ПРОГРАММЫ, ПОДПРОГРАММЫ И ФУНКЦИИ

ОСНОВНАЯ ПРОГРАММА

ОСНОВНАЯ ПРОГРАММА.

***Основная программа - это любая программная единица, у которой первый оператор - это не* FUNCTION, SUBROUTINE *или* BLOCK DATA**

***Первым оператором основной программы может быть оператор* PROGRAM**

***Если у основной программы нет оператора* PROGRAM, *ей будет присвоено имя* MAIN**

- ***В этом случае, имя* MAIN *не может быть использовано как имя любого другого глобального объекта***

Выполнение программы всегда начинается с первого выполняемого оператора основной программы.

- ***Соответственно, в каждой выполняемой программе должна быть только одна основная программа***

Оператор PROGRAM.

Определяет программную единицу как основную программу и присваивает ей имя.

Синтаксис:

PROGRAM имя_программы

Где

имя_программы - *это определяемое пользователем имя основной программы.*

Особенности:

- ***Поэтому оно не может совпадать с именем любой внешней процедуры или именем* COMMON-блока.**
- ***Оно также является локальным именем основной программы и не должно вступать в противоречие с любым локальным именем в основной программе.***

ОПЕРАТОР PROGRAM

Пример:

```
PROGRAM GAUSS

REAL COEF (10,10), COST (10)

. . . . . .

. . . . . .

. . . . . .

END
```

ОПЕРАТОР PROGRAM

ПОДПРОГРАММЫ

ПОДПРОГРАММЫ.

***Подпрограмма - это автономно компилируемая программная единица, которая может быть вызвана из другой программной единицы с помощью оператора* CALL.**

Будучи вызванной, подпрограмма производит набор действий, определенных ее выполняемыми операторами, и затем возвращает управление на оператор, следующий непосредственно за вызвавшим ее оператором, или на оператор, определенный переменной меткой возврата .

***Подпрограмма не возвращает величины прямо, хотя они могут быть возвращены в вызывающую программную единицу через параметры и* COMMON-блоки.**

Оператор SUBROUTINE.

Определяет программную единицу как подпрограмму, присваивает ей имя и определяет формальные параметры для этой подпрограммы.

Формальные параметры могут содержать переменную метку возврата (*).

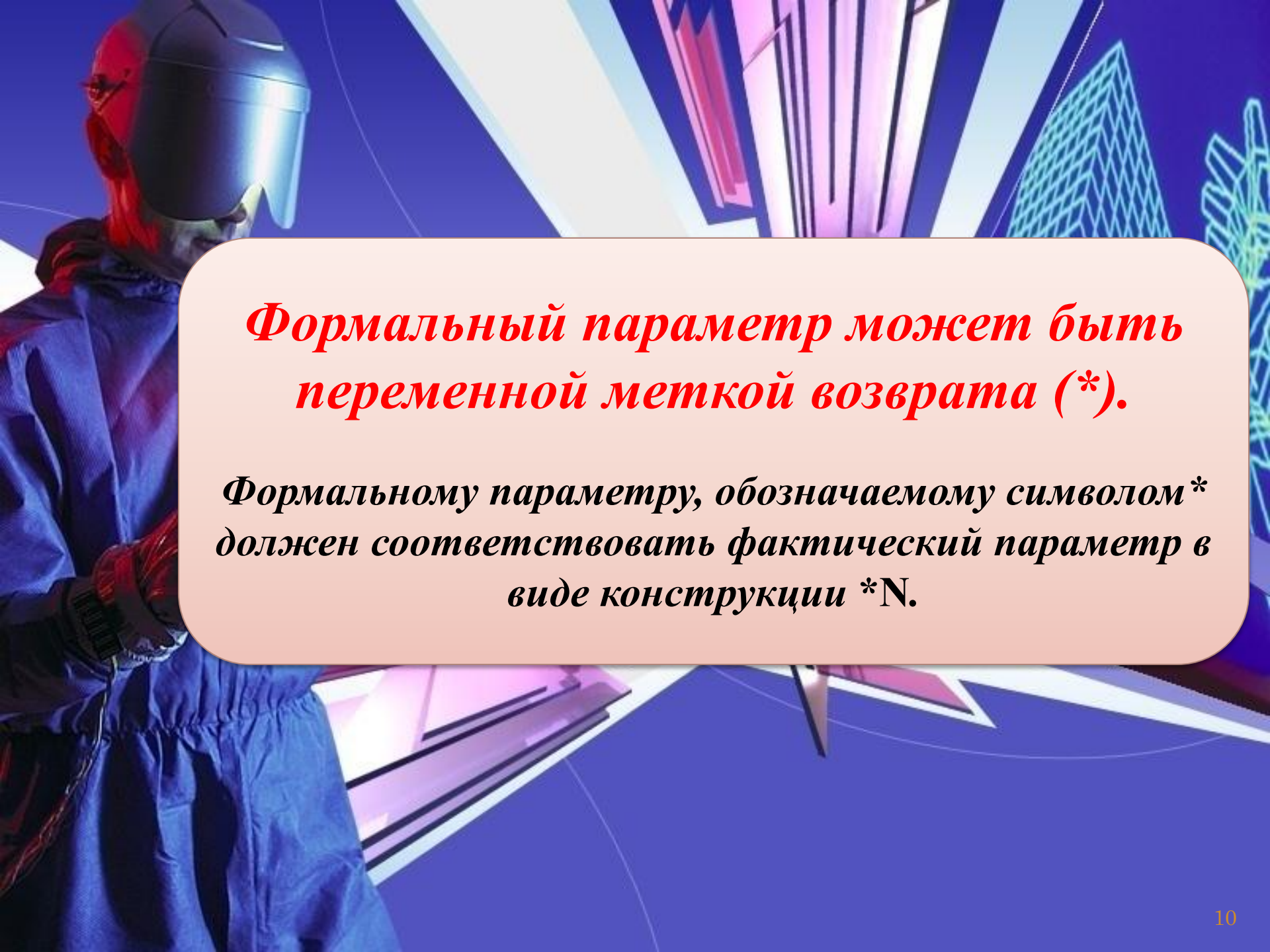
Синтаксис:

SUBROUTINE имя_подпрограммы [([форм-пар[,форм-пар]…])]

Где

имя_подпрограммы - *это определяемое пользователем глобальное внешнее имя подпрограммы*

форм-пар - *это определяемое пользователем имя формального параметра, называемого также фиктивным параметром.*



Формальный параметр может быть переменной меткой возврата (*).

