Simpla (**Simp**le **la**nguage)

- Paradigma imperativo
- Dichiarazioni di variabili, funzