1. Функции в C++

Функции нужны, чтобы разбить программу на логические блоки - подпрограммы, которые выполняют законченные задачи.

Например, можно описать функции расчёта заработной платы, сортировки массива, работы с пользовательским вводом и т.д.

Использование функций дает преимущества:

- Повторное использование кода. Одну функцию можно вызывать из разных частей программы
- Удобство вложенности. Функция может вызывать другие функции
- Локализация данных. Переменные в функции видны только внутри неё

Так код становится понятнее и легче для изменения.

2. Объявление функции

При объявлении указывается:

- тип возвращаемого значения (например, int, double)
- уникальное имя функции
- список параметров в скобках (могут отсутствовать)
- тело функции в фигурных скобках

Пример объявления:

```
double calcAvg(int a, int b) {
  return (a + b) / 2.0;
}
```

3. Вызов функции

Чтобы использовать функцию, нужно вызвать её по имени и передать аргументы:

```
int x = 5, y = 10;
double avg = calcAvg(x, y); // вызов функции
```

В avg будет сохранен результат работы функции calcAvg с аргументами x и y.

4. Перегрузка функций

Перегрузка функций позволяет использовать одно имя для разных вариантов функции. Например:

```
int sum(int a, int b) {
  // код
}

double sum(double a, double b) {
  // код
}
```

Здесь есть 2 функции sum, которые отличаются списком параметров.

5. Процедуры

Процедура в C++ - это разновидность функции, которая ничего не возвращает. Она нужна для выполнения некоторой последовательности действий.

Объявление процедуры выглядит так:

```
void printArray(int arr[], int n) {

  // цикл перебора элементов
  for(int i = 0; i < n; i++) {

    cout << arr[i] << " ";

  }

  cout << endl;

}
```

Здесь используется ключевое слово void вместо возвращаемого типа. И нет оператора return.

Вызов процедуры аналогичен функции:

```
printArray(myArray, 5);
```

6. Рекурсивные функции

Рекурсия подразумевает, что функция вызывает саму себя для решения подзадачи.

Например, рекурсивный расчёт факториала n:

```
int factorial(int n) {

  // условие завершения рекурсии
  if (n <= 0)
    return 1;

  // рекурсивный вызов:
  // 5! = 5 * 4!
  // 4! = 4 * 3!
  return n * factorial(n - 1);
}
```

Таким образом решение строится на повторном применении того же алгоритма для данных меньшего размера.

7. Пример программы

```cpp
// Подключаем библиотеку ввода-вывода
#include <iostream>
using namespace std;

// Объявляем функцию расчёта факториала
int factorial(int x) {
  int result = 1;

  for(int i = 1; i <= x; i++) {
    result *= i;
  }

  return result;
}

int main() {

  int num; // переменная для числа

  // Ввод числа
  cout << "Введите число: ";
  cin >> num;

  // Вызываем функцию factorial() и передаём число
  int fac = factorial(num);

  // Выводим результат
  cout << "Факториал числа: " << fac;

  return 0;
}
```

Описание работы:

Здесь мы объявили функцию factorial(), которая считает факториал переданного числа. Эту функцию мы вызываем в main(), передав туда пользовательский ввод, и выводим результат вычислений на экран.

Таким образом функции позволяют инкапсулировать код для решения подзадач.

8. Номера задач по теме

- Задача №306. Минимум 4 чисел
- Задача №307. Степень
- Задача №308. Исключающее ИЛИ
- Задача №309. Голосование
- Задача №310. Проверка на простоту
- Задача №252. Степень для отрицательного показателя
- Задача №311. Быстрое возведение в степень
- Задача №312. Числа Фибоначчи
-
- Задача №113652. Получить из 1 число N
- Задача №113653. Наибольшая цифра
- Задача №113654. Количество цифр
- Задача №113655. Вставить звёздочки
- Задача №113656. Расставить скобки
- Задача №113657. Сформировать новую строку со скобками
- Задача №113658. Сократить буквы
- Задача №113659. Форум. Удаление ветви

Сайт с задачами: https://informatics.msk.ru