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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «АиСД»

Тема: Иерархический список и его обработка

Студент гр. 8304 _____ Порывай П.А

Преподаватель _____ Фирсов М.А

Санкт-Петербург

2019

Цель работы.

Подсчет суммы длин плеч бк заданного через иерархически спиисок(вариант 3)

Выполнение работы

Структура s_expr описывает груз или бинарное коромысло(tag = false, true).Чтобы не занимать дополнительную память в ней опимано объединение состоящее из груза и 2 указателей на коромысло(struct two_ptr).Через typedef на структуру заведен указатель lisp(Те при объявлении lisp bink, bink указатель на элемент типа s_expr).

Функция head возвращает указатель «головы» s_expr в случае если элемент типа s_expr не груз иначе выдает ошибку.

Функция tail возвращает указатель «хвоста» s_expr если элемент типа s_expr не груз иначе выдает ошибку

Функция cons(const lisp h, const lisp t) выделяет память под новый указатель на элемент типа s_expr через оператор new (это будет коромысло в моей задаче) при этом указателем головы этого элемента становится h,хвоста t.Возвращает указатель на созданный элемент

Функция make_atom выделяет память под указатель на элемент s_expr с отрицательным тегом(груз, длина) и возвращает этот указатель

Функция isAtom проверяет является ли элемент грузом или длиной(tag = false), возращает значение типа bool

Функция isNull проверяет является ли элемент нулевым указателем , возвращает значение типа bool

Функция read_lisp(lisp& y) принимает ссылку на тип lisp(обрабатывается как обычная переменная типа lisp) и читает иерархический список заданный определенным образом.На деле считывает пробелы и вызывает read_s_expr

Функция read_s_expr(char prev, lisp& y) в случае первого прочитанного символа(не скобки и не пробела)в теле условия else if (prev != '(') отправляет

этот символ обратно во входной поток считывает число, создает груз если встретился пробел или '(' вызывает read_seq

В Функции read_seq(lisp& y) создается после удачного выполнения в ней же read_s_expr(), read_seq() создается коромысло

Перегруженные функции void read_lisp(lisp& y, std::ifstream &data), void read_s_expr(char prev, lisp& y, std::ifstream &data), void read_seq(lisp& y, std::ifstream &data) аналогичны вышенаписанным за исключением того, что ввод в них из файла.

Функция write_lisp(const lisp x) печатает элемент на экране в случае если это атом(в нашем случае атомом может быть длина или груз), если нет вызывает write_seq(const lisp x)

Функция write_seq(const lisp x) в случае непустого указателя рекурсивно следующую последовательность write_lisp(head(x) , out) ,write_seq(tail(x) , out).

Перегруженные функции void write_lisp(const lisp x, std::ofstream& out), void write_seq(const lisp x, std::ofstream& out) аналогичны вышенаписанным только вводят информацию в файл

Функция sum рекурсивно проходит список и если элемент является длиной(первым элементом после '(' прибавляет его к общей переменной которая передается через указатель int *s,

Выводы.

Получены навыки работы со структурой данных «Иерархический список», рекурсивной обработкой её элементов.

