

# ПОДПРОГРАММЫ



**СГУГиТ**  
СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

Р. В. Гришин  
mail-to: [r.grishin54@gmail.com](mailto:r.grishin54@gmail.com)

# СОДЕРЖАНИЕ ЛЕКЦИИ

- ☐ Введение
- ☐ Основы подпрограмм
- ☐ Среды локальных ссылок
- ☐ Методы передачи параметров
- ☐ Перегруженные подпрограммы
- ☐ Определяемые пользователем перегруженные операторы
- ☐ Замыкания
- ☐ Корутины



**СГУиТ**

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

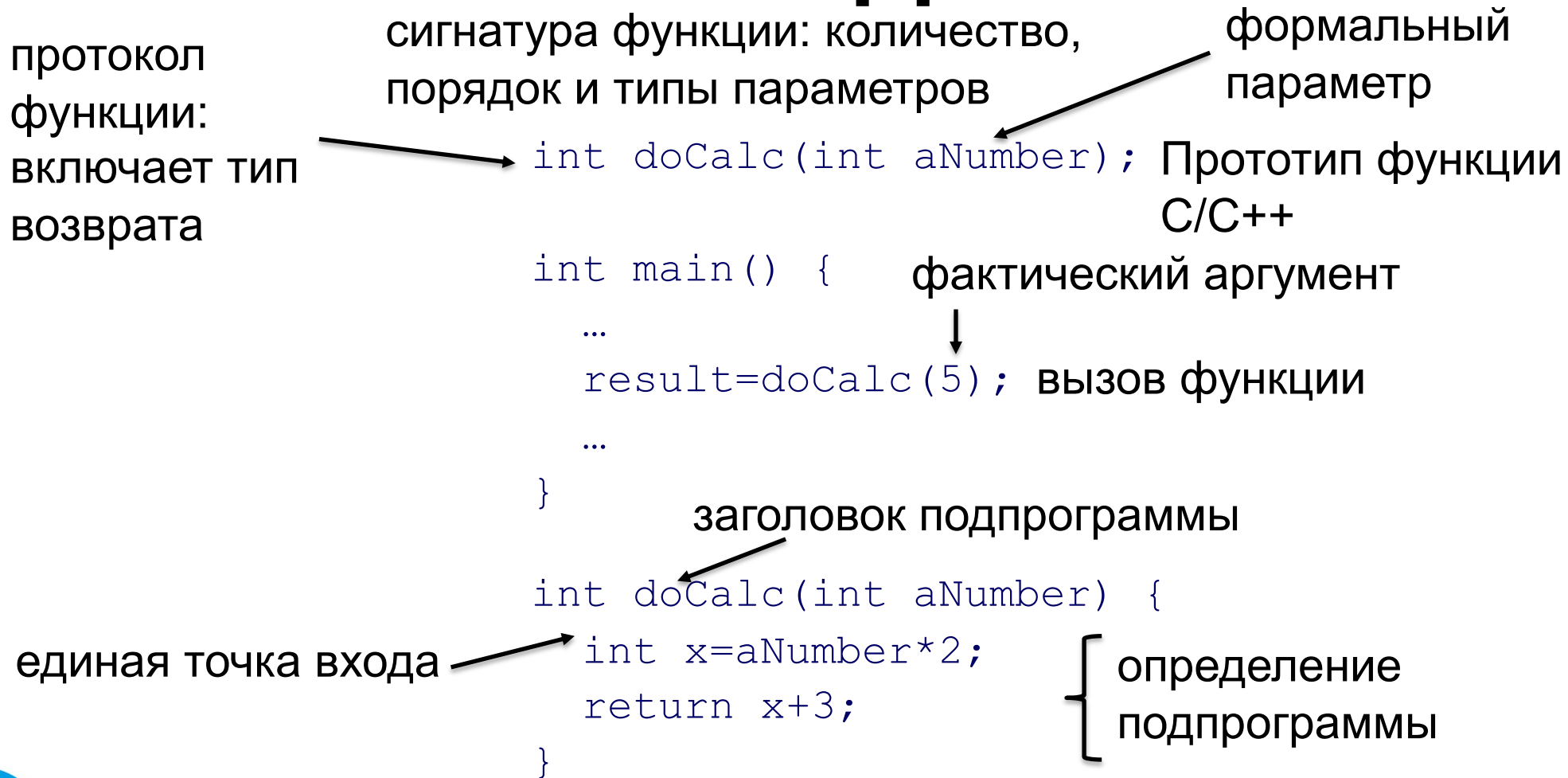
# ВВЕДЕНИЕ

## Два фундаментальных средства абстракции:

- ☐ Абстракция процессов - подчеркивается с первых дней существования, детали того, как выполняется вычисление или процесс, "абстрагируются" вызовом процедуры
- ☐ Абстракция данных - подчеркнуто в 1980-х годах



# ОСНОВЫ ПОДПРОГРАММ




СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

# ОСНОВЫ ПОДПРОГРАММ

## Позиционные параметры

```
int doIt(int a, double b);  
  
