Лекция: **Функции, Подпрограммы и Рекурсия в C++**

1. **Введение**

Функции и подпрограммы являются основными строительными блоками программы на C++, позволяющими разбивать задачи на отдельные модули. Рекурсия — это одна из техник, используемая для решения задач с помощью функций, когда функция вызывает саму себя.

2. **Функции в C++**

Функция — это независимый блок кода, который можно вызывать в программе по мере необходимости. Это помогает сделать программу более читаемой и повторно используемой. В C++ функции могут возвращать значение и принимать параметры.

2.1 Синтаксис функции

Общий синтаксис функции в C++:

```
возвращаемый_тип имя_функции(параметры) {
    // тело функции
    return значение; // (опционально)
}
```

- **Возвращаемый тип** — указывает, какой тип данных функция возвращает. Если функция ничего не возвращает, используется `void`.
- **Имя функции** — идентификатор, через который можно вызвать функцию.
- **Параметры** — значения, передаваемые в функцию при её вызове.
- **Тело функции** — код, который выполняется при вызове функции.

2.2 Пример простой функции

```
#include <iostream>
using namespace std;

int add(int a, int b) {
    return a + b;
}

int main() {
    int result = add(5, 3);
    cout << "Сумма: " << result << endl;
    return 0;
}
```

В этом примере функция `add` принимает два целых числа, складывает их и возвращает результат.

2.3 Параметры функции

Функции могут принимать аргументы двумя способами:

- **Передача по значению**: копия значения передаётся в функцию.
- **Передача по ссылке**: функция получает доступ к исходным данным и может их изменять.

Пример передачи по ссылке:

```
void increment(int &x) {
    x = x + 1;
}
```

Здесь параметр `x` передаётся по ссылке, и любые изменения в функции затронут оригинальную переменную.

3. **Подпрограммы**

Подпрограммы в C++ — это функции, которые выполняют специфические задачи и могут вызываться из других функций. Основное преимущество подпрограмм — это возможность повторного использования кода и логического разделения программы.

3.1 Пример подпрограммы

Подпрограммы, по сути, ничем не отличаются от обычных функций в C++, однако в контексте крупных программ их часто используют для модульного решения задач.

```
#include <iostream>
using namespace std;

void greet() {
    cout << "Привет, мир!" << endl;
}

int main() {
    greet(); // Вызов подпрограммы
    return 0;
}
```

Функция `greet` — это подпрограмма, которую можно вызывать в любом месте программы для вывода сообщения.

4. **Рекурсия**

Рекурсия — это метод, при котором функция вызывает саму себя для решения подзадачи. Важно, чтобы у каждой рекурсивной функции был базовый случай, который завершает рекурсию.

4.1 Пример рекурсивной функции

Факториал числа можно вычислить как:

- `n! = n * (n-1) * (n-2) * ... * 1`
- Базовый случай: `0! = 1`

```
int factorial(int n) {
    if (n == 0) {
        return 1;  // Базовый случай
    } else {
        return n * factorial(n - 1);  // Рекурсивный случай
    }
}
```

4.2 Работа рекурсии

Каждый вызов рекурсивной функции откладывается до тех пор, пока не будет достигнут базовый случай. После этого начинается сворачивание стека вызовов и возврат значений обратно.

Пример вычисления факториала 3:

1. `factorial(3)` вызывает `factorial(2)`
2. `factorial(2)` вызывает `factorial(1)`
3. `factorial(1)` вызывает `factorial(0)`
4. Базовый случай: `factorial(0) = 1`
5. Сворачивание: `factorial(1) = 1 * 1 = 1`
6. Сворачивание: `factorial(2) = 2 * 1 = 2`
7. Сворачивание: `factorial(3) = 3 * 2 = 6`

4.3 Типичные ошибки при использовании рекурсии

1. **Отсутствие базового случая** — если не определить базовый случай, рекурсия станет бесконечной и программа завершится с ошибкой.
2. **Неправильное уменьшение параметров** — если не изменить параметры в рекурсивных вызовах, функция также не завершится.

5. Примеры использования рекурсии

5.1 Числа Фибоначчи

Числа Фибоначчи можно вычислить рекурсивно. Числа Фибоначчи определяются как:

- `F(0) = 0`
- `F(1) = 1`
- `F(n) = F(n-1) + F(n-2)` для `n > 1`

```
int fibonacci(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
```

```
    return fibonacci(n-1) + fibonacci(n-2);
}
```

5.2 Рекурсивный обход файловой системы

Рекурсия также часто используется для задач обхода дерева (например, файловой системы), когда одна директория может содержать вложенные директории и файлы, требующие аналогичной обработки.

6. **Заключение**

Функции и подпрограммы в C++ позволяют организовывать код, делая его более структурированным и повторно используемым. Рекурсия — мощный инструмент для решения задач, где решение одной части задачи зависит от решения более мелких её частей. Однако необходимо осторожно использовать рекурсию, чтобы избежать ошибок, таких как бесконечная рекурсия или неправильное уменьшение аргументов.