Programowanie w języku Fortran

dr inż. Maciej Woźniak [1]

[1]Katedra Informatyki, Wydział Informatyki, Elektroniki i Telekomunikacji,
Akademia Górniczo-Hutnicza, Kraków, Polska

Testy jednostkowe

Istnieje kilka bibliotek do testów jednostkowych

- FUnit
- Fortran Unit Test Framework (FRUIT)
- FortUnit
- Ftnunit
- **pFUnit**
- Vegetables
- UnitTest
- Zofu

pFUnit

- Wspiera programowanie równoległe (MPI, OpenMP)
- Przetestowany z kompilatorami GNU, Intel, NAG, PGI Fortran
- Budowanie przezGNU Make albo CMake
- Wyjście w zestandaryzowanym **xml**
- Można podpiąć do systemów ciągłej integracji - np. **Jenkins**

Plik **test.pf**

```
@test
subroutine test_knot1()
  use knot_vector, only: FillOpenKnot
  use pfunit_mod
  implicit none
  integer (kind = 4) :: n, p
  real (kind = 8), allocatable, dimension(:) :: U
  real (kind = 8), allocatable, dimension(:) :: res
```

Plik **test.pf** - cd.

```
  ! Number of subintervals is N = n-p+1
    n = 0
    p = 0
  ! N = 1
    allocate(res(n + p + 2))
    call FillOpenKnot(res, n, p)
    U = (/ 0.0, 1.0 /)
    @assertEqual(U, res)
    deallocate(res)
  end subroutine test_knot1
```

Plik **testSuites.inc**

ADD_TEST_SUITE(test_knot_suite)

Późna ewaluacja

```
type(integer_operand), target :: x
type(integer_operand), target :: y
type(integer_relation), pointer :: relation

relation => x+y == 0
x=1
y=-1
write(*,*) integer_relation_eval(relation)
```

Późna ewaluacja

```
function integer_relation_eval(realtion) result (value)
  type(integer_relation) :: relation
  logical :: value

  call integer_eval(relation%first)
  call integer_eval(relation%second)
```

Późna ewaluacja

```
recursive subroutine integer_eval(x)
  type(integer_operand) :: x
  if ( associated (x%first) ) call integer_eval(x%first)
  x%value = x%first%value + x%secod%value
```

Ciekawostka

Istnieje biblioteka do programowania w Fortran-ie **Functional Fortran**

`github.com/wavebitscientific/functional-fortran`