// in main  
doIt(5, 4.6);
```



## Именованные параметры

```
summer(length=my_length,  
        list=my_array,  
        sum=my_sum)
```

## Совмещенное использование

```
summer(my_length,  
        sum=my_sum,  
        list=my_array)
```

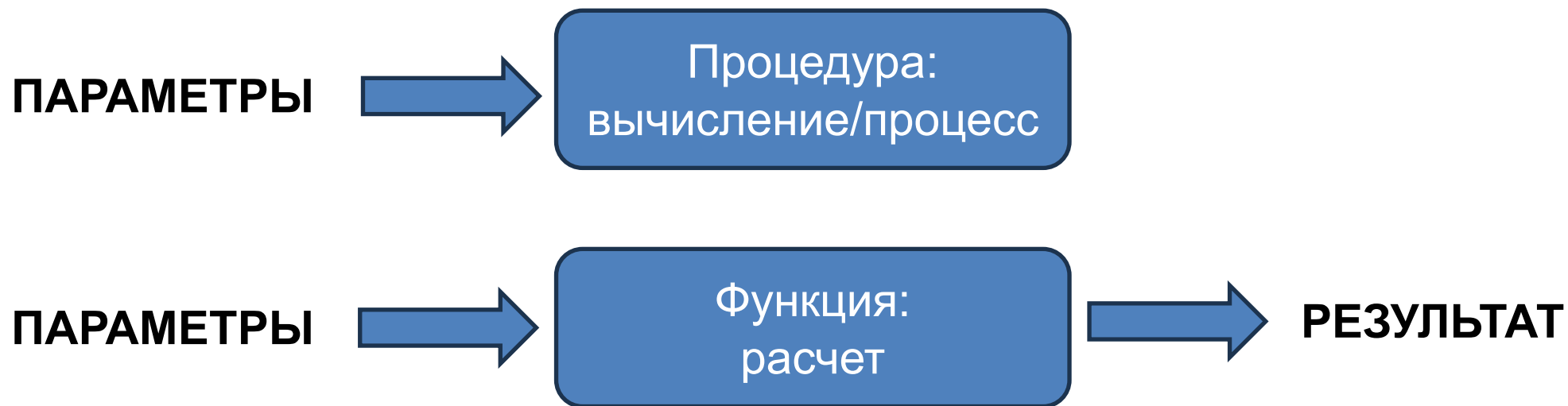


**СГУГиТ**

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

# ПРОЦЕДУРЫ И ФУНКЦИИ

Существует две категории подпрограмм



# СРЕДЫ ЛОКАЛЬНЫХ ССЫЛОК

```
int adder(int list[], int listlen) {  
    static int sum = 0;  
    int count;  
    for (count = 0; count < listlen; count ++)  
        sum += list [count];  
    return sum;  
}
```



**СГУГиТ**

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

# МЕТОДЫ ПЕРЕДАЧИ ПАРАМЕТРОВ

## Три режима передачи параметров:

- ☐ режим in (параметры передаются из программы в подпрограмму)
- ☐ режим out (параметры передаются из подпрограммы в программу)
- ☐ режим inout (оба)