Формальному параметру, обозначаемому символом* должен соответствовать фактический параметр в виде конструкции *N.

Особенности:

ОПЕРАТОР SUBROUTINE

Особенности:

Фактические параметры в операторе CALL, вызывающем

виду

- *Компилятор проверяет их на соответствие, если известны формальные параметры.*
- *Правила соответствия формальных и фактических параметров приведены в описании оператора* **CALL.**

ОПЕРАТОР SUBROUTINE

Пример:

```
      SUBROUTINE GETNUM (NUM, UNIT)
      INTEGER NUM, UNIT
   10 READ (UNIT, '(I10)', ERR=10) NUM
      RETURN
      END
```

Оператор RETURN.

Возвращает управление в вызываемую программную единицу.

Синтаксис:

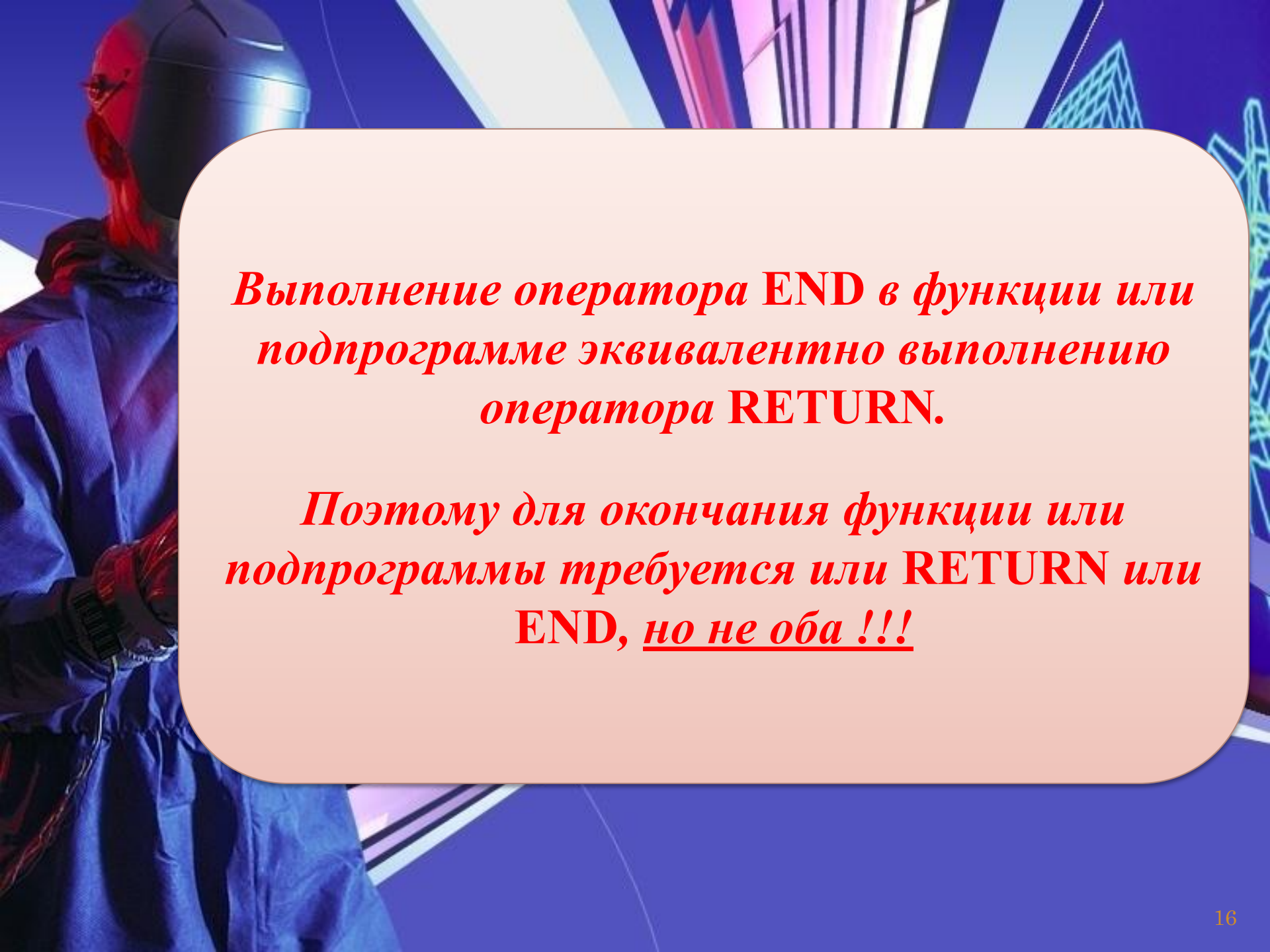
RETURN [I]

Где

I - *простая переменная целого типа стандартной длины или целая константа без знака.*

***Использование оператора* RETURN *в основной программе равносильно оператору* STOP.**

I - *является порядковым номером того символа * среди всех таких символов (звездочек) в списке формальных параметров данной подпрограммы (считая слева направо), которому соответствует в списке фактических параметров метка оператора возврата в вызывающую программу.*



***Выполнение оператора* END *в функции или подпрограмме эквивалентно выполнению оператора* RETURN.**

Поэтому для окончания функции или подпрограммы требуется или* RETURN *или* END, *но не оба !!!