Приложение А. Исходный код

```
#include<iostream>
#include<cstdlib>
#include<string>//getline()
#include<fstream>//ifstream()
//VAR 3 Считаю сумму длин всех плеч бк
//using namespace std;
typedef int base; // базовый тип элементов (атомов)

struct s_expr;
struct  two_ptr
{
      s_expr* hd;//не могу инициализировать
      s_expr* tl;//не могу инициализировать
};    //end two_ptr;

struct s_expr {
      bool tag; // true: atom, false: pair
      union
      {
            base atom;//не могу инициализировать
            two_ptr pair{ nullptr, nullptr };
      } node;           //end union node
};                //end s_expr

typedef s_expr* lisp;

lisp head(const lisp s);
lisp tail(const lisp s);
lisp cons(const lisp h, const lisp t);
lisp make_atom(int x);
bool isAtom(const lisp s);
bool isNull(const lisp s);
void read_lisp(lisp& y);                  // основная
void read_s_expr(char prev, lisp& y);
void read_seq(lisp& y);
void read_lisp(lisp& y, std::ifstream &data);
void read_s_expr(char prev, lisp& y, std::ifstream &data);
void read_seq(lisp& y, std::ifstream &data);

// функции вывода:
void write_lisp(const lisp x);           // основная
```

```
void write_seq(const lisp x);
void write_lisp(const lisp x, std::ofstream& out);
void write_seq(const lisp x, std::ofstream& out);

lisp head(const lisp s)
{// PreCondition: not null (s)
      if (s != nullptr)
            if (!isAtom(s))
                  return s->node.pair.hd;
            else {

                  std::cerr << "Error: Head(atom) \n"; exit(1);
            }
      else {

            std::cerr << "Error: Head(nil) \n";
            exit(1);
      }
}

lisp tail(const lisp s)
{// PreCondition: not null (s)
      if (s != nullptr)
            if (!isAtom(s))
                  return s->node.pair.tl;
      else {
                  std::cerr << "Error: Tail(atom) \n"; exit(1);
      }
      else {
            std::cerr << "Error: Tail(nil) \n";
            exit(1);
      }
}


lisp make_atom(int x)
{
      lisp s;
      s = new s_expr;
      s->tag = true;
      s->node.atom = x;
      return s;
```

```
}

bool isNull(const lisp s)
{
      return s == nullptr;
}

bool isAtom(const lisp s)
{
      if (s == nullptr)
            return false;
      else
            return (s->tag);
}


lisp cons(const lisp h, const lisp t)
// PreCondition: not isAtom (t)
{
      lisp p;
      if (isAtom(t)) {
            std::cerr << "Error: Tail(nil) \n"; exit(1);
      }
      else {
                  p = new s_expr;
                  p->tag = false;
                  p->node.pair.hd = h;
                  p->node.pair.tl = t;
                  return p;

      }
}
void read_lisp(lisp& y)
{
      char x;
      do
            std::cin >> x;
      while (x == ' ');
      read_s_expr(x, y);
} //end read_lisp
//.........................
```

```
void read_s_expr(char prev, lisp& y)
{ //prev - ранее прочитанный символ}
      if (prev == ')') {

            std::cerr << " ! List.Error 1\n"; exit(1);

      }
      else if (prev != '(') {

            std::cin.putback(prev);
            int digit;
            std::cin >> digit;
            y = make_atom(digit);
      }
      else read_seq(y);
}

void read_seq(lisp& y)
{
      char x;
      lisp p1, p2;

      if (!(std::cin >> x)) {

            std::cerr << " ! List.Error 2 \n"; exit(1);
      }
      else {
            while (x == ' ')
                  std::cin >> x;
            if (x == ')')
                  y = nullptr;
            else {

                  read_s_expr(x, p1);
                  read_seq(p2);
                  y = cons(p1, p2);

            }
      }
} //end read_seq

void read_lisp(lisp& y,std::ifstream &data)
```

```
{
	char x;
	do
		data >> x;
	while (x == ' ');
	//cout << "1";
	read_s_expr(x, y,data);

} //end read_lisp
//.........................
void read_s_expr(char prev, lisp& y, std::ifstream &data)
{ //prev - ранее прочитанный символ}
	//cout << "2";
	if (prev == ')') {

		std::cerr << " ! List.Error 1 \n"; exit(1);
	}
	else if (prev != '(') {
		data.putback(prev);
		int digit;
		data >> digit;
		y = make_atom(digit);

	}
	else read_seq(y , data);
} //end read_s_expr

void read_seq(lisp& y, std::ifstream &data)
{
	//cout << "3";
	char x;
	lisp p1, p2;

	if (!(data >> x)) {
		std::cerr << " ! List.Error 2 \n"; exit(1);
	}
	else {
		while (x == ' ')
			data >> x;
		if (x == ')')
			y = nullptr;
		else {
```

```
                //cout << x << endl;