## Реализации:

- ☐ Передача по значению (in)
- ☐ Передача по результату (out)
- ☐ Передача по значению-результату (inout)
- ☐ Передача по ссылке (inout)
- ☐ Передача по имени (варьируется)

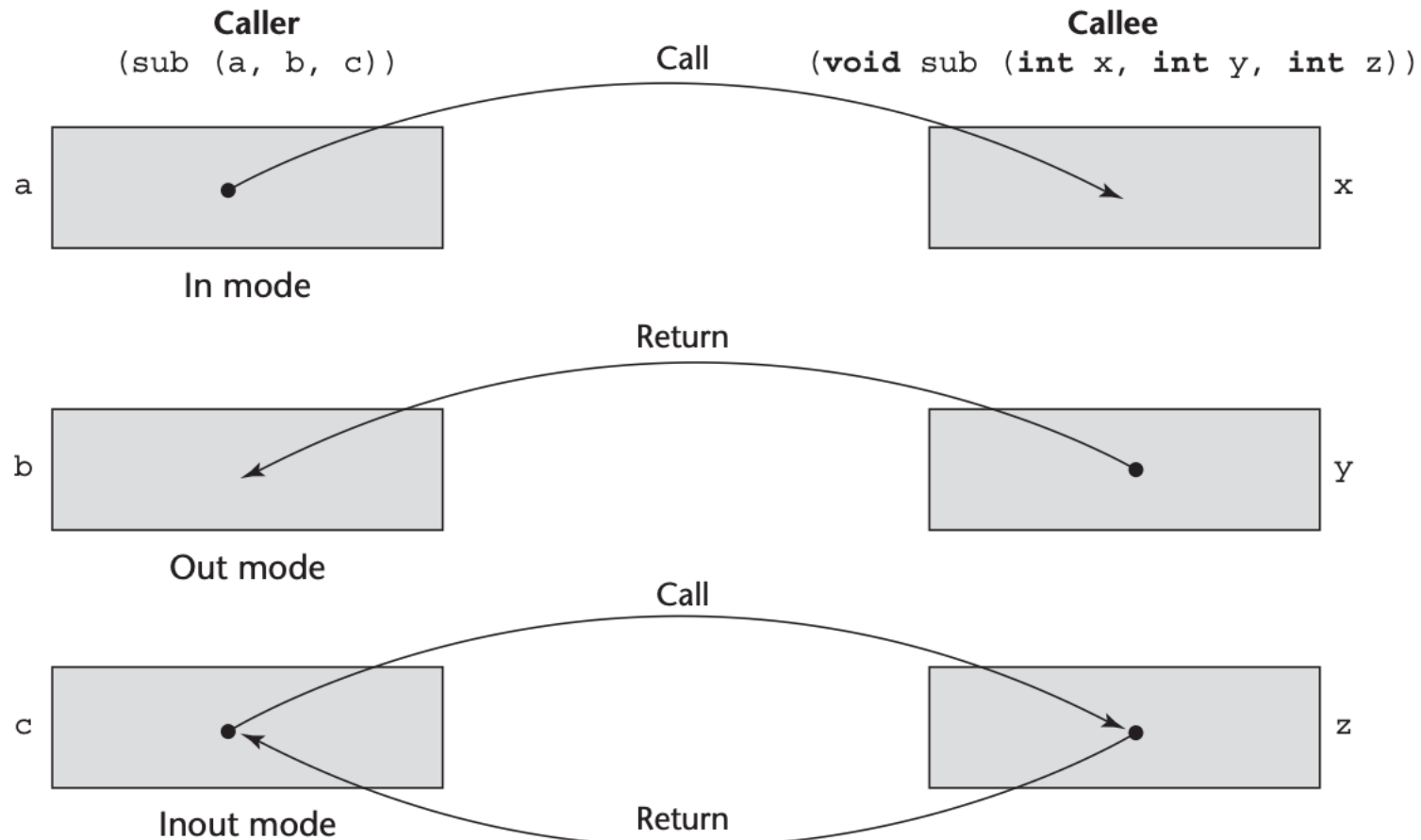


**СГУГиТ**

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ



# МЕТОДЫ ПЕРЕДАЧИ ПАРАМЕТРОВ



# МЕТОДЫ ПЕРЕДАЧИ ПАРАМЕТРОВ

## Передача по значению

- ☐ фактический параметр используется для инициализации формального параметра
- ☐ с этого момента формальный параметр действует как локальная переменная и полностью независима от фактического параметра
- ☐ реализуется путем физической передачи данных (копирование значения)
- ☐ Это может быть неэффективно, если переменная представляет собой массив или запись (структуру)