Пример:

```
C   Пример оператора RETURN
C
C   Эта подпрограмма выполняет цикл
C   пока вы не наберете "Y"
C
      SUBROUTINE LOOP
      CHARACTER IN
C
10    READ (*,'(A1)') IN
      IF (IN.EQ.'Y') RETURN
      GO TO 10
C
C   Неявный RETURN
      END
```

Оператор CALL.

Вызывает и выполняет подпрограммы и другие программные единицы.

Синтаксис:

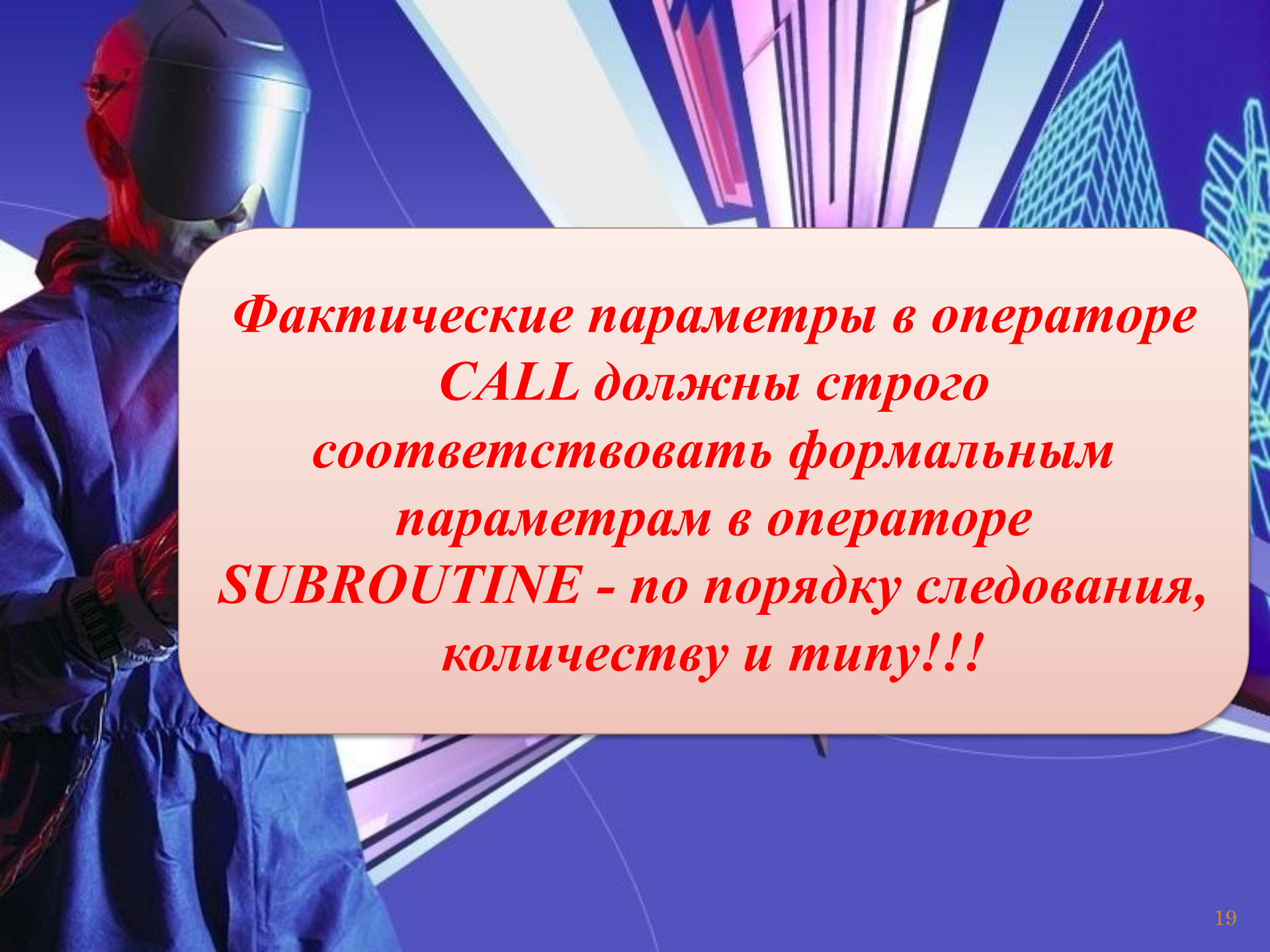
CALL имя[([параметр[,параметр]...])]