                read_s_expr(x, p1,data);
                read_seq(p2,data);
                y = cons(p1, p2);
            }
      }
} //end read_seq
//.........................
// Процедура вывода списка с обрамляющими его скобками - write_lisp,
//  а без обрамляющих скобок - write_seq
void write_lisp(const lisp x)
{//пустой список выводится как ()
      if (isNull(x))
            std::cout << " ()";
      else if (isAtom(x))
            std::cout << ' ' << x->node.atom;
      else { //непустой список}
            std::cout << " (";
            write_seq(x);
            std::cout << " )";
      }
} // end write_lisp
//.........................
void write_seq(const lisp x)
{//выводит последовательность элементов списка без обрамляющих его скобок
      if (!isNull(x)) {
            write_lisp(head(x));
            write_seq(tail(x));
      }
}

void write_lisp(const lisp x,std::ofstream &out)
{//пустой список выводится как ()
      if (isNull(x))
            out << " ()";
      else if (isAtom(x))
            out << ' ' << x->node.atom;
      else { //непустой список}
            out << " (";
            write_seq(x , out);
            out << " )";
      }
```

```
} // end write_lisp
//.........................
void write_seq(const lisp x, std::ofstream &out)
{//выводит последовательность элементов списка без обрамляющих его скобок
	if (!isNull(x)) {
		write_lisp(head(x) , out);
		write_seq(tail(x) , out);
	}
}

void sum(lisp p, int* s) {

	if (isAtom(p))
		* s += p->node.atom;
	else if (p->node.pair.hd != nullptr) {

		sum(p->node.pair.hd, s);

		if (p->node.pair.tl != nullptr && !isAtom(p->node.pair.tl->node.pair.hd))
			sum(p->node.pair.tl, s);
	}

}

int main(int argc, char* argv[]) {

	setlocale(LC_ALL, "Russian");
	std::cout << "Ввод из файла или из консоли? (f , c)\nОба пункта? (fc)\n";
	std::string arg;
	std::getline(std::cin , arg);

	if (arg == "f") {

		std::ifstream data(argv[1]);
		std::ofstream fout("out.txt");//Через linux не пашет

		if (data) {

			lisp bin_k;
			read_lisp(bin_k, data);
```

```
            while (bin_k->tag == false) {
                  int total = 0;

                  write_lisp(bin_k, fout);
                  write_lisp(bin_k);
                  sum(bin_k, &total);
                  std::cout << " sum = " << total << "\n";
                  fout << " sum = ";
                  fout << total;
                  fout << "\n";
                  read_lisp(bin_k, data);

            }
      }
      else
            std::cout << "Неверное имя файла\n";

      data.close();
      fout.close();
}
else if (arg == "c") {

      lisp bin_k;
      read_lisp(bin_k);

      std::ofstream fout("out.txt");
      while (bin_k->tag == false) {

            int total = 0;

            write_lisp(bin_k, fout);
            write_lisp(bin_k);

            sum(bin_k, &total);

            std::cout << " sum = " << total<<"\n";
            fout << " sum = ";
            fout << total;
            fout << "\n";
            read_lisp(bin_k);
```

```
	}

	fout.close();
}
else if (arg == "fc") {

	lisp bin_k;

	std::ifstream data(argv[1]);
	std::ofstream fout1("out1.txt");
	if (data) {

		lisp bin_k;
		read_lisp(bin_k, data);

		std::cout << "File:\n";
		while (bin_k->tag == false) {
			int total = 0;

			write_lisp(bin_k, fout1);
			write_lisp(bin_k);

			sum(bin_k, &total);

			std::cout << " sum = " << total <<"\n";
			fout1 << " sum = ";
			fout1 << total;
			fout1 << "\n";
			read_lisp(bin_k, data);


		}
	}
	else
		std::cout << "Неверное имя файла\n";

	std::cout << "Consol:\n";

	read_lisp(bin_k);

	std::ofstream fout2("out2.txt");
```

```
            while (bin_k->tag == false) {

                  int total = 0;

                  write_lisp(bin_k, fout2);
                  write_lisp(bin_k);

                  sum(bin_k, &total);

                  std::cout << " sum = " << total << "\n";
                  fout2 << " sum = ";
                  fout2 << total;
                  fout2 << "\n";
                  read_lisp(bin_k);


            }

            data.close();
            fout1.close();
            fout2.close();
      }
      return 0;
}
```

ПРИЛОЖЕНИЕ В
ТЕСТЫ

Ввод

((2 3)(7 89))

((12((14 15)(16 17)))(22 13))

((11((12 13)(21((121 11)(111 11)))))(12((567 789)(334 112))))

((1223112 1)(1 2))

Вывод

((2 3)(7 89)) sum = 9

((12((14 15)(16 17)))(22 13)) sum = 64

((11((12 13)(21((121 11)(111 11)))))(12((567 789)(334 112)))) sum = 1189

((13112 1)(1 2)) sum = 13113