МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

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

Дисциплина: Алгоритмы и структуры данных

Лабораторная работа № 5

**Структуры данных «линейные списки» (Pascal/С)**

|  |  |
| --- | --- |
|  | Выполнила: ст. группы ПВ-21  Зановская А.И.  Проверил: Синюк В.Г. |

Белгород

2017

Л а б о р а т о р н а я р а б о т а № 5

**Структуры данных «линейные списки» (Pascal/С)**

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

З а д а н и е

1. Для СД типа «линейный список» определить:

1.1. Абстрактный уровень представления СД:

1.1.1. Характер организованности и изменчивости.

Динамическая структура

1.1.2. 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.

1.2. Физический уровень представления СД:

1.2.1. Схему хранения.

Связная.

1.2.2. Объем памяти, занимаемый экземпляром СД.

4 байта.

1.2.3. Формат внутреннего представления СД и способ его интерпретации.

1.2.4. Характеристику допустимых значений.

Зависит от базового типа.

1.2.5. Тип доступа к элементам.

Последовательный.

1.3. Логический уровень представления СД.

Способ описания СД и экземпляра СД на языке программирования.

typedef struct element \*ptrel;

typedef struct element

{

BaseType data;

ptrel next;

} element;

typedef struct

{

ptrel Start;

ptrel ptr;

int N;

} List;

2. Реализовать СД «линейный список» в соответствии с вариантом индивидуального задания (см. табл.14) в виде модулей на языках Pascal и С.

Таблица 14

**Варианты индивидуальных заданий**

|  |  |  |
| --- | --- | --- |
| Номер варианта | Номер модуля | Задача |
| 10 | 2 | 10 |

Задача

10. Проверить, удовлетворяют ли элементы списка (базовый тип integer) закону x = f(х0,h), где х — элемент списка,h — шаг, x0 — начальный элемент списка.

Пример: x0 = 5, h = 1. x1 = 6, x2 = 7, x3 = 8... Элементы списка удовлетворяют закону х = f(5,1).

2. ОЛС в динамической памяти (базовый тип определяется задачей). Выделение памяти под информационную часть элемента ОЛС и запись в нее значения осуществляется при выполнении процедуры PutList. При выполнении процедуры GetList память, занимаемая элементом, освобождается.

#include <stdio.h>

#include <stdlib.h>

const short ListOk=0;

const short ListNotMem=1;

const short ListEmpty=2;

const short ListError;

typedef int BaseType;

typedef struct element \*ptrel;

typedef struct element

{

BaseType data;

ptrel next;

} element;

typedef struct

{

ptrel Start;

ptrel ptr;

} List;

//Инициализация списка

void InitList(List \*L)

{

L->Start = (ptrel)malloc(sizeof(element));

if (L->Start==NULL)

{

ListError=ListNotMem;

return;

}

L->ptr=L->Start;

L->ptr->next=NULL;

ListError=ListOk;

}

//Вставка в список

void PutList(List \*L, BaseType E)

{

ptrel tmp = (ptrel)malloc(sizeof(element));

tmp->data=E;

tmp->next=L->ptr->next;

L->ptr->next=tmp;

L->ptr=L->ptr->next;

}

//Взятие из списка

void GetList(List \*L, BaseType \*E)

{

ptrel tmp;

if (EmptyList(L))

{

return;

}

tmp=L->ptr->next;

L->ptr->next=tmp->next;

\*E = tmp->data;

free(tmp);

}

//Чтение из списка

void ReadList(List \*L, BaseType \*E)

{

if (EmptyList(L))

return;

\*E=L->ptr->data;

}

//Предикат - список пуст

int EmptyList(List \*L)

{

if (L->Start->next == NULL)

{

ListError = ListEmpty;

return 1;

}

return 0;

}

//Предикат - конец списка

int EndList(List \*L)

{

if (L->ptr->next == NULL)

return 1;

return 0;

}

//Колличество элементов в списке

unsigned int Count(List \*L)

{

unsigned int counter = 0;

ptrel tmp = L->Start->next;

while (tmp->next != NULL)

{

counter++;

tmp=tmp->next;

}

return counter;

}

//Перемещение текущего указателя в начало списка

void BeginPtr(List \*L)

{

L->ptr=L->Start;

}

//Перемещение текущего указателя на один эелемент вперёд

void MovePtr(List \*L)

{

L->ptr=L->ptr->next;

}

//Перемещение текущего указателя на позицию n

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

{

unsigned int counter=0;

ptrel tmp = L->Start;

while(counter++!=n)

tmp = tmp->next;

L->ptr=tmp;

}

//Очистка спиcка

void ClearList(List \*L)

{

ptrel tmp, tmpnext;

tmp = L->Start;

tmpnext=tmp->next;

while (EmptyList(L))

{

tmp=tmpnext;

tmpnext=tmpnext->next;

free(tmp);

}

free(L->Start);

ListError=ListEmpty;

}

//Копирование первого списка во второй

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

{

ptrel tmp = L1->Start;

while (tmp->next!=NULL)

{

PutList(L2, tmp->next->data);

tmp=tmp->next;

}

}

3. Разработать программы на языках Pascal и С для решения задачи в соответствии с вариантом индивидуального задания (см. табл.14) с использованием модуля, полученного в результате выполнения пункта 2 задания.

void InputList(List \*L)

{

int i;

BaseType E;

InitList(L);

printf ("i = ");

scanf ("%d",&i);

while(i--!=0)

{

scanf ("%d",&E);

PutList(L,E);

L->N+=1;

}

}

int check (List \*L)

{

int fl=1,t,f;

BaseType E,x,T;

BeginPtr(L);

MovePtr(L);

if (EmptyList(L))

return 0;

ReadList(L,&x);

if (EndList(L))

return 1;

MovePtr(L);

ReadList(L,&T);

f=T-x;

while (fl&&!EndList(L))

{

MovePtr(L);

ReadList(L,&E);

fl=(f==E-T);

T=E;

}

return fl;

}

void output (List \*L)

{

BeginPtr(L);

MovePtr(L);

while (L->ptr!=NULL)

{

printf ("%d ", L->ptr->data);

MovePtr(L);

}

}

void main ()

{

List \*L;

L=(struct List\*)malloc(sizeof ( List));

InputList (L);

output(L);

printf ("\n%d",check (L));

}