C Лабораторная работа 5 Структуры

*структуры, стеки, очереди, списки, деревья, многоразрядные числа*.

**Контрольные вопросы**

1. Что такое структура? Для чего в программировании используются

структуры?

2. Являются ли следующие утверждения верными?

1) Структуры могут содержать только один тип данных.

2) Наличие псевдонима, определенного с помощью оператора **typedef**,

является для структур обязательным.

3) Члены одной структуры должны иметь уникальные имена.

4) Структуры могут передаваться функциям только по значению.

5) Динамические структуры стек и очередь являются частным видом

списков.

3. Как можно проинициализировать структуру? Можно ли присваивать

одной структуре другую? Можно ли их сравнивать?

4. Как можно обратиться к конкретному члену структуры? Как это сделать с

помощью указателя на структуру?

5. Чем массив структур отличается от структуры массивов? Приведите

примеры использования этих конструкций.

6. В чем заключаются преимущества и недостатки динамических структур

данных по сравнению с обычными?

7. Что такое самоссылочные структуры? В чем заключается их основное

преимущество? Какие наиболее распространенные самоссылочные

структуры вы знаете?

8. Как называется вид списка, в котором элементы могут вставляться и

удаляться только из конца? Вид списка, в котором элементы вставляются

в конец, а удаляются из начала?

9. В чем заключается разница между однонаправленным и

двунаправленным списком, между линейным и кольцевым списком?

10. Как наиболее быстро развернуть направление кольцевого списка?

11. Что такое дерево? Чем оно отличается от списков?

12. Что такое бинарное дерево поиска? В чем заключается его основное

преимущество?

13. Какое максимальное число узлов может содержать бинарное дерево,

имеющее три уровня?

14. Объявлена следующая структура:

struct list { int value; list \*next; };

Что делают следующие функции?

int F1(list \*p)

{

int n;

for (n = 0; p != NULL; p = p->next, n++) ;

return n;

}

int F2(list \*p, int n)

{

for (; n != 0 && p != NULL; n--, p = p->next) ;

return p->value;

}

15. Какие ошибки содержатся в следующих объявлениях структур?

1) structure {

char itable;

int num[20]

char \*togs

}

2) struct person {

char lastName[15];

char firstName[15];

int age;

}

3) struct key {

int number;

};

key d;

4) struct key {

int number;

} \*ptr;

ptr->number = 128;

5) struct card {

char \*face;

char \*suit;

} \*ptr;

printf(″%s″, \*ptr->face);

printf(″%s″, \*ptr.suit);

**Примеры решения задач**

**Пример 5.1.** Разработать функцию, суммирующую два многоразрядных

числа. Для хранения многоразрядного числа использовать динамический

двунаправленный список.

#include <stdio.h>

#include <alloc.h>

#include <string.h>

/\* Структура, описывающая элемент двунаправленного списка \*/

typedef struct item {

int digit;

struct item \*next;

struct item \*prev;

} Item;

/\* Структура, описывающая многоразрядное число \*/

typedef struct mnumber {

Item \*head;

Item \*tail;

int n;

} MNumber;

MNumber CreateMNumber(char \*initStr);

void AddDigit(MNumber \*number, int digit);

void PrintMNumber(MNumber number);

MNumber SumMNumber(MNumber n1, MNumber n2);

void main(void)

{

MNumber a = CreateMNumber("123456789123456789");

MNumber b = CreateMNumber("987654321987654321");

MNumber c = SumMNumber(a, b);

PrintMNumber(a);

PrintMNumber(b);

PrintMNumber(c);

}

/\* Создает многоразрядное число из цифр строки \*/

MNumber CreateMNumber(char initStr[])

{

MNumber number = {NULL, NULL, 0};

int n;

for (n = strlen(initStr)-1; n >= 0; n--)

AddDigit(&number, initStr[n]-'0');

return number;

}

/\* Добавляет цифру в многоразрядное число \*/

void AddDigit(MNumber \*number, int digit)

{

Item \*p = (Item \*)malloc(sizeof(Item));

p->digit = digit;

p->next = p->prev = NULL;

if (number->head == NULL)

number->head = number->tail = p;

else {

number->tail->next = p;

p->prev = number->tail;

number->tail = p;

}

number->n++;

}

/\* Возвращает сумму двух многоразрядных чисел \*/

MNumber SumMNumber(MNumber n1, MNumber n2)

{

MNumber sum = CreateMNumber("");

Item \*p1 = n1.head, \*p2 = n2.head;

int digit, pos = 0, s1, s2;

while (p1 || p2) {

if (p1) { s1 = p1->digit; p1 = p1->next; }

else s1 = 0;

if (p2) { s2 = p2->digit; p2 = p2->next; }

else s2 = 0;

digit = (s1 + s2 + pos) % 10;

pos = (s1 + s2 + pos) / 10;

AddDigit(&sum, digit);

}

if (pos) AddDigit(&sum, pos);

return sum;

}

/\* Выводит многоразрядное число на экран \*/

void PrintMNumber(MNumber number)

{

Item \*p = number.tail;

printf("\nNumber: ");

while (p) {

printf("%d", p->digit);

p = p->prev;

}

}

Структура **MNumber**, описывающая многоразрядное число, содержит

указатели на начало и конец динамического двунаправленного списка, в

котором будут храниться цифры числа, и количество цифр в нем – **n**.

В функции **main()** создаются два многоразрядных числа – **a** и **b**, их

сумма присваивается числу **с**. После этого все три числа выводятся на экран.

Функция **CreateMNumber()** принимает инициализирующую строку, в

которой записаны цифры числа. Вначале в функции создается «пустое»

многоразрядное число, а затем последовательно, элемент за элементом,

значения из строки копируются в двунаправленный список. Обратите

внимание, что строку мы копируем, начиная с конца: для арифметических

расчетов удобнее, чтобы список «рос» от младших разрядов к старшим.

В функции **SumMNumber()** происходит суммирование

многоразрядных чисел. Цикл **while (p1 || p2)** выполняется, пока не закончатся

разряды в обоих слагаемых. На каждой итерации вычисляется очередная

цифра суммирующего числа **digit** и «перенос» на следующий разряд **pos**.

Если после выполнения цикла появился перенос в самом старшем разряде, то

к сумме добавляется еще один (старший) разряд.

Функция **PrintMNumber()** выводит число в обратном порядке, от

«хвоста» к «голове» списка, т.к. мы добавляли цифры, начиная с конца числа.

В виде полезного упражнения модифицируйте программу, чтобы

пользователь мог сам инициализировать переменные **a** и **b**; разработайте

функцию **DeleteMNumber()**, удаляющую передаваемое ей многоразрядное

число.

**Пример 5.2.** В текстовом файле содержится набор чисел. С помощью

бинарного дерева поиска вывести числа на печать по возрастанию.

#include <stdio.h>

#include <stdlib.h>

#include <alloc.h>

/\* Структура, описывающая узел дерева \*/

typedef struct item {

int data;

struct item \*left;

struct item \*right;

} Item;

void AddNode(int data, Item \*\*node);

void LeftOrder(Item \*node);

void main(void)

{

char buffer[128];

int i;

Item \*root = NULL; /\* корень дерева \*/

FILE \*fp = fopen("ex52.txt", "r");

if (!fp) exit(1); /\* выйти, если файл не удалось открыть \*/

/\* Считываем значения из файла построчно \*/

while (fgets(buffer, 128, fp) != NULL)

AddNode(atoi(buffer), &root); /\* и добавляем в дерево \*/

LeftOrder(root);

fclose(fp);

}

/\* Добавить узел в бинарное дерево поиска \*/

void AddNode(int data, Item \*\*node)

{

if (\*node == NULL) {

\*node = (Item \*)calloc(1, sizeof(Item));

(\*node)->data = data;

(\*node)->left = (\*node)->right = NULL;

} else {

if (data < (\*node)->data)

AddNode(data, &(\*node)->left);

else if (data > (\*node)->data)

AddNode(data, &(\*node)->right);

else

puts("There is such element in the tree");

}

}

/\* Обход дерева слева (вывод по возрастанию) \*/

void LeftOrder(Item \*node)

{

if (node->left)

LeftOrder(node->left);

printf("%d ", node->data);

if (node->right)

LeftOrder(node->right);

}

В начале программы объявляется структура для работы с бинарным

деревом. В функции **main()** после открытия текстового файла значения в

цикле считываются в строку **buffer**, пока не будет достигнут конец файла. С

помощью стандартной функции **atoi()** цифры, записанные в строке,

преобразуются в целые значения, которые и добавляются в дерево.

Т.к. функция **AddNode()** строит бинарное дерево поиска, обход слева

этого дерева выводит значения по возрастанию (обход справа вывел бы их по

убыванию).

**Задачи**

**Требования к программам**

1. Код программы должен быть логично разбит на функции.

2. Работа программы должна точно удовлетворять условиям задачи.

3. Программа не должна содержать предупреждений компилятора.

4. Программа должна быть хорошо протестирована.

5. Программа должна иметь дружественный интерфейс.

6. Код программы должен быть выдержан в хорошем стиле.

**5.1.** С помощью двунаправленного списка определить, является ли

строка палиндромом (т.е. одинаково читается справа налево и слева

направо). Пробелы и знаки пунктуации – игнорировать.

**5. 7.** Разработать функцию, которая инвертирует заданный список, т.е.

первый элемент становится последним, второй – предпоследним и

т.д.

**5. 9.** Разработать функцию вычисления *n*!, при *n* > 100. Для хранения

многоразрядного числа использовать динамический

двунаправленный список.