

Лабораторная работа №2
Указатели, арифметика указателей.

Выполнил:

Студент 1-го курса

Группы ИВТ-1.1

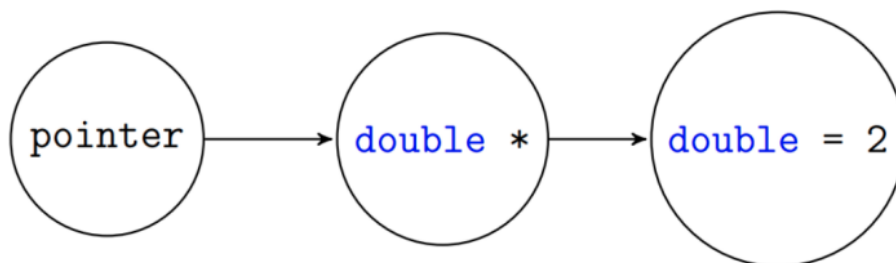
Шардт Максим Александрович

Оглавление

Комплект 1: Указатели, арифметика указателей	3
Задание 1.1	3
Задание 1.2	4
Задание 1.3	5
Задание 1.4	6
Задание 1.5	8
Задание 1.6	9
Задание 1.7	10

1. Комплект 1: Указатели, арифметика указателей

1.1. Внутри функции `int main(void) { /*... */ }` определите указатель `double **pointer = NULL;`. Инициализируйте этот указатель адресом другого указателя типа `double *`, который указывает, в свою очередь, на переменную `double`. Используйте `pointer` для записи и чтения в эту переменную значения 2.



При этом выполните следующее:

- используйте функции типа `*alloc(...)` для выделения оперативной памяти под динамические объекты;
- запишите и выведите число, указанное в крайней правой окружности, на экран, используя указатель `double **pointer = NULL;`;
- используйте функцию `free(...)` для освобождения оперативной памяти, выделенной под динамические объекты.

Математическая модель

Под переменную выделяется память размером 4 байта. Указатель на начало этой памяти записывается в переменную “`p`”. Далее создается второй указатель “`pp`”, указывающий на первый указатель. С помощью второго указателя в выделенную память записывается значение 2.

После вывода значений, выделенная память освобождается.

Список идентификаторов

<i>Имя переменной</i>	<i>Тип данных</i>	<i>Смысловое значение</i>
<code>p</code>	Указатель на Double	Указатель на выделенную память
<code>pp</code>	Указатель на Double	Указатель на <code>p</code> (<code>**pointer</code> из задания)

Код программы

Листинг 1: Lab2 - 1-1.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void)
5 {
6     double *p = NULL;
7     double **pp = NULL;
8
9     p = (double *)malloc(sizeof(double));
10    pp = &p;
11    **pp = 2;
12
13    printf("Ptr to ptr: %p\n", pp);
14    printf("Ptr: %p\n", *pp);
15    printf("Value: %lf\n", **pp);
16
17    free(p);
18
19    return 0;
20 }
```

Результат выполненной работы

```
Ptr to ptr: 0x7ffe9f703da8
Ptr: 0x55bce25242a0
Value: 2.000000
```

Выводится 3 значения: первого указателя, второго указателя и содержимого в памяти.

1.2. Напишите программу, которая складывает два числа с использованием указателей на эти числа.

Математическая модель

Под две переменные выделяется память по 4 байта. С помощью указателей на эти участки памяти переменные складываются.

Список идентификаторов

Имя переменной	Тип данных	Смысловое значение
----------------	------------	--------------------

pointer1	Указатель на Int	Указатель на выделенную память 1
pointer2	Указатель на Int	Указатель на выделенную память 2

Код программы

Листинг 2: Lab2 - 1-2.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void)
5  {
6      int *pointer1 = (int *)malloc(sizeof(int));
7      int *pointer2 = (int *)malloc(sizeof(int));
8      *pointer1 = 2;
9      *pointer2 = 4;
10
11     printf("Value: %d\n", *pointer1 + *pointer2);
12     free(pointer1);
13     free(pointer2);
14
15     return 0;
16 }

```

Результат выполненной работы

Value: 6

Выводится единственная строка: сумма переменных.

