i,int j,bool* flag){
    if(head==NULL){
        cerr<<"Не хвататет значений, констант или знаков. Проверьте
введенные данные и повторите попытку."<<endl;
        return false;
    }
    if(head->tag==true){
        for(int j=0;j< i;j++)
            cout<<'\t';
        if(head->node.atom=='+' || head->node.atom=='-' || head-
>node.atom=='*'
           || head->node.atom=='/' || (isalpha(head->node.atom))){
            if(head->node.atom=='/' && (*flag)==true){
                errors(2);
                return false;
            }
            cout<<head->node.atom<<endl;
        }else{
            cout<<head->node.atom<<endl;
            if(head->node.atom == '0')
                (*flag)=true;
            else
                (*flag)=false;
        }
        return true;
    }else{
        write(head,i,j,flag);
    }
}

int count(string array) {//считает количество символов, которе нужно
заменить

  unsigned int i, k=0;
  for (i=0; i<array.length(); i++)
    if (isalpha(array[i])) k++;
  return k;
}

void destroy (lisp s) {
    if ( s != NULL) {
        if (s->tag == false){
            destroy ( s->node.pair.hd);
            destroy ( s->node.pair.tl);
        }
        delete s;
    };
}
```

```
Pairs pairs(int n, string str, Pairs pp[],int &Err) {//создает пару
переменная-константа
  int ptr=2, i=0;
  while (str[ptr] != '\0') {
      if (!isspace(str[ptr]) && str[ptr] != '(' && str[ptr] != ')'){
          pp[i].var=str[ptr];
          pp[i].con=str[ptr+2];
          ptr+=3;
          i+=1;
      }
    ptr+=1;
  }
  if (n>i){
      cerr<<"Мало пар."<<endl;
      Err = 1;
  }
  return *pp;
}

string comp_pair(int n, string array, Pairs pp[]){//заменяет
переменную на число в строке

  int i, j=0;
  for (i=0; i<n; i++) {
    while (array[2*i]!=pp[j].var)
      j++;
    array[2*i]=pp[j].con;
    j = 0;
  }
  return array;
}

void process(string array, string str) {
  stack* Stack = new stack;
  Stack->topIndex = -1;
  for (int i=0;i<100;i++)
  Stack->arr[i] = 0;
  Stack->flag = true;
  lisp head=NULL;
  bool flag;
  flag=false;
  int ptr=0,i=0,j=0,Err=0;
  Pairs pp[26];
  char sym[100];
  char sygn[100];
  int n=0,k=0;
  int len = array.size()-1;
  int number = count(array);
  *pp = pairs(number, str, pp,Err);
  for(int i =0; i < number; i++){
    cout<<"[" << pp[i].var << " " << pp[i].con << "]" << endl;
```

```
  }
  array = comp_pair(number, array, pp);
  for (int i=0;i<array.size();i++){
    if (isdigit(array[i])){
      push(Stack,array[i]-'0');
    }
    if (array[i] == '*'){
      mul(Stack);
    }
    if (array[i] == '/'){
      div(Stack);
    }
    if (array[i] == '+'){
      add(Stack);
    }
    if (array[i] == '-'){
      sub(Stack);
    }
  }
  for (int i=0;i<array.size();i++){
    if (isalpha(array[i]) || isdigit(array[i])){
      sym[n++]=array[i];
    }
    if (ispunct(array[i])){
      sygn[k++]=array[i];
    }
  }
  sym[n]='\0';
  sygn[k]='\0';
  int l=0,m=0;
  for (int i=0;i<array.size();i++){
    if (isalnum(array[i]) || ispunct(array[i])){
      if (l<n){
        array[i] = sym[l];
        l++;
      }
       else if (m<k){
        array[i]=sygn[k-1-m];
        m++;
      }

  }
}
  read_lisp(array,&head,&ptr);
  if(head!=NULL && ptr==array.length()+1)
      if(write(head,i,j,&flag) && flag==false)
          cout<< "Корректный ввод."<<endl;
  if(head!=NULL && ptr< array.length()+1){
      cerr<<"ЗНеверный ввод."<<endl;
      return;
    }
```

```
  float x = Stack->arr[Stack->topIndex];
  cout << "Ответ: " << x << endl;
  destroy(head);
}

int main(){
    char num=0;
    while(num != 'q'){
        cout << "Выберите дальнейшие действия и введите цифру:"<<endl;
        cout << "1. Ввести выражение вручную."<<endl;
        cout << "2. Считать выражение из файла test1.txt."<<endl;
        cout << "q. Завершить работу."<<endl;
        cin >> num;
        switch(num){
            case '1': {
                getchar();
                string array, str;
                getline(cin,array);
                getline(cin,str);
                process(array, str);
                break;
            }
            case '2': {
                getchar();
                string array, str;
                ifstream infile("test.txt");
                getline(infile,array);
                getline(infile,str);
                infile.close();
                cout << array <<endl <<str <<endl;
                process(array, str);
                break;
            }
            case 'q':
                return 0;
            default:
                cerr << "Проверьте введенные данные и повторите
попытку." << endl;
                break;
        }
    }
    return 0;
}
```