Где

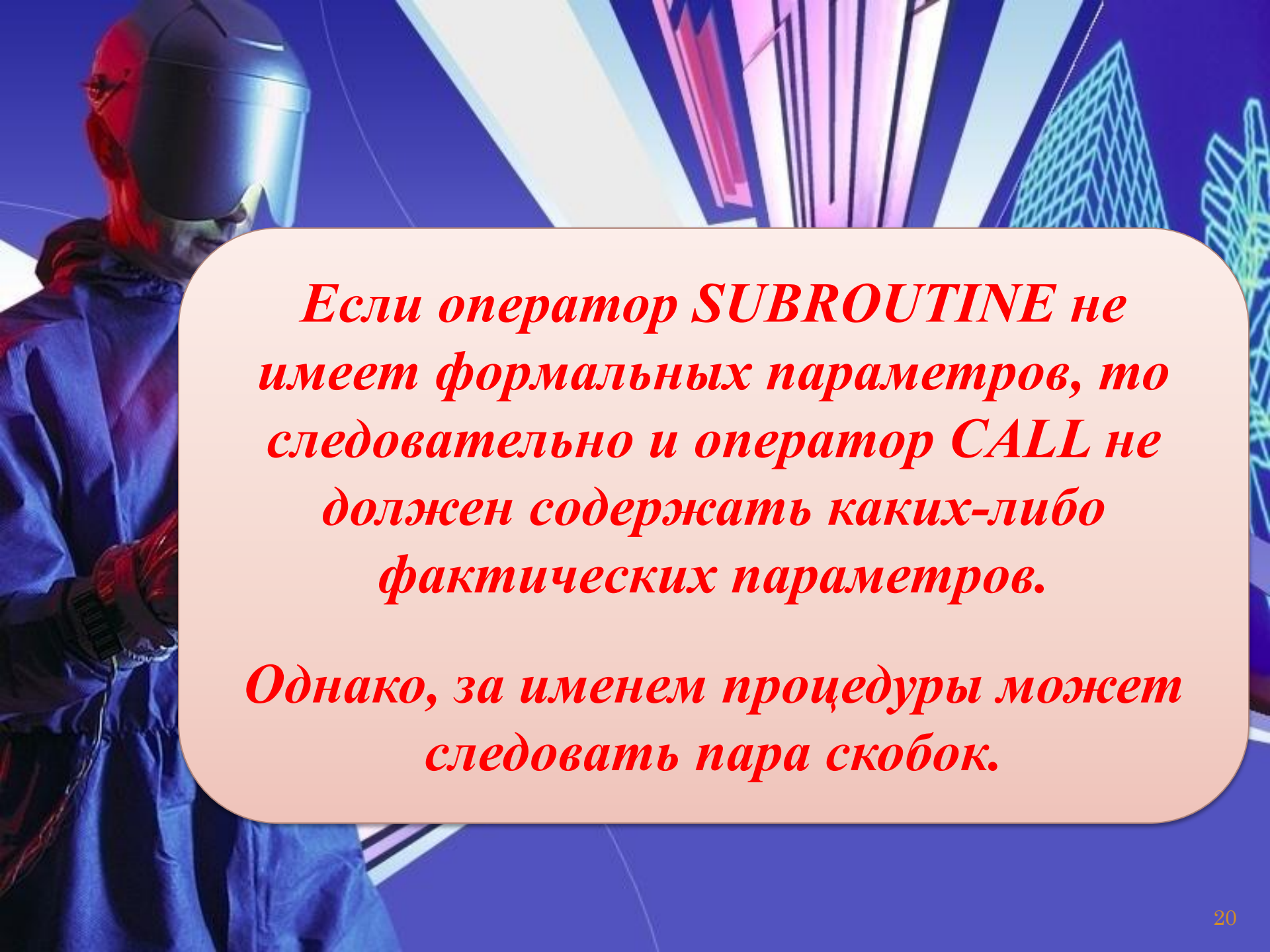
имя - *простая переменная целого типа стандартной длины или целая константа без знака.*

параметр - *фактический параметр, который может быть:*

1. *Определителем альтернативного возврата* (***n**).
2. *Выражением.*
3. *Константой (или выражением из констант).*
4. *Переменной.*
5. *Элементом массива.*
6. *Массивом.*
7. *Подпрограммой.*
8. *Внешней функцией.*
9. *Внутренней функцией, используемой как параметр.*



Фактические параметры в операторе CALL должны строго соответствовать формальным параметрам в операторе SUBROUTINE - по порядку следования, количеству и типу!!!



Если оператор SUBROUTINE не имеет формальных параметров, то следовательно и оператор CALL не должен содержать каких-либо фактических параметров.

Однако, за именем процедуры может следовать пара скобок.

Процесс выполнения оператора CALL

Фортран не допускает рекурсивного вызова подпрограмм.

Это означает, что подпрограмма не может вызывать сама себя, не может вызвать другую подпрограмму, в которой есть обращение к ней самой.

Компилятор не создает рекурсивных обращений, даже если они указаны!!!

Пример:

```
C   Пример оператора CALL
    IF (IERR.NE.0) CALL ERROR(IERRR)
    END
C
    SUBROUTINE ERROR(IERRNO)
    WRITE (*,200) IERRNO
200   FORMAT (1X,'ERROR',I5,'DETECTED')
    END
```

Пример:

```
C   Пример альтернативных возвратов
    CALL BAR (I,*10,J,*20,*30)
    WRITE (*,*)'normal return'
    GOTO 40
10  WRITE (*,*) 'return to 10'
    GOTO 40
20  WRITE (*,*) 'return to 20'
    GOTO 40
30  WRITE (*,*) 'return to 30'
40  CONTINUE
    . . .
    SUBROUTINE BAR (I,*,J,*,*)
    IF(I,EQ.10) RETURN 1
    IF(I.EQ.20) RETURN 2
    IF(I.EQ.30) RETURN 3
    RETURN
```

ОПЕРАТОР CALL

ПОДПРОГРАММА ДАННЫХ.

Определяет подпрограмму блока данных, в которой присваиваются начальные значения переменным и элементам массивов из поименованных* COMMON *блоков.

Синтаксис:

BLOCK DATA [имя]

Где

имя - *глобальное символьное имя подпрограммы, определяемой оператором* BLOCK DATA.

***Имя должно быть уникально среди имен локальных переменных или массивов которые определены в данной подпрограмме , а также среди имен внешних процедур,* COMMON*-блоков и других подпрограмм* BLOCK DATA.

Особенности:

ПОДПРОГРАММА ДАННЫХ

Ограничения на использование подпрограмм BLOCK DATA:

данных могут быть использованы только операторы **COMMON**, **DIMENSION**, **PARAMETER**, **IMPLICIT**, **EQUIVALENCE**, **SAVE**, **DATA**, **END**

- Именные **COMMON-блоки**, определенные в подпрограмме **BLOCK DATA** должны иметь уник

ПОДПРОГРАММА ДАННЫХ

ПОДПРОГРАММА ДАННЫХ

- Если не все элементы подпрограммы **COMMON-блока** определены (описаны) сначала, то это должно быть сделано в **COMMON-блоке**.

ФУНКЦИИ

ФУНКЦИИ.

- *На функцию ссылаются в выражении, и она возвращает величину, которая используется при вычислении этого выражения.*
- *Существует три вида функций:*

1. *Внешние функции*
2. *Встроенные функции*
3. *Функции-операторы.*

- *Ссылка на функцию может появиться в арифметическом или логическом выражении.*
- *Когда выполняется ссылка на функцию, функция вызывается, а величина результата используется как операнд в выражении, которое содержит ссылку на функцию.*

Форма ссылки на функцию следующая:

имя-функции ([пар[,пар]...])

