МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №5

по дисциплине: Алгоритмы и структуры данных
тема: «Структуры данных «Линейные списки» (С)»

Выполнил: ст. группы ПВ-202
Буйвало Анастасия Андреевна

Проверил:
Кабалянц Петр Степанович
Маньшин Илья Михайлович

Белгород 2021 г.

Лабораторная работа №5

«Сравнительный анализ алгоритмов поиска (С)»

Цель работы:

Изучить СД типы «линейный список», научиться их программно реализовывать и использовать.

Задание:

1. Для СД типа «линейный список» определить:
   1.1. Абстрактный уровень представления СД:
      1.1.1. Характер организованности и изменчивости.
      1.1.2. Набор допустимых операций.
   1.2. Физический уровень представления СД:
      1.2.1. Схему хранения.
      1.2.2. Объем памяти, занимаемый экземпляром СД.
      1.2.3. Формат внутреннего представления СД и способ его интерпретации.
      1.2.4. Характеристику допустимых значений.
      1.2.5. Тип доступа к элементам.
   1.3. Логический уровень представления СД.
      a. Способ описания СД и экземпляра СД на языке программирования.
2. Реализовать СД типа «линейный список» в соответствии с вариантом индивидуального задания в виде модуля.
3. Разработать программу для решения задачи в соответствии с вариантом индивидуального задания с использованием модуля, полученного в результате выполнения пункта задания 2.

7	7	7

7. Многочлен $P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ с целыми коэффициентами можно представить в виде списка, причем если $a_i = 0$, то соответствующее звено не включать в список. Определить процедуру, которая стоит многочлен p — сумму многочленов q и r;

Выполнение работы:

СД «линейный список»:

Абстрактный уровень представления СД:

1. Характер организованности – последовательность;
2. Изменчивость – динамическая СД;

Набор допустимых операций: инициализация, включение элемента, исключение элемента, чтение текущего элемента, переход в начало списка, переход в конец списка, переход к следующему элементу, переход к i-му элементу, определение длины списка, уничтожение списка.

Физический уровень представления СД:

1. Схема хранения: последовательная или связная.
2. Объем памяти, занимаемый экземпляром СД: в зависимости от реализации.
3. Формат внутреннего представления СД(ПЛС) и способ его интерпретации:

Массив структур (одна структура M байт)	Индекс текущего	Длина списка	Размер массива (N)
M*N+4+4			

N – размер массива
M – размер структуры

4. Характеристика допустимых значений: в зависимости от реализации
5. Тип доступа к элементам: в зависимости от реализации

Логический уровень представления СД:

Способ описания СД (ПЛС 4 поля) и экземпляра СД на языке С:
Дескриптор ПЛС состоит из 4-х полей:
1 — указатель на массив, на основе которого реализуется ПЛС;
2 — количество элементов массива;
3 — индекс текущего элемента;
4 — длина ПЛС.

Способ описания СД:

```
Typedef struct List {TMemList* PMemList;

ptrel ptr;
unsigned int N; // длина списка
unsigned int SizeMem;};// размер массива
```

Способ описания экземпляра: `struct List L;`

Задание 2:

(__LIST7_H.h)

```
//
// Created by настя буйвало on 09/10/2021.
//

#if !defined(__LIST7_H)

#include <stdlib.h>
#include <stdio.h>
#define Index 1000

typedef struct monom {
    size_t power;
    int coef;
};
typedef struct monom basetype;
typedef basetype TMemList[Index];
typedef unsigned ptrel;
typedef struct List {
    TMemList* PMemList;
    ptrel ptr;//
    unsigned int N; // длина списка
    unsigned int SizeMem;
};// размер массива


//выделение памяти под массив типа basetype
void get_mem(struct List *L,unsigned SizeMem);

//инициализация списка
void InitList(struct List *L,unsigned SizeMem);

//включение элемента в список.
void PutList(struct List *L, basetype E);

//исключение элемента из списка.
void GetList(struct List *L, basetype *E);

//чтение элемента списка.
void ReadList(struct List *L,basetype *E);

//проверка: свободен ли список.
int FullList(struct List *L);

//проверка: является ли элемент последним.
int EndList(struct List *L);

//возвращает количество элементов в списке.
unsigned Count(struct List *L);

//устанановка в начало списка.
void BeginPtr(struct List *L);

//устанановка в конец списка.
void EndPtr(struct List *L);
```

```
//переход к следующему элементу.
void MovePtr(struct List *L);

//переход к n-му элементу.
void MoveTo(struct List *L, unsigned int n);

//удаление списка.
void DoneList(struct List *L);

//копирование списка L1 в список L2.
void CopyList(struct List *L1,struct List *L2);

#endif
```

