Основы программирования

ФИСТ 1 курс
Власенко
Олег
Федосович

Лекция 9

Функции.
Глобальные и локальные переменные.
Передача параметров.
Возвращение результата.

Что такое подпрограмма?

Процедуры и функции в Pascal.

```
procedure ReadArray;
begin
end;

function Abs(x:single): single;
begin
end;
```

Функции в Си.

```
void read_array () {
}

float abs(float f) {
}
```

Зачем нужны подпрограммы?

Зачем нужны подпрограммы?

- **Писать меньше кода - Повторяющийся код реализовать один раз, а вызывать многократно (sin(), printf() …)**
- **Сделать код проще для редактирования - Разделить длинный код на части (произвольно)**
- **Упростить код - Разбить сложный алгоритм на части**
- **Повысить уровень абстракции – уйти от низкоуровневых операций на уровень предметной области**

- **Создавать библиотеки для повторного использования – стандартная библиотека Си состоит из функций**
- **Писать большие программы (до десятков и сотен тысяч строк кода)**

Поехали!

Задача 1: Ввести массив. Найти среднее арифметическое четных элементов. Те четные элементы, которые меньше среднего арифметического заменить на 0. Получившийся массив вывести.

Задача 1*: Ввести массив. Найти среднее арифметическое четных элементов. Найти минимум из четных элементов. Те четные элементы, которые меньше среднего арифметического заменить на минимальный из четных элементов. Получившийся массив вывести.

Реализация без функций

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

#define NUM 4

void main() {
	int arr[NUM];
	int i;
	int s;
	int cnt_even;
	double average_even;
```

Реализация без функций (2)

```
// Ввод элементов с клавиатуры
i = 0;
while (i < NUM) {
        scanf("%d", &arr[i]);
        i++;
}
```

Реализация без функций (3)

```
// Подсчет среднего арифметического четных элементов
        s = 0;
        cnt_even = 0;
        i = 0;
        while (i < NUM) {
                if (arr[i] % 2 == 0) {
                        s += arr[i];
                        cnt_even++;
                }
                i++;
        }
        average_even = s / cnt_even;
```

Реализация без функций (4)

```
// Четные элементы меньше среднего заменяем на 0
        i = 0;
        while (i < NUM) {
                if (arr[i] % 2 == 0 && arr[i] < average_even) {
                        arr[i] = 0;
                }
                i++;
        }
```

Реализация без функций (5)

```
// Вывод массива
i = 0;
while (i < NUM) {
        printf("%d ", arr[i]);
        i++;
}

}
```

Локальные и глобальные переменные

```
#define NUM 4

void main() {
// локальные переменные
        int arr[NUM];
        int i;
        int s;
        int cnt_even;
        double average_even;
```

```
#define NUM 4

// глобальные переменные
int arr[NUM];
int i;
int s;
int cnt_even;
double average_even;

void main() {
```

Имена функций (1)

```
// Ввод элементов с клавиатуры
// read_array
i = 0;
while (i < NUM) {
        scanf("%d", &arr[i]);
        i++;
}
```

Имена функций (2)

```
// Подсчет среднего арифметического четных элементов
        // get_average_even
        s = 0;
        cnt_even = 0;
        i = 0;
        while (i < NUM) {
                if (arr[i] % 2 == 0) {
                        s += arr[i];
                        cnt_even++;
                }
                i++;
        }
        average_even = s / cnt_even;
```

Имена функций (3)

```
// Четные элементы меньше среднего заменяем на 0
        // change_even_elements
        i = 0;
        while (i < NUM) {
                if (arr[i] % 2 == 0 && arr[i] < average_even) {
                        arr[i] = 0;
                }
                i++;
        }
```

Имена функций (4)

```
        // Вывод массива
        // write_array
        i = 0;
        while (i < NUM) {
                printf("%d ", arr[i]);
                i++;
        }

}
```

Итак – выносим код в функции (1)

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

#define NUM 4



int arr[NUM];
int i;
int s;
int cnt_even;
double average_even;
```

Итак – выносим код в функции (2)

```
void read_array() {
        i = 0;
        while (i < NUM) {
                scanf("%d", &arr[i]);
                i++;
        }
}
```

Итак – выносим код в функции (3)

```
void get_average_even() {
        s = 0;
        cnt_even = 0;
        i = 0;
        while (i < NUM) {
                if (arr[i] % 2 == 0) {
                        s += arr[i];
                        cnt_even++;
                }
                i++;
        }
        average_even = s / cnt_even;
}
```

Итак – выносим код в функции (4)

```
void change_even_elements() {
        i = 0;
        while (i < NUM) {
                if (arr[i] % 2 == 0 && arr[i] < average_even) {
                        arr[i] = 0;
                }
                i++;
        }
}
```

Итак – выносим код в функции (5)

```
void write_array() {
        i = 0;
        while (i < NUM) {
                printf("%d ", arr[i]);
                i++;
        }
}
```

Итак – выносим код в функции (6)

```
void main() {

        // Ввод элементов с клавиатуры
        read_array();

        // Подсчет среднего арифметического четных элементов
        get_average_even();

        // Четные элементы меньше среднего заменяем на 0
        change_even_elements();

        // Вывод массива
        write_array();

}
```

Плюсы и минусы этого решения

- Какие проблемы решены?
- Какие появились?
- Какие проблемы еще остались?