Где

имя-функции - *это определенное пользователем имя внешней или встроенной функции или функции-оператора.*

пар - *это фактический параметр.*

Правила для параметров функций аналогичны правилам для подпрограмм (за исключением переменного возврата, который недопустим) и приведены в описании оператора CALL.

Внешние функции

- Внешняя функция определена программной единицей функции.
- Она начинается оператором **FUNCTION** и заканчивается оператором **END**.
- Она может содержать любые виды операторов, кроме **PROGRAM**, **FUNCTION**, **SUBROUTINE** или **BLOCK DATA**

Оператор FUNCTION.

Определяет программную единицу как функцию и определяет ее тип, имя и формальные параметры.

Синтаксис:

[тип] FUNCTION имя-функции
([параметр [,параметр]...])

Где

тип - *один из следующих:*
INTEGER, INTEGER*2, INTEGER*4,
REAL, REAL*4, REAL*8, DOUBLE PRECISION,
LOGICAL, LOGICAL*2, LOGICAL*4,
CHARACTER, CHARACTER*n,
COMPLEX, COMPLEX*8, COMPLEX*16.

имя-функции - *имя функции, задаваемое пользователем.*

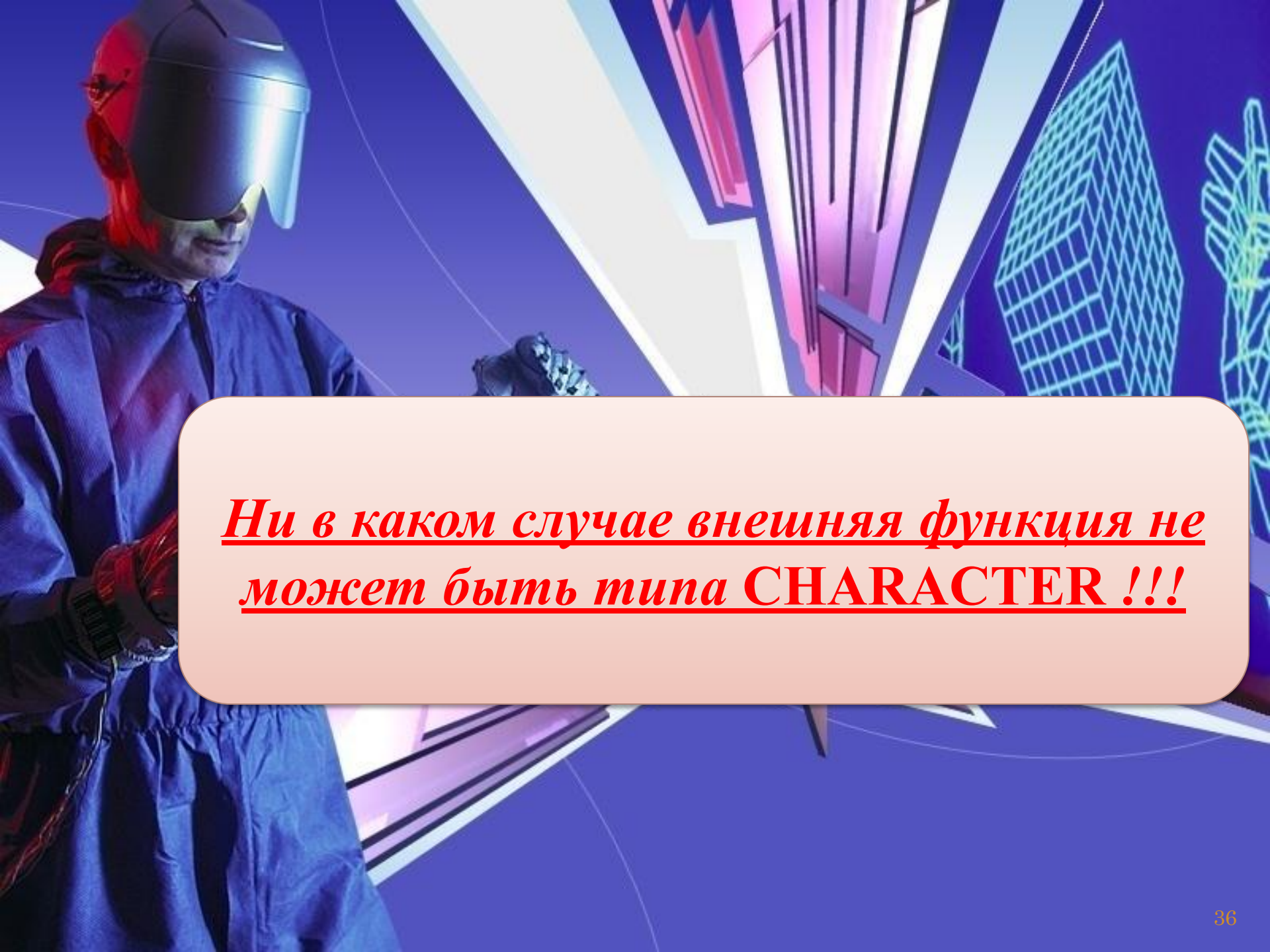
параметр - *имя формального параметра.*

Особенности:

ОПЕРАТОР FUNCTION

Особенности:

