Указатели, арифметика указателей

Задача 1

Постановка задачи

Внутри функции int main(void) { /.../ } определите указатель double ***pointer = NULL; Инициализируйте этот указатель адресом другого указателя типа double **, который указывает на переменную double *, которая указывает на double. Используйте pointer для записи и чтения значения 2.0 в сегмент оперативной памяти для double.

Список идентификаторов

Имя переменной	Тип данных	Описание
pointer	double ***	Указатель на указатель на указатель на double

Код программы

```c
#include <stdio.h>
#include <stdlib.h>


int main(void) {
    double ***pointer = NULL;
    (pointer = (double ***)malloc(sizeof(double **)), *pointer = (double
**)malloc(sizeof(double *)), **pointer = (double
*)malloc(sizeof(double)), ***pointer = 2.0, printf("\n%.1f\n\n",
***pointer), free(**pointer), free(*pointer), free(pointer));
}
```

Результат работы программы

```
2.0
```

Задача 2

Постановка задачи

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

Список идентификаторов

Имя переменной	Тип данных	Описание
a, b	int	Слагаемые
*pa, *pb	int *	Указатели на int

Код программы

```c
#include <stdio.h>
#include <stdlib.h>


int main(void) {
    int a = 10;
    int b = 20;
    int *pa = &a;
    int *pb = &b;

    printf("\n%d + %d = %d\n\n", *pa, *pb, *pa + *pb);
}
```

Результат работы программы

```
10 + 20 = 30
```

Задача 3

Постановка задачи

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

Список идентификаторов

Имя переменной	Тип данных	Описание
a , b	int	Сравниваемые числа
*a, *b	int *	Указатели на a и b
max	int	Функция max()

Код программы

```c
#include <stdio.h>
#include <stdlib.h>


int max(int *a, int *b) {
    if (a == NULL || b == NULL)
        return EXIT_FAILURE;
    return *a > *b ? *a : *b;
}

int main(void) {
    int a = 100;
    int b = 20;

    printf("\nmax(%d, %d) = %d\n\n", a, b, max(&a, &b));
}
```

Результат работы программы

```
max(100, 20) = 100
```

```
max(1005, 20000) = 20000
```

Задача 4

Постановка задачи

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 оператор доступа к элементу массива [].

Список идентификаторов

Имя переменной	Тип данных	Описание
*pa	double *	Указатель на массив double
size	int	Размер массива

Код программы

```c
#include <stdio.h>
#include <stdlib.h>


int main(void) {
    double *pa = NULL;
    int size;

    // Вводим длину массива
    printf("\nInput size of array, pls: ");
    scanf("%d", &size);

    // Проверим доступность оперативки
    pa = (double *)malloc(sizeof(double) * size);
    if (pa == NULL) {
        printf("Ошибочка :(\n\n");
        return EXIT_FAILURE;
    }

    // Ввод массива
    printf("\nInput elements of array, pls.\n");
    for (int i = 0; i < size; i++) {
        printf("a[%d] : ", i);
        scanf("%lf", pa + i);
    }
```

```
    // Вывод массива
    printf("\nHere is your array :)\n[");
    for (int i = 0; i < size; i++)
        printf(i == size - 1 ? "%.2f]\n\n" : "%.2f, ", *(pa + i));

    // Не забываем освободить память
    free(pa);

    return 0;
}
```

Результат работы программы

```
Input size of array, pls: 4

Input elements of array, pls.
a[0] : 2.0
a[1] : 0
a[2] : 0.55
a[3] : 785.9

Here is your array :)
[2.00, 0.00, 0.55, 785.90]
```

```
Input size of array, pls: 24141111115555555555555555555555322222
Ошибочка :(
```

Задача 5

Постановка задачи

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

Список идентификаторов

Имя переменной	Тип данных	Описание
*pa	int *	Указатель на массив int
size	int	Размер массива
*p	int *	Счётчик-указатель, убывает, перебирая элементы с конца массива
buf	int	Буфер для замены местами элементов массива