## Передача по результату

- ☐ формальный параметр действует как локальная переменная
- ☐ значение передается обратно в фактический параметр после завершения подпрограммы
- ☐ опять же, передача осуществляется путем физической передачи данных



**СГУГиТ**

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

# МЕТОДЫ ПЕРЕДАЧИ ПАРАМЕТРОВ

## Передача по значению-результату

- ☐ сочетает в себе передачу по значению и передачу по результату
- ☐ также называется передачей по копии
- ☐ требует двух действий копирования, что приводит к тем же недостаткам, что и при передаче по значению и передаче по результату

## Передача по ссылке

- ☐ передача пути доступа к ячейке памяти, хранящей переменную (то есть передача указателя)
- ☐ физическое копирование ограничивается адресом, а не данными, поэтому этот способ более эффективен, если параметр представляет собой массив или структуру и ничего не передается обратно
- ☐ создает псевдоним между формальными и фактическими параметрами
- ☐ требуется косвенная адресация для доступа к значению параметра



**СГУГиТ**

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

# МЕТОДЫ ПЕРЕДАЧИ ПАРАМЕТРОВ

## Передача по имени

- ☐ Вместо передачи параметра (значения или указателя) передается имя переменной
- ☐ в подпрограмме происходит текстовая подстановка
- ☐ заменить все экземпляры формального параметра на символы, составляющие реальный параметр.
- ☐ Этот подход используется для макроподстановки (в языках, поддерживающих расширение макросов), но не как форма передачи параметров
- ☐ В языке C макросы расширяются во время компиляции с помощью `#define`
- ☐ C++ и Ada используют это для общих подпрограмм
- ☐ В Lisp есть макроподстановка во время выполнения



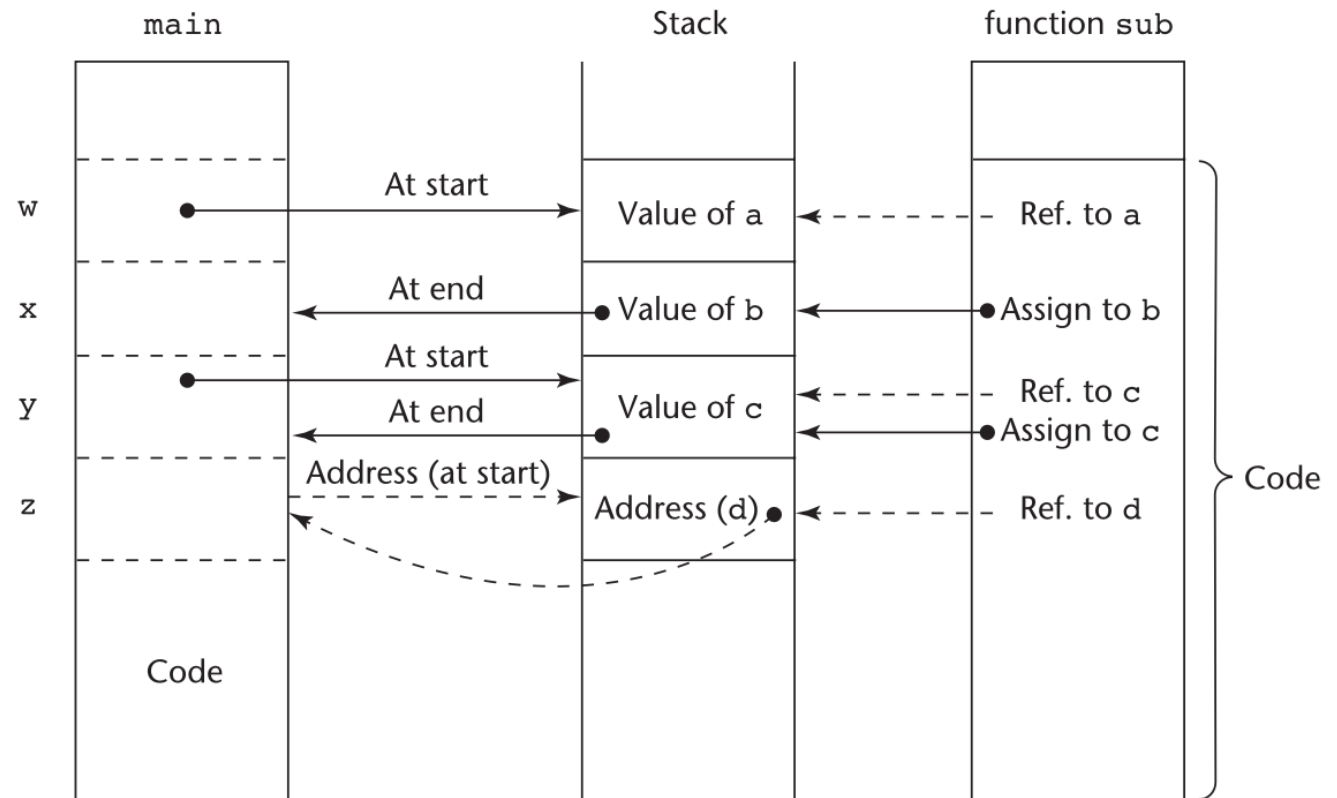
СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

