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» (Pascal/C) »

Выполнил: ст. группы ПВ-211

Чувилко Илья Романович

Проверил:

Синюк Василий Григорьевич

Белгород 2022 г.

Цель работы: изучить СД типа «линейный список», научиться их программно реализовывать и использовать.

Задание

1. Для СД типа «линейный список» определить:
    1. Абстрактный уровень представления СД:
        1. Характер организованности и изменчивости.
        2. Набор допустимых операций.
    2. Физический уровень представления СД:
        1. Схему хранения.
        2. Объем памяти, занимаемый экземпляром СД.
        3. Формат внутреннего представления СД и способ его и интерпретации.
        4. Характеристику допустимых значений.
        5. Тип доступа к элементам.
    3. Логический уровень представления СД. Способ описания СД и экземпляра СД на языке программирования.
2. Реализовать СД типа «линейный список» в соответствии с вариантом индивидуального задания (см. табл.14) в виде модуля.
3. Разработать программу для решения задачи в соответствии с вариантом индивидуального задания (см. табл.14) с использованием модуля, полученного в результате выполнения пункта 2 задания.

Вариант 23. Модуль 7. Задача 1.

Модуль 7: ПЛС. Массив, на основе которого реализуется ПЛС, находится в динамической памяти (базовый тип элемента определяется задачей). Память под массив выделяется при инициализации ПЛС и количество элементов сохраняется в дескрипторе.

Реализация на языке С:

Содержимое заголовочного файла:

```
#ifndef CODE_LIST7_H
#define CODE_LIST7_H
#define capacity 1000

#include <malloc.h>
#include <stdlib.h>
#include <stdbool.h>

/// переменные ошибок
// успешное завершение операции
```

```
static const short LIST_OK = 0;
// недостаток памяти
static const short LIST_NOT_MEM = 1;
// указатель в конце списка
static const short ListUnder = 2;
static const short ListEnd = 3;
static short ListError;

typedef int BaseType;
typedef unsigned ptrel;
struct List {
  BaseType *PMemList;
  ptrel ptr;
  unsigned int N; // длина списка
  unsigned int SizeMem; // размер массива
} typedef List;

void InitList(List *L, unsigned SizeMem);

void PutList(List *L, BaseType x);

void GetList(List *L, BaseType x);

void ReadList(List *L, BaseType *x);

bool  FullList(List *L);

bool  EndList(List *L);

unsigned int Count(List *L);

void BeginPtr(List *L);

void EndPtr(List *L);

void MovePtr(List *L);

void MoveTo(List *L, unsigned int n);

void CopyList(List *L1,List *L2);

bool  isEqual(List *L1,List *L2);

void CopyFromArray(List *L, const int *a, int n);

#endif //CODE_LIST7_H
```

Содержимое файла реализации:

```
#include "List7.h"

void InitList(List *L, unsigned SizeMem) {
  L->PMemList = malloc(sizeof(BaseType) * SizeMem);
  L->SizeMem = SizeMem;
  L->N = capacity;
```

```
  L->ptr = 0;
}

void PutList(List *L, BaseType x) {
  if (FullList(L)) {
    ListError = LIST_NOT_MEM;
    return;
  }
  L->PMemList[++(L->ptr)] = x;
  L->N++;
}

void GetList(List *L, BaseType x) {
  bool  isFind = false;
  for (int i = 0; i < L->SizeMem; i++) {
    if (isFind) {
      L->PMemList[i - 1] = L->PMemList[i];
    } else if (L->PMemList[i] == x) {
      isFind = true;
      (L->N)--;
    }
  }
}

bool  FullList(List *L) {
  return L->N == capacity;
}

bool  EndList(List *L) {
  return L->N == capacity  - 1;
}

unsigned int Count(List *L) {
  return L->N;
}

void BeginPtr(List *L) {
  L->ptr = 0;
}

void EndPtr(List *L) {
  L->ptr = L->N;
}

void MovePtr(List *L) {
  (L->ptr)++;
}

void MoveTo(List *L, unsigned int n) {
  L->ptr = n;
}

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

void CopyList(List *L1, List *L2) {
```

```
  L2->PMemList = L1->PMemList;
  L2->SizeMem = L1->SizeMem;
  L2->N = L1->N;
  L2->ptr = L1->ptr;
}
```

Задача 1: Многочлен P(x)=anx n + an-1x n-1 + ... + a1x + a0 с целыми коэффициентами можно представить в виде списка, причем если ai = 0, то соответствующее звено не включать в список. Определить логическую функцию РАВНО(p, q), проверяющие на равенство многочлены p и q.

Для этого необходимо было реализовать две дополнительные функции:

```
// Возвращает true, если список L1 == L2. False в ином случае.
bool  isEqual(List *L1, List *L2) {
  if (L1->N != L2->N)
    return false;

  for (int i = 0; i < L1->N; i++) {
    if (L1->PMemList[i] != L2->PMemList[i])
      return false;
  }
  return true;
}

// Записывает в список L значения из массива a размера n
void CopyFromArray(List *L, int const *a, int const n) {
  for (int i = 0; i < n; i++)
    L->PMemList[i] = a[i];
  L->N = n;
}
```

Тесты:

```
#include <stdio.h>
#include "List7/List7.h"

void test1() {
  int a1[7] = {3, 5, 7, 1, 0, 5};
  List l1;
  InitList(&l1, 6);
  CopyFromArray(&l1, a1, 6);

  int a2[6] = {3, 5, 7, 1, 0, 5};
  List l2;
  InitList(&l2, 6);
  CopyFromArray(&l2, a2, 6);

  if (isEqual(&l1, &l2) == 1)
    printf("First test passed\n");
  else
    printf("First test failed\n");
}

void test2() {
  int a1[7] = {3, 5, 7, 1, 0, 5, 3};
  List l1;
  InitList(&l1, 7);
  CopyFromArray(&l1, a1, 7);
```

```
  int a2[6] = {3, 5, 7, 1, 0, 5};
  List l2;
  InitList(&l2, 6);
  CopyFromArray(&l2, a2, 6);

  if (isEqual(&l1, &l2) == 0)
    printf("Second test passed\n");
  else
    printf("Second test failed\n");
}

void test3() {
  int a1[6] = {3, 5, 7, 1, 1, 5};
  List l1;
  InitList(&l1, 6);
  CopyFromArray(&l1, a1, 6);

  int a2[6] = {3, 5, 7, 1, 0, 5};
  List l2;
  InitList(&l2, 6);
  CopyFromArray(&l2, a2, 6);

  if (isEqual(&l1, &l2) == 0)
    printf("Third test passed\n");
  else
    printf("Third test failed\n");
}

int main() {
  test1();
  test2();
  test3();

  return 0;
}
```

Результат тестов:

```
"D:\BGTU\ASD\Lab 5\code\cmake-build-debug\code.exe"
First test passed
Second test passed
Third test passed

Process finished with exit code 0
```

Вывод: в ходе лабораторной работы была изучена СД типа «линейный список», научились программно реализовывать и использовать такую СД.