Код программы

```c
#include <stdio.h>
#include <stdlib.h>


int main(void) {
    int *pa = NULL;
    int size;

    // Вводим длину массива
    printf("\nInput size of array, pls: ");
    scanf("%d", &size);

    // Проверим доступность оперативки
    pa = (int *)malloc(sizeof(int) * size);
    if (pa == NULL) {
        printf("Ошибочка :(\n\n");
        return EXIT_FAILURE;
    }

    // Ввод массива
    printf("\nInput elements of array, pls.\n");
    for (int i = 0; i < size; i++) {
        printf("a[%d] : ", i);
        scanf("%d", pa + i);
```

```
    }

    // Инверсия массива
    int *p = pa + size - 1;
    int buf;
    for (int i = 0; i < size / 2; i++, p--) {
        buf = *p;
        *p = *(pa + i);
        *(pa + i) = buf;
    }

    // Вывод массива
    printf("\nHere is your array, but inverted :)\n[");
    for (int i = 0; i < size; i++)
        printf(i == size - 1 ? "%d]\n\n" : "%d, ", *(pa + i));

    // Не забываем освободить память
    free(pa);

    return 0;
}
```

Результат работы программы

```
Input size of array, pls: 5

Input elements of array, pls.
a[0] : 1
a[1] : 2
a[2] : 3
a[3] : 4
a[4] : 5

Here is your array, but inverted :)
[5, 4, 3, 2, 1]
```

Задача 6

Постановка задачи

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

Список идентификаторов

Имя переменной	Тип данных	Описание
a	int	Переменная целого типа
*pa	unsigned char *	Указатель на первый байт переменной a
*p	unsigned char *	Счётчик-указатель, возрастает, перебирая байты

Код программы

```c
#include <stdio.h>
#include <stdlib.h>


int main(void) {
    int a = 1234567890;
    unsigned char *pa = (unsigned char *)&a;

    for (unsigned char *p = pa; p < pa + sizeof(a); p++)
        printf("%d\n", *p);

    return 0;
}
// Использую unsigned char, так как с char выводятся некорректные
значения.
```

Результат работы программы

```
210
2
150
73
```

Задача 7

Постановка задачи

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

Список идентификаторов

Имя переменной	Тип данных	Описание
**mx	int **	Указатель на массив указателей на элементы двумерного массива
h, w	int	Высота и ширина массива (кол-во строк и столбцов)
r	int	Счётчик для перебора строк

Код программы

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>


int main(void) {
    int **mx = NULL;
    int h = 4;
    int w = 3;

    srand(time(0));

    // Выделим память
    mx = (int **)malloc(sizeof(int *) * h);
    if (mx == NULL) {
        printf("Ошибочка: не удалось выделить память для строк :(\n\n");
        return EXIT_FAILURE;
    }

    for (int r = 0; r < h; r++) {
        *(mx + r) = (int *)malloc(sizeof(int) * w);

        if (mx[r] == NULL) {
            printf("Ошибочка: не удалось выделить память для столбца :
(\n");
```

```
            // Нужно освободить всё, что уже успели занять
            for (int j = 0; j < r; j++) {
                free (mx[j]);
            }
            free (mx);
            return EXIT_FAILURE;
        }
    }

    // Заполним и сразу выведем матрицу
    printf("\n\n");
    for (int r = 0; r < h; r++) {
        for (int c = 0; c < w; c++) {
            mx[r][c] = rand() % 10;
            printf("%d ", mx[r][c]);
        }
        printf("\n");
    }
    printf("\n");

    // Освободим память
    for (int r = 0; r < h; r++) {
        free(mx[r]);
    }
    free(mx);

    return 0;
}
```

Результат работы программы

```
4 9 8
8 9 5
0 8 6
3 5 6
```