

Массивы:

Одномерные массивы

```
#include <stdio.h>
int main() {
    // Массив из 10 чисел типа int. Без инициализации ( значения могут быть какими угодно )
    int a[10];
    // Можно считать, что мы создали сразу 10 переменных типа int под именами
    // a[0], a[1], a[2], ... a[9]

    // Считаем числа в массив с помощью scanf
    for (int i = 0; i < 10; i++)
        scanf("%d", &a[i]);
    // Если вы знаете, что будет храниться в массиве, то можно сразу его инициализировать
    int b[10] = {4, 8, 15, 16, 23, 42}; // оставшиеся 4 числа будут установлены нулями
}
```

1. **Объявление:** Объявить следующие массивы и напечатайте их адрес и размер:
 - массив из 10 элементов типа char
 - массив из 20 элементов типа float
 - массив из 30 элементов типа unsigned long long
2. **Инициализация:** Объявить массив под названием days_in_month с 12 элементами типа int и инициализировать его следующими значениями: 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31. Отдельной операцией изменить второе значение с массиве с 28 на 29.
3. **Печать:** Напечатать содержимое массива days_in_month на экран. Числа должны быть напечатаны в одну строку через пробел.

Передача массивов в функцию

```
// Массивы в функцию всегда передаются через указатель (int arr[] и int* arr одно и то же)
void print_array(int n, int* arr) {
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
// Следовательно, при изменении массива в функции, он меняется и вне функции
void square_array(int n, int arr[]) {
    for (int i = 0; i < n; i++)
        arr[i] = arr[i] * arr[i];
}
int main() {
    // Часто, мы не знаем размер массива заранее, поэтому берём с запасом
    // (1000 элементов) и работаем только с n первыми элементами
    int a[1000];
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
        scanf("%d", &a[i]);
    print_array(n, a);
    square_array(n, a);
    print_array(n, a);
}
```

4. **Сумма массива:** Написать функцию `int sum(int n, int* arr)`, которая будет возвращать сумму массива целых чисел. Входные параметры такие же как и предыдущей задаче. Функция не должна ничего печатать и считывать. Использовать эту функцию в функции `main()` чтобы найти сумму чисел массива `days_in_month`.
5. **Умножение массива:** Написать функцию `void multiply(int n, int* arr, int k)`, которая будет умножать все элементы массива на число `k`. Протестируйте эту функцию в `main()`.
6. **Индекс минимального элемента:** Написать функцию `int min_index(int n, int* arr)`, которая будет возвращать индекс наименьшего числа в массиве. Входные параметры такие же как и предыдущей задаче. Функция не должна ничего печатать и считывать. Если в массиве есть несколько минимальных элементов, то функция должна вывести индекс первого из них.
7. **Сортировка выбором:** Написать функцию `void selection_sort(int n, int* arr)`, которая будет сортировать массив методом выбора.
Алгоритм сортировки методом выбора для сортировки массива `[0:n]`:
 - Находим индекс минимального элемента
 - Переставляем местами 0-й элемент с минимальным
 - Повторяем то же самое для подмассива `[1:n]`Протестируйте работу функции на массиве из хотя бы 30-ти элементов.
8. **Сортировка выбором по убыванию:** Написать функцию `void selection_sort_descend(int n, int* arr)`, которая будет сортировать массив методом выбора по убыванию.
9. **Реккурсивная сортировка выбором:** Написать функцию `void selection_sort_rec(int lo, int hi, int* arr)`, которая будет сортировать часть массива от индекса `lo` до `hi` (не включая `hi`) методом выбора. Используйте рекурсию.

Вложенные циклы

```
for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++) {
        printf("%d : %d\n", i, j);
    }
```

10. **Таблица умножения:** Напечатать таблицу умножения с помощью вложенных циклов. Таблица должна быть квадратной, примерно такой, как печатают на тетрадках. Используйте модификатор `"%5d"`.

Двумерные циклы и массивы

```
// Зададим константу MAX = 100
#define MAX 100
void print_array(int n, int m, int arr[MAX][MAX]) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++)
            printf("%d ", arr[i][j]);
        printf("\n");
    }
}

int main() {
    // Создаём массивы с запасом
    int a[MAX][MAX] = {{7, 7, 2}, {1, 8, 3}, {2, 1, 6}};
    int b[MAX][MAX] = {{5, 2, 9}, {-4, 2, 11}, {7, 1, -5}};
    // Печатаем только 9 элементов
    print_array(3, 3, a);
}
```

11. **Умножение двумерного массива на число:** Написать функцию `void multiply_num(int n, int m, int arr[MAX][MAX], int x)`, которая будет умножать двумерный массив на число. Протестируйте работу функции в `main()`.
12. **Сложение двумерных массивов:** Написать функцию `void sum(int n, int m, int A[MAX][MAX], int B[MAX][MAX], int C[MAX][MAX])`, которая будет складывать двумерные массивы и записывать их в массив `C`. Протестируйте работу функции в `main()`.
13. **Перемножение двумерных массивов:** Написать функцию `void mult(int n, int m, int A[MAX][MAX], int B[MAX][MAX], int C[MAX][MAX])`, которая будет перемножать двумерные массивы и записывать их в массив `C`. Протестируйте работу функции в `main()`.
14. **N-я степень:** Написать функцию `void power(int n, int m, int A[MAX][MAX], int N, int C[MAX][MAX])`, которая будет возводить матрицу `A` в степень `N` и записывать результат в массив `C`. Протестируйте работу функции в `main()`.