Идем дальше – минимум глобальных переменных

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

#define NUM 4



int arr[NUM];
double average_even;
```

минимум глобальных переменных (2)

```
void read_array() {
        int i = 0;
        while (i < NUM) {
                scanf("%d", &arr[i]);
                i++;
        }
}
```

минимум глобальных переменных (3)

```
void get_average_even() {
        int s = 0;
        int cnt_even = 0;
        int i = 0;
        while (i < NUM) {
                if (arr[i] % 2 == 0) {
                        s += arr[i];
                        cnt_even++;
                }
                i++;
        }
        average_even = s / cnt_even;
}
```

минимум глобальных переменных (4)

```
void change_even_elements() {
        int i = 0;
        while (i < NUM) {
                if (arr[i] % 2 == 0 && arr[i] < average_even) {
                        arr[i] = 0;
                }
                i++;
        }
}
```

минимум глобальных переменных (5)

```
void write_array() {
        int i = 0;
        while (i < NUM) {
                printf("%d ", arr[i]);
                i++;
        }
}
```

минимум глобальных переменных (6)

```
void main() {

        // Ввод элементов с клавиатуры
        read_array();

        // Подсчет среднего арифметического четных элементов
        get_average_even();

        // Четные элементы меньше среднего заменяем на 0
        change_even_elements();

        // Вывод массива
        write_array();

}
```

Полный отказ от глобальных переменных

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

#define NUM 4
```

Без глобальных переменных (2)

```
void read_array(int arr[NUM]) {
        int i = 0;
        while (i < NUM) {
                scanf("%d", &arr[i]);
                i++;
        }
}
```

Без глобальных переменных (3)

```
double get_average_even(int arr[NUM]) {

        int s = 0;
        int cnt_even = 0;
        int i = 0;
        while (i < NUM) {
                if (arr[i] % 2 == 0) {
                        s += arr[i];
                        cnt_even++;
                }
                i++;
        }

        return s / cnt_even;
}
```

Без глобальных переменных (4)

```
void change_even_elements(int arr[NUM],
                                        double average_even) {
        int i = 0;
        while (i < NUM) {
                if (arr[i] % 2 == 0 && arr[i] < average_even) {
                        arr[i] = 0;
                }
                i++;
        }
}
```

Без глобальных переменных (5)

```
void write_array(int arr[NUM]) {
        int i = 0;
        while (i < NUM) {
                printf("%d ", arr[i]);
                i++;
        }
}
```

Без глобальных переменных (6)

```
void main() {
        int arr[NUM];
        double average_even;

        // Ввод элементов с клавиатуры
        read_array(arr);

        // Подсчет среднего арифметического четных элементов
        average_even = get_average_even(arr);

        // Четные элементы меньше среднего заменяем на 0
        change_even_elements(arr, average_even);

        // Вывод массива
        write_array(arr);
}
```

Плюсы и минусы этого решения

- Какие проблемы решены?
- Какие появились?

Задача 2

Легенда:

Есть 4 группы – ПИбд, ИВТ…

Каждая группа содержит разное количество студентов.

Проведена контрольная работа по Физике.

Нужно выяснить в какой из групп лучшие результаты по Физике.

Постановка задачи:

Ввести 4 массива из не более чем 30 элементов в каждом.

Выяснить в каком из них среднее арифметическое элементов больше всего. Вывести номер этого массива (0, 1, 2 или 3). Для контроля вывести среднее арифметическое каждого из массивов.

Реализация

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

#define MAX_NUM 30

void read_array(int arr[], int n);
double get_average(int arr[], int n);
void write_array(double arr[], int n);
int get_max_index(double arr[], int n);

void main() {
```

Реализация (2)

```
int a0[MAX_NUM], a1[MAX_NUM];
int a2[MAX_NUM];
int a3[MAX_NUM];

int n0 = 4, n1 = 5, n2 = 6, n3 = 3;
double average[4];
int max_index;

// Ввод элементов с клавиатуры
read_array(a0, n0);
read_array(a1, n1);
read_array(a2, n2);
read_array(a3, n3);
```

Реализация (3)

```
// Подсчет среднего арифметического четных элементов
        average[0] = get_average(a0, n0);
        average[1] = get_average(a1, n1);
        average[2] = get_average(a2, n2);
        average[3] = get_average(a3, n3);

        max_index = get_max_index(average, 4);

// Вывод массива
        write_array(average, 4);
        printf("Max index = %d\n", max_index);
}
```

Реализация (4)

```
void read_array(int arr[], int n) {
        int i;

        printf("write %d elements:\n", n);

        i = 0;
        while (i < n) {
                scanf("%d", &arr[i]);
                i++;
        }
}
```

Реализация (5)

```
double get_average(int arr[], int n) {
        int s = 0;
        int i = 0;
        while (i < n) {
                s += arr[i];
                i++;
        }
        return ((double)s) / n;
}
```

Реализация (6)

```
void write_array(double arr[], int n) {
        int i = 0;
        while (i < n) {
                printf("%lf ", arr[i]);
                i++;
        }
        printf("\n");

}
```

Реализация (7)

```
int get_max_index(double arr[], int n) {
        int ind_max = 0;
        double max = arr[0];
        int i = 1;
        while (i < n) {
                if (arr[i] > max) {
                        max = arr[i];
                        ind_max = i;
                }
                i++;
        }
        return ind_max;
}
```

Домашнее задание

1. Собрать и запустить на компьютере все полные примеры
2. Переделать все примеры через for() {}

3. Переделать домашнюю работу от лекции 8 через функции.