# МЕТОДЫ ПЕРЕДАЧИ ПАРАМЕТРОВ

Передача параметров через  
стек времени выполнения

- ☐ по значению - копирование значения в стек
- ☐ по результату - копирование значения из стека
- ☐ По ссылке - копирование адреса параметра, автоматическое разыменование параметра в подпрограмме



# ПЕРЕГРУЗКА ПОДПРОГРАММ

**Перегруженная подпрограмма** - это подпрограмма, которая имеет то же имя, что и другая подпрограмма в той же среде ссылок.

- ❑ C++, Java, C# и Ada включают предопределенные перегруженные подпрограммы
- ❑ В Ada тип возврата перегруженной функции может быть использован для разграничения вызовов (таким образом, две перегруженные функции могут иметь одинаковые параметры)
- ❑ Ada, Java, C++ и C# позволяют писать несколько версий подпрограмм с одним и тем же именем

```
void fun(float b = 0.0);  
void fun();  
...  
fun();
```



# ПЕРЕГРУЗКА ОПЕРАТОРОВ

Операторы могут быть перегружены в Ada, C++, Python и Ruby

```
def __add__ (self, second):  
    return Complex(self.real + second.real, self.imag +  
        second.imag)
```

```
Complex operator +(Complex &second) {  
    return Complex(real + second.real, imag + second.imag);  
}
```



СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

# ЗАМЫКАНИЯ

```
function makeAdder(x) {  
    return function(y) {return x + y;}  
}  
...  
var add10 = makeAdder(10);  
var add5 = makeAdder(5);  
document.write("Add 10 to 20: " + add10(20) +  
    "<br />");  
document.write("Add 5 to 20: " + add5(20) +  
    "<br />");
```



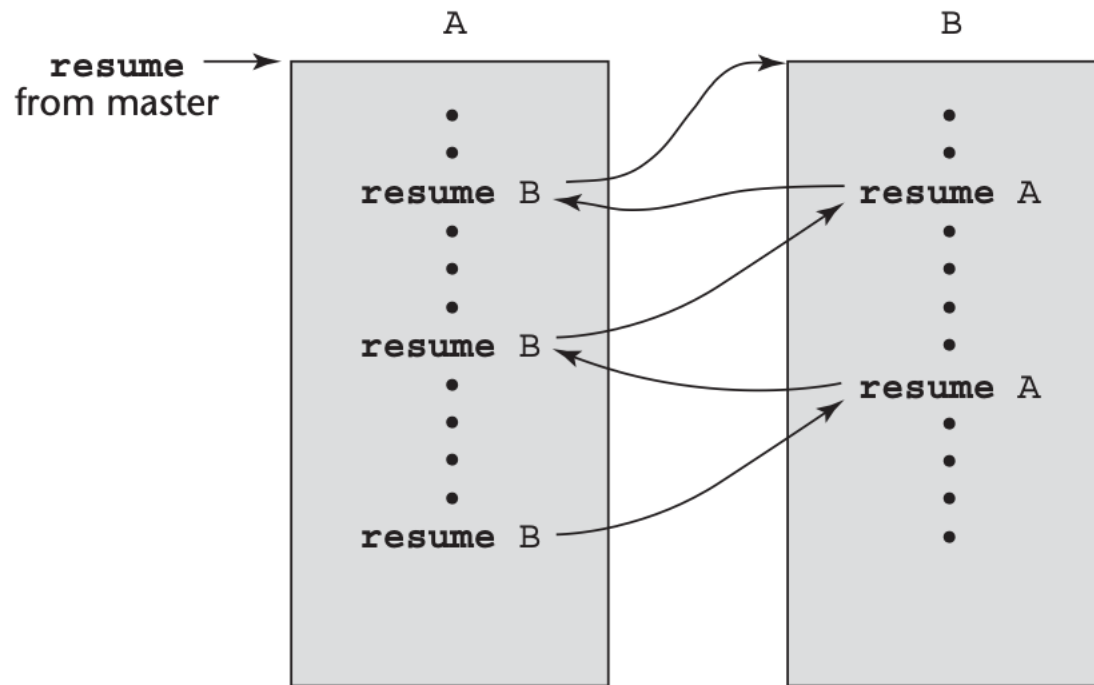


# КОРУТИНЫ

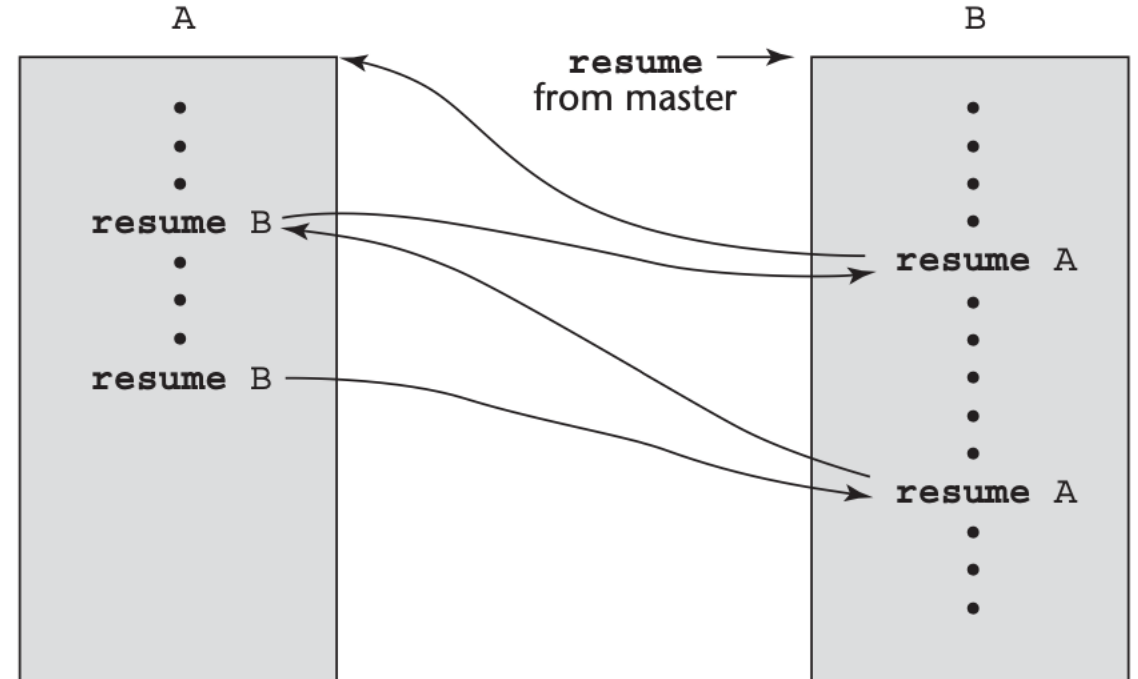
- ☐ **Корутина** - это подпрограмма, которая имеет несколько входов и сама управляет ими (поддерживает состояние).
- ☐ Вызов корутины называется возобновлением
- ☐ Первое возобновление корутины происходит в ее начало, но последующие вызовы происходят в точке сразу после последнего выполненного оператора в корутине
- ☐ Корутины многократно возобновляют друг друга, возможно, до бесконечности
- ☐ Корутины обеспечивают квазипоследовательное выполнение блоков программы (корутин); их выполнение чередуется, но не перекрывается



# КОРУТИНЫ



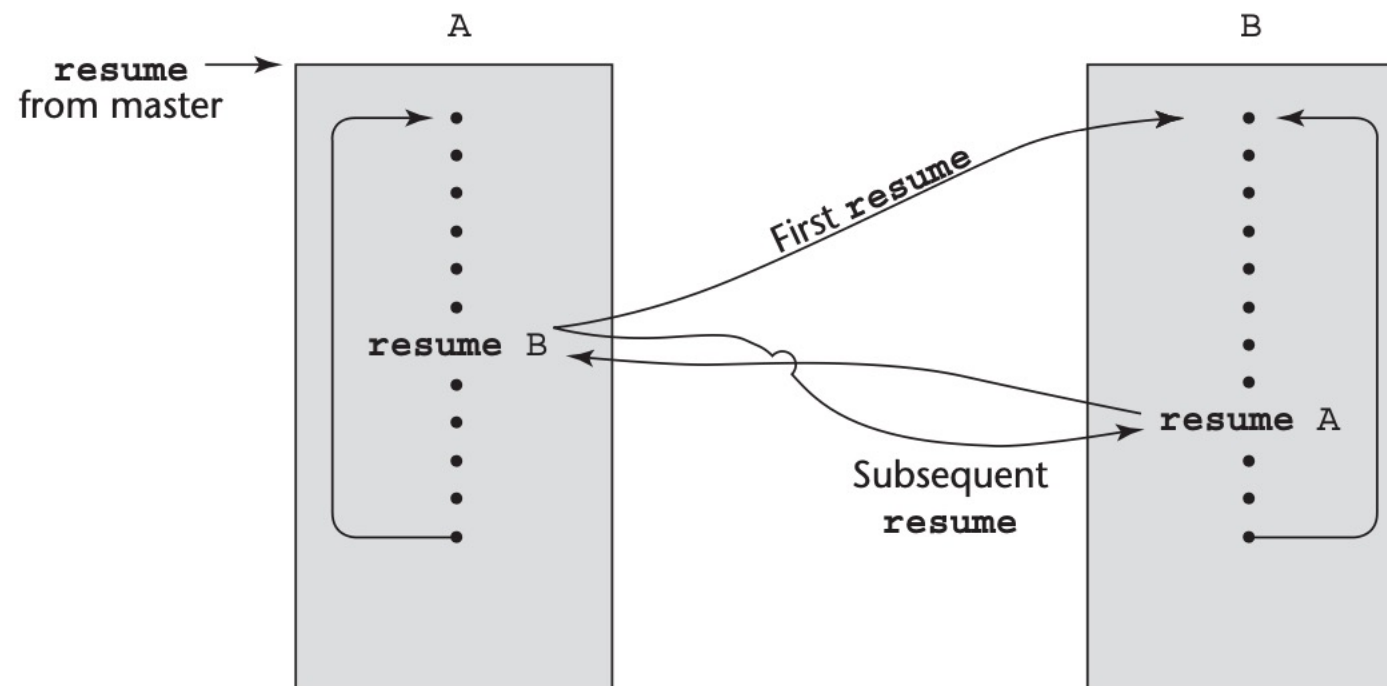
(a)



(b)



# КОРУТИНЫ



СГУГиТ

СИБИРСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ГЕОСИСТЕМ И ТЕХНОЛОГИЙ

«Величайшим мастерством программирования является искусство создания функций. Они не только упрощают жизнь разработчика, но и делают программу более гибкой и масштабируемой.»