(__LIST7_H.c)

```
//
// Created by настя буйвало on 10/10/2021.
//

#include "__LIST7_H.h"

const ListOk = 0;
const ListNotMem = 1;
const ListUnder = 2;
const ListEnd = 3;

short ListError = ListOk;

//выделение памяти под массив типа basetype
void get_mem(struct List *L,unsigned SizeMem)
{
    L->PMemList = (basetype*)malloc(SizeMem * sizeof(basetype));
};

//инициализация списка
void InitList(struct List *L,unsigned SizeMem)
{
    get_mem(L, SizeMem);
    L->SizeMem = SizeMem;
    L->N = 0;
    L->ptr = 0;
};

//возвращает количество элементов в списке.
unsigned Count(struct List *L)
{
    return(L->N);
};

//переход к следующему элементу.
void MovePtr(struct List *L)
{
    if(L->ptr + 1 < L->SizeMem)
        (L->ptr)++;
    else
        ListError = ListUnder;
};

//проверка: свободен ли список.
```

```
int FullList(struct List *L)
{
    return(Count(L) < L->SizeMem);
};

//включение элемента в список.
void PutList(struct List *L, basetype E)
{
    if(FullList(L)) {
        L->N++;
        L->PMemList[L->ptr]->coef = E.coef;
        L->PMemList[L->ptr]->power = E.power;
        MovePtr(L);
    }
    else
        ListError = ListUnder;

};

//чтение элемента списка.
void ReadList(struct List *L,basetype *E)
{
    E->coef = L->PMemList[L->ptr]->coef;
    E->power = L->PMemList[L->ptr]->power;
};

//исключение элемента из списка.
void GetList(struct List *L, basetype *E)
{
    ReadList(L, E);
    L->N--;
    L->ptr--;
};

//проверка: является ли элемент последним.
int EndList(struct List *L)
{
    return(L->ptr == Count(L));
};

//устанановка в начало списка.
void BeginPtr(struct List *L)
{
    L->ptr = 0;
};

//устанановка в конец списка.
void EndPtr(struct List *L)
{
    L->ptr = (L->N)-1;
};

//переход к n-му элементу.
void MoveTo(struct List *L, unsigned int n)
{
    if(L->ptr + 1 < L->SizeMem)
        L->ptr = n;
    else
        ListError = ListUnder;
};

//удаление списка.
```

```
void DoneList(struct List *L)
{
    free(L->PMemList);
    L->N = 0;
    L->ptr = 0;
};

//копирование списка L1 в список L2.
void CopyList(struct List *L1,struct List *L2)
{
    if(L2->SizeMem >= L1->N) {
        for (int i = 0; i < L1->N; i++){
            BeginPtr(L1);
            BeginPtr(L2);
            PutList(L2, *L1->PMemList[L1->ptr]);
            MovePtr(L1);
        }
    }
    else
        ListError = ListNotMem;
};
```

Задание 3:

(task.h)

```
//
// Created by настя буйвало on 10/10/2021.
//

#ifndef AISD5_TASK_H
#define AISD5_TASK_H

#include "__LIST7_H.h"

//ввод многочлена
void input_pol(struct List *L);

//ввод одночлена
void input_monom(basetype* L);

//запись суммы многочленов s1 и s2 в s3
void summ_polinoms(struct List *L1,struct List *L2, struct List *L3);

//вывод многочлена
void output_pol(struct List *L);

//вывод одночлена
void output_monom(basetype L);

#endif //AISD5_TASK_H
```

(task.c)

```
//
// Created by настя буйвало on 10/10/2021.
//

#include "task.h"

//ввод одночлена
```

```
void input_monom(basetype* L)
{
    printf("power = ");
    scanf("%d", &(L->power));
    printf(" coef = ");
    scanf("%d", &(L->coef));
}

//ввод многочлена
void input_pol(struct List *L)
{
    int n = 0;
    scanf("%d", &n);
    for(int i = 0; i < n; i++){
        input_monom(L->PMemList[L->ptr]);
        MovePtr(L);
        (L->N)++;
    }
}

//запись суммы многочленов s1 и s2 в s3
void summ_polinoms(struct List *L1,struct List *L2, struct List *L3)
{
    basetype T;
    BeginPtr(L1);
    BeginPtr(L2);
    while(FullList(L3) && (L1->ptr < Count(L1) || L2->ptr < Count(L2))){
        if((int)(L2->PMemList[L2->ptr]->power) == (int)(L1->PMemList[L1->ptr]-
>power)){
            T.coef = L2->PMemList[L2->ptr]->coef + L1->PMemList[L1->ptr]->coef;
            T.power = L2->PMemList[L2->ptr]->power;
            PutList(L3, T);
            MovePtr(L2);
            MovePtr(L1);
        }
        else{
            if(((int)(L2->PMemList[L2->ptr]->power) && ((int)(L2->PMemList[L2->ptr]-
>power) < (int)(L1->PMemList[L1->ptr]->power)) || (((int)(L2->PMemList[L2->ptr]-
>power) > (int)(L1->PMemList[L1->ptr]->power)) && !((int)(L1->PMemList[L1->ptr]-
>power))))) {
                PutList(L3, *L2->PMemList[L2->ptr]);
                MovePtr(L2);
            }
            else {
                PutList(L3, *L1->PMemList[L1->ptr]);
                MovePtr(L1);
            }
        }
    }
}

//вывод одночлена
void output_monom(basetype L)
{
    printf("%d*X^%d", L.coef, L.power);
}

//вывод многочлена
void output_pol(struct List *L)
{
    BeginPtr(L);
    for(int i = 0; i < Count(L); i++){
```

```
        output_monom(*L->PMemList[L->ptr]);
        MovePtr(L);
        printf(" + ");
    }
    printf("\b\b");
}
```

(main.c)

```
#include <stdio.h>
#include "task.h"

#define N 10

int main()
{
    struct List L1;
    struct List L2;
    InitList(&L1, N);
    InitList(&L2, N);

    printf("введите первый многочлен\n");
    input_pol(&L1);

    printf("введите второй многочлен\n");
    input_pol(&L2);

    struct List Lrez;
    InitList(&Lrez, N);
    summ_polinoms(&L1, &L2, &Lrez);

    output_pol(&Lrez);

    return 0;
}
```

Вывод: были изучены виды СД «линейный список», реализован формат соответствующего СД ПЛС (с 4 полями) варианта (7), с помощью модуля выполнено задание варианта.