- *Например, недопустим такой оператор:*
- **CHARACTER* (*) FUNCTION F(X)**



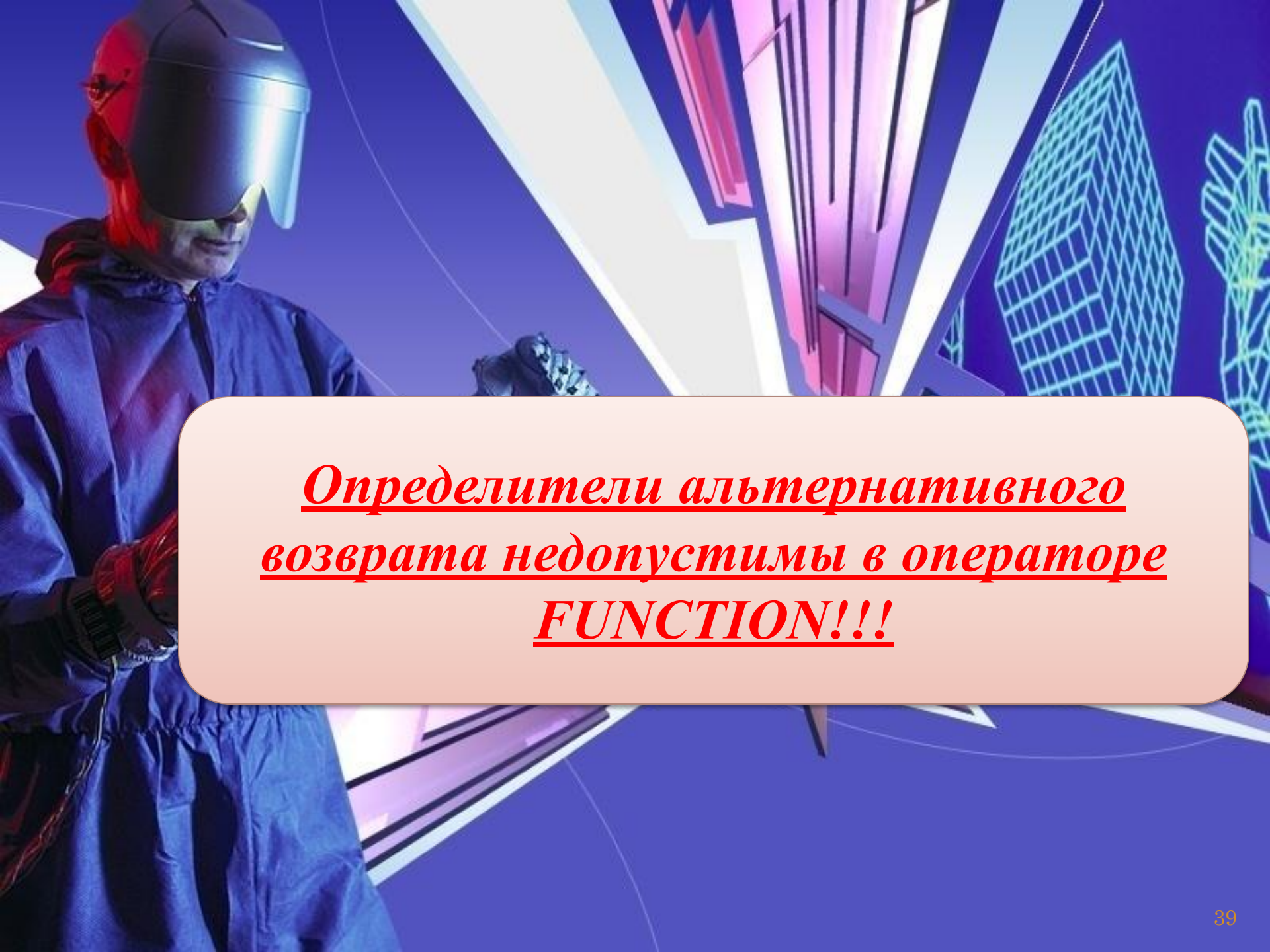
***Ни в каком случае внешняя функция не может быть типа* CHARACTER !!!**

Особенности:

Особенности:

- *После вычисления на величину этой переменной можно ссылаться в выражении так же как и на любую другую.*
- *Каждое вычисление функции должно сопровождаться присвоением величины этой переменной.*
- *Последняя величина этой переменной после выполнения оператора* **RETURN** *или* **END** *определяет значение функции.*

ОПЕРАТОР FUNCTION



Определители альтернативного возврата недопустимы в операторе FUNCTION!!!

Особенности:

Пример:

```
C   Пример использования функции GETNO,
C   которая читает число из файла.
    I=2
10  IF (GETNO(I) .EQ. 0.0) GOTO 10
    STOP
    END
C
    FUNCTION GETNO(NOUNIT)
    READ (NOUNIT,'(F10.5)') R
    GETNO=R
    RETURN
    END
```

Функции-операторы

- Функция-оператор определена единственным оператором и по виду подобна оператору присваивания.
- Функция-оператор может появиться только после операторов описания и перед любыми выполняемыми операторами в программной единице.

Функция-оператор.

Определяет функцию в виде одного оператора.

Синтаксис:

имя-функции ([пар [,пар]...])= выражение

Где

имя-функции - *имя функции-оператора, задаваемое пользователем.*

пар - *имя формального параметра.*

выражение - *любое выражение.*

Особенности:

- *Однако тело функции-оператора служит для определения значения функции-оператора.*

ФУНКЦИЯ-ОПЕРАТОР

- *оно не должно быть использовано, где-либо еще, кроме имени* **COMMON**-*блока или имени формального параметра в другой функции-операторе.*

- *Поэтому имена формальных параметров могут быть переопределены как другие имена пользователя в оставшейся части программной единице, за исключением определителя функции-оператора.*

