

Dokumentacja języka „Julian++”

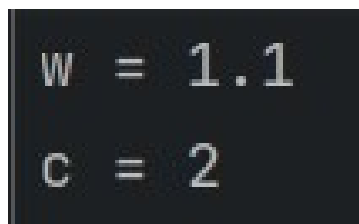
1) Typy danych

I. Integer

II. Real

Oba typy danych nie muszą być specjalnie deklarowane, program automatycznie rozpozna czy liczba jest całkowita czy zmienna-przecinkowa.

Przykładowa deklaracja:



```
w = 1.1  
c = 2
```

Rys. 1 Typy danych

2) Podstawowe operacje matematyczne

I. Dodawanie

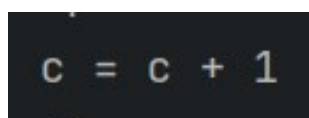
II. Odejmowanie

III. Mnożenie

IV. Dzielenie

Możliwe jest wykonanie powyższych działań matematycznych z wykorzystaniem zmiennych bądź liczb, jednakże działania muszą być wykonywane dla tego samego typu zmiennej całkowitej lub zmiennoprzecinkowej.

Przykładowa operacja matematyczna:



```
c = c + 1
```

Rys. 2 Dodawanie

3) Wczytywanie oraz wyświetlanie danych

- I. Funkcja read()
- II. Funkcja write()

Możliwe jest wyświetlanie zmiennych za pomocą funkcji write() oraz wczytywanie wartości zmiennych z klawiatury za pomocą funkcji read().

Przykładowe użycie funkcji write():

```
x = 5  
write x
```

Rys. 3 Funkcja write()

4) Instrukcje warunkowe

- I. Równe: ==
- II. Różne: !=
- III. Większe: >
- IV. Mniejsze: <

Możliwe jest zdefiniowanie instrukcji warunkowej według poniższego schematu:

if warunek **then**

treść

endif

Przykładowe użycie instrukcji warunkowej:

```
if x < 5 then  
  f = c + x  
endif  
if x == 5 then  
  f = c * x  
endif  
if x > 5 then  
  f = x - c  
endif
```

Rys. 4 Instrukcje warunkowe

5) Pętle

W języku Julian++ możliwe jest zastosowanie pętli, liczba powtórzeń może być określona przez liczbę bądź zmienną według poniższego schematu:

loop liczba_powtórzeń

treść

endloop

Przykładowe zastosowanie pętli:

```
c=1
j = 2
loop j
loop j
    c = c + 1
endloop
endloop
```

Rys. 5 Pętla loop

6) Funkcje

Dostępne jest również stosowanie funkcji w języku Julian++. Przy definiowaniu funkcji należy zadeklarować, czy będzie ona zwracała wartości całkowite typu int czy zmiennie-przecinkowe typu real. Wewnątrz funkcji można wczytywać oraz wyświetlać wartości, stosować wyrażenia warunkowe oraz pętle.

Dodatkowo wewnątrz funkcji można deklarować zmienne lokalne, jeśli zmienna o tej samej nazwie występuje poza funkcją – jako globalna, wewnątrz funkcji po użyciu tej samej nazwy zostanie nadpisana.

Funkcje są zbudowane według poniższego schematu:

function typ_danych(“int” lub “real”) nazwa_funkcji

treść

endfunction

Przykładowa implementacja funkcji w języku Julian++:

```
function real e
  t = 22.3
  write t
  d = d + w
  write d
  d = w + 1.1
  e = d + w
endfunction
t = e
```

Rys. 6 Funkcja typu real