1.3. Напишите программу, которая находит максимальное число из двух чисел, используя указатели на эти числа.

Математическая модель

Выделяются два блока памяти по 4 байта. Пользователь вводит переменные “a” и “b”, результат сравнения этих переменных выводится на экран.

Список идентификаторов

Имя переменной	Тип данных	Смысловое значение
ptr1	Указатель на Int	Указатель на выделенную память 1

ptr2	Указатель на Int	Указатель на выделенную память 2
------	------------------	----------------------------------

Код программы

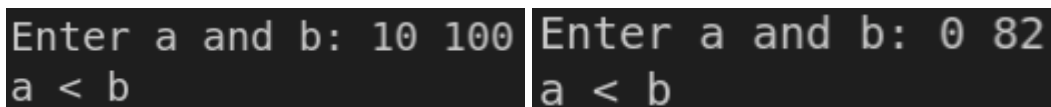
Листинг 3: Lab2 - 1-3.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void)
5  {
6      int *ptr1 = (int *)malloc(sizeof(int));
7      int *ptr2 = (int *)malloc(sizeof(int));
8
9      printf("Enter a and b: ");
10     scanf("%d", ptr1);
11     scanf("%d", ptr2);
12
13     if (*ptr1 > *ptr2)
14         printf("a > b\n");
15     else if (*ptr1 < *ptr2)
16         printf("a < b\n");
17     else
18         printf("a = b\n");
19     free(ptr1);
20     free(ptr2);
21     return 0;
22 }
23

```

Результат выполненной работы



```

Enter a and b: 10 100
a < b
Enter a and b: 0 82
a < b

```

1.4. Напишите программу, которая создаёт одномерный динамический массив из чисел с плавающей точкой двойной точности, заполняет его значениями с клавиатуры и распечатывает все элементы этого массива, используя арифметику указателей (оператор +), а не обычный оператор доступа к элементу массива - [].

Математическая модель

Создается массив, размер которого задается пользователем. С помощью арифметики указателей массив заполняется значениями, вводимыми в консоль.

Итоговый массив выводится на экран.

Список идентификаторов

Имя переменной	Тип данных	Смысловое значение
length	Integer	Размер массива
array	Double	Исходный массив
input	Double	Переменная для ввода значений
start	Double	Указатель на начало массива

Код программы

Листинг 4: Lab2 - 1-4.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void)
5  {
6      int length;
7      printf("Enter length of array: ");
8      scanf("%d", &length);
9
10     double *array = (double *)malloc(length * sizeof(int));
11
12     double input;
13     for (double *start = array; start < &array[length]; start++)
14     {
15         printf("[%ld]: ", start - array);
16         scanf("%lf", &input);
17
18         *start = input;
19     }
20     printf("\nArray: [ ");
21     for (double *start = array; start < &array[length]; start++)
22         printf("%lf, ", *start);
23     printf("]\n");
24
25     free(array);
26     return 0;
27 }
```

Результат выполненной работы

```
Enter length of array: 5
[0]: 1.1234
[1]: 2
[2]: 3.11
[3]: 4.0
[4]: 0.5

Array: [ 1.123400, 2.000000, 3.110000,
4.000000, 0.500000, ]
```

Вводится массив длиной в 5 элементов. Каждый из вводимых элементов нумеруется. Затем массив выводится на экран.

1.5. Вывести элементы динамического массива целых чисел в обратном порядке, используя указатель и операцию декремента ($--$).

Математическая модель

Создается массив, размер которого задается пользователем. С помощью арифметики указателей массив заполняется значениями, вводимыми в консоль. Итоговый массив выводится на экран в обратном порядке.

Список идентификаторов

Имя переменной	Тип данных	Смысловое значение
length	Integer	Размер массива
array	Double	Исходный массив
input	Double	Переменная для ввода значений
start	Double	Указатель на начало массива

Код программы

Листинг 5: Lab2 - 1-5.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void)
5 {
6     int length;
7     printf("Enter length of array: ");
8     scanf("%d", &length);
9
10    double *array = (double *)malloc(length * sizeof(int));
11    double input;
12    for (double *start = array; start < &array[length]; start++)
13    {
14        printf("[%ld]: ", ((start - array)));
15        scanf("%lf", &input);
16
17        *start = input;
18    }
19    printf("\nArray: [");
20    for (double *start = &array[length - 1]; start >= array; start--)
21        printf("%lf, ", *start);
22    printf("]\n");
23
24    free(array);
25    return 0;
26 }

```

Результат выполненной работы

```

Enter length of array: 5
[0]: 1
[1]: 2
[2]: 3
[3]: 4
[4]: 5

Array: [5.000000, 4.000000, 3.000000, 2.000000,
1.000000, ]

```

Вводится массив длиной в 5 элементов. Каждый из вводимых элементов нумеруется. Затем массив выводится на экран в обратном порядке.

1.6. Определите переменную целого типа `int a = 1234567890`; и выведите побайтово её содержимое на экран, используя указатель `char *`.

Математическая модель

1234567890 в памяти занимает 4 байта и будет представляться в бинарной системе счисления как 01001001 10010110 00000010 11010010 или 49 96 02 D2 в шестнадцатеричной.

Список идентификаторов

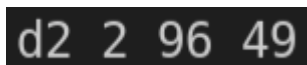
Имя переменной	Тип данных	Смысловое значение
a	Integer	Изначальное число
start	Char	Указатель для побайтового вывода числа “a”

Код программы

Листинг 6: Lab2 - 1-6.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int a = 1234567890;
6
7     for (char *start = (char *)&a; start < (char *)&a + sizeof(int);
8         start++)
9         printf("%hhx ", *start);
10    printf("\n");
11    return 0;
12 }
```

Результат выполненной работы



Выводится 4 значения. Так как программа компилировалась и запускалась на процессоре архитектуры x86, в котором используется обратный порядок байтов (little-endian), ожидаемые значения выведены в обратном порядке.

1.7. Выделите память под двумерный динамический массив, используя массив указателей на строки (см. лекции). Затем освободите корректно оперативную память.

Математическая модель

Матрица будет состоять из указателя на массив строк (m), каждый элемент которого будет содержать указатель на массив столбцов (n).

$$A_{m \times n} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mj} & \dots & a_{mn} \end{pmatrix}$$

Список идентификаторов

Имя переменной	Тип данных	Смысловое значение
matrix	Указатель на Integer	Указатель на матрицу
n_rows	Integer	Количество строк
n_columns	Integer	Количество столбцов

Код программы

Листинг 7: Lab2 - 1-6.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 int main(void)
6 {
7     srand(time(0));
8
9     int **matrix = NULL;
10    int n_rows = 3;
11    int n_columns = 3;
12
13    matrix = (int **)malloc(n_rows * sizeof(int));
14
15    for (int i = 0; i < n_rows; i++)
16        matrix[i] = (int *)malloc(n_columns * sizeof(int));
17
```

```

18     for (int i = 0; i < n_rows; i++)
19         for (int j = 0; j < n_columns; j++)
20             matrix[i][j] = rand() % 100;
21
22     printf("Matrix:\n");
23     for (int i = 0; i < n_rows; i++)
24     {
25         for (int j = 0; j < n_columns; j++)
26             printf("%d ", matrix[i][j]);
27         printf("\n");
28     }
29
30     for (int i = 0; i < n_rows; i++)
31         free(matrix[i]);
32     free(matrix);
33     return 0;
34 }

```

Результат выполненной работы

```

Matrix:
72 24 73
13 52 54
72 79 92

```