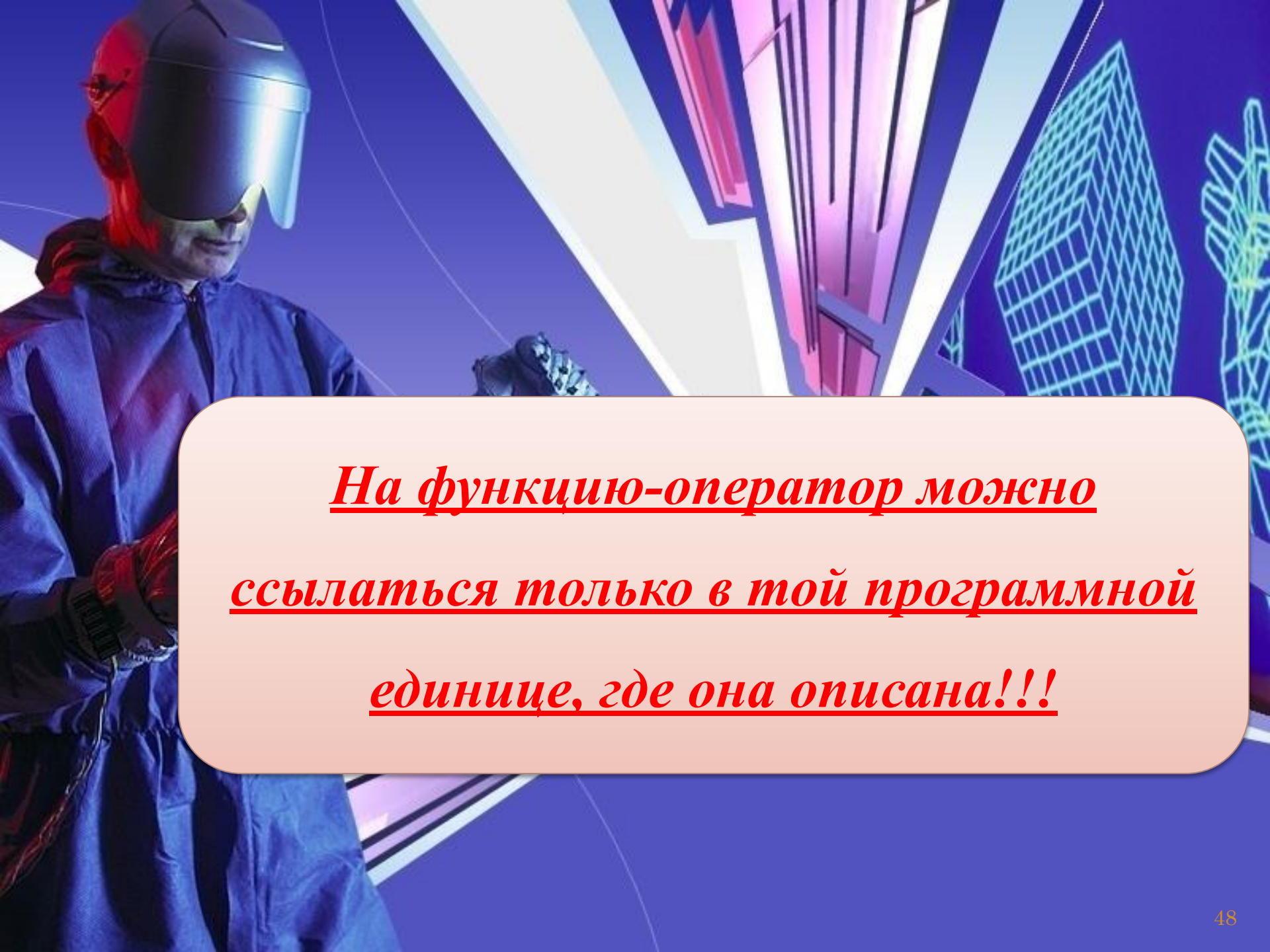
ФУНКЦИЯ-ОПЕРАТОР

- ***Тип выражения должен быть совместим с типом имени функции-оператора.***
- ***в последнем случае тип такого использования должен быть одинаковым***

ФУНКЦИЯ-ОПЕРАТОР

- ***Ссылки на функции-операторы должны относится к функциям, описанным до того, как они употреблены здесь.***
- ***Ссылка на функцию-оператор не может быть вызвана рекурсивно, как прямо так и косвенно.***

ФУНКЦИЯ-ОПЕРАТОР



На функцию-оператор можно ссылаться только в той программной единице, где она описана!!!

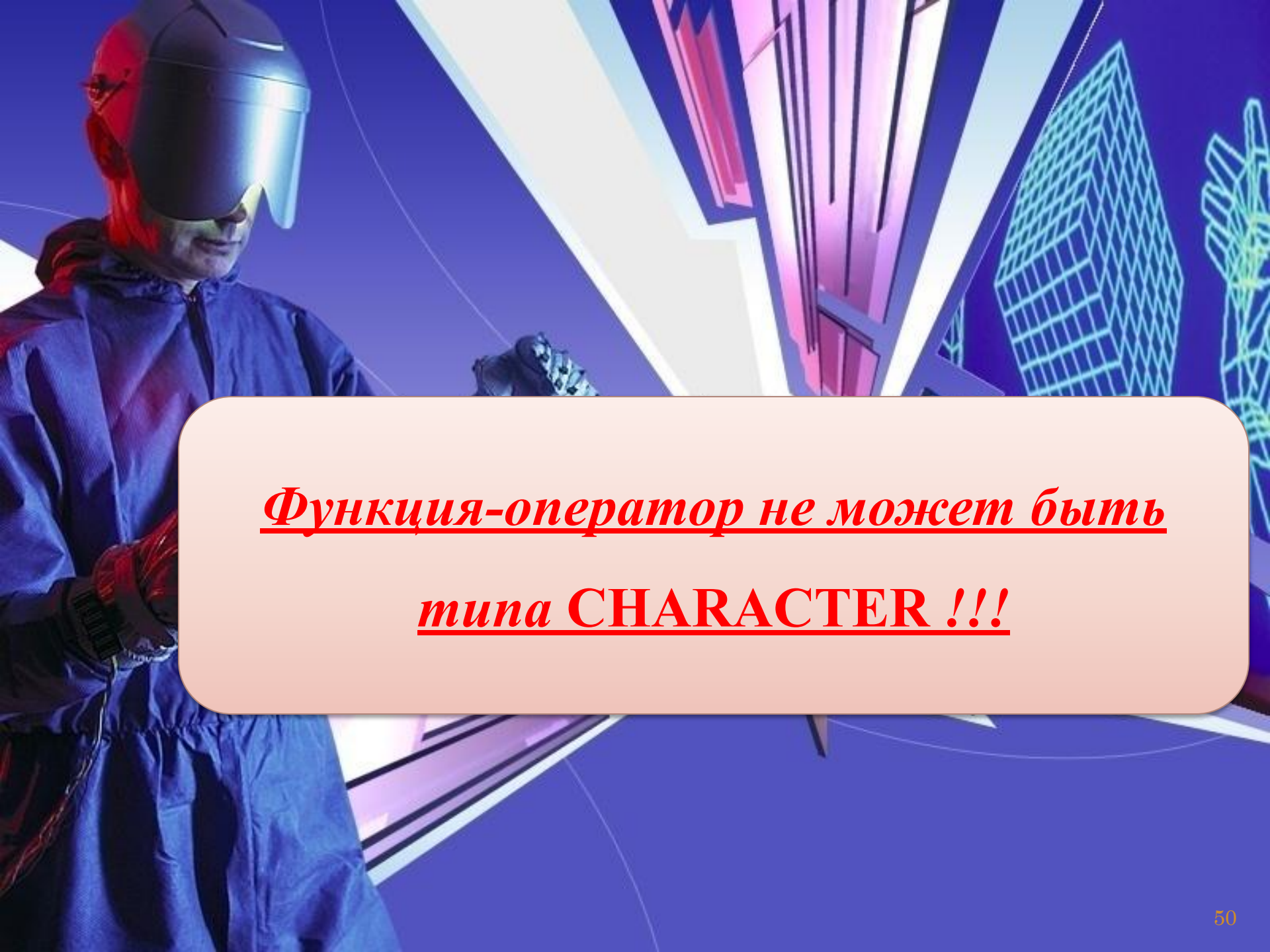
Имя функции-оператора не должно появляться ни в каких описывающих операторах, за исключением:

- **операторов описания типа**

  (которые не могут описывать это имя, как массив)

- **оператора COMMON**

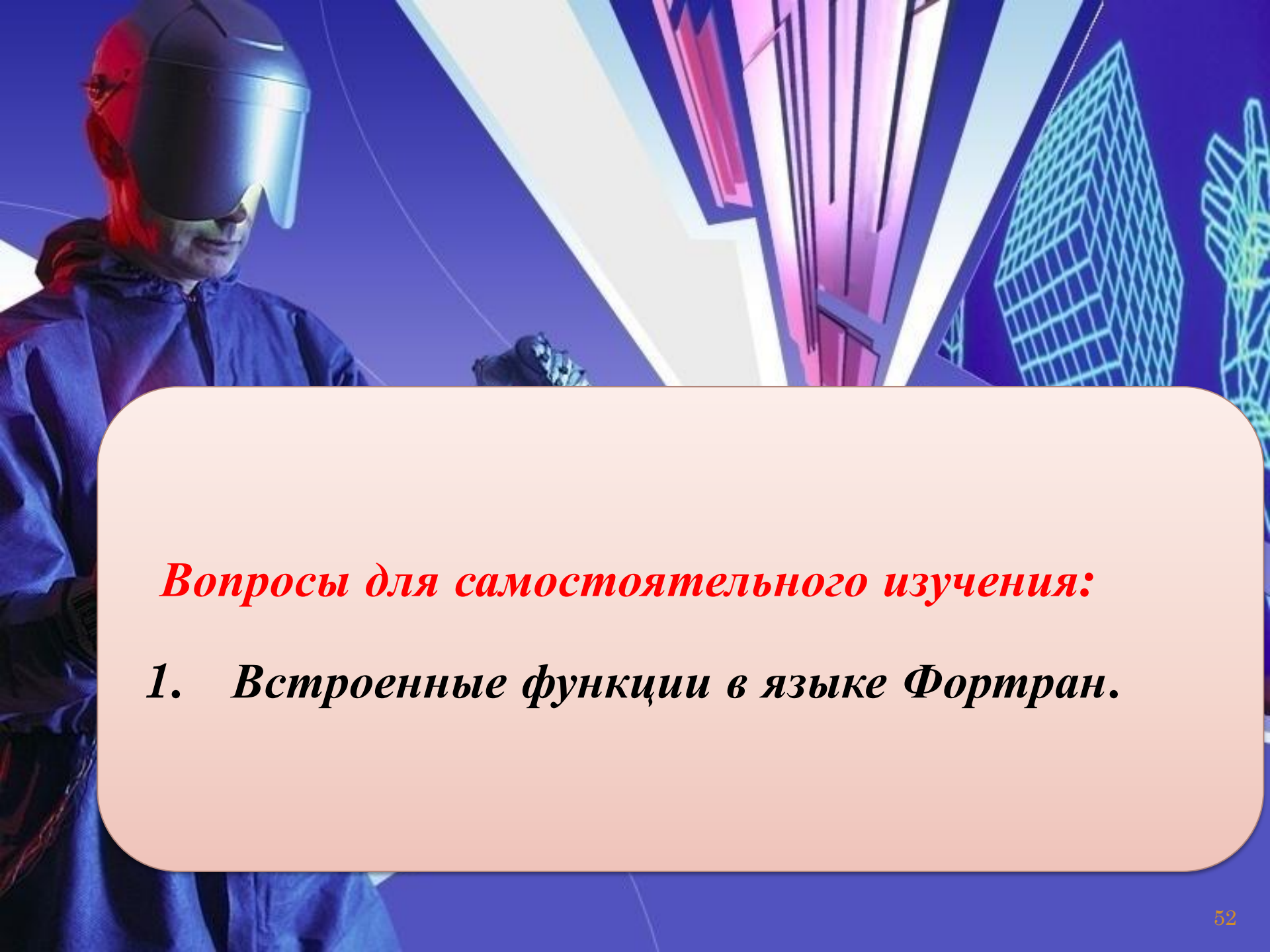
  (как имя **COMMON***-блока)*



***Функция-оператор не может быть типа* CHARACTER !!!**

Пример:

```
C   Пример оператора функция-оператор
    DIMENSION X(10)
    ADD(A,B)=A+B
C
    DO 1 I=1,10
    X(I)=ADD(Y,Z)
1   CONTINUE
```



Вопросы для самостоятельного изучения:

1. ***Встроенные функции в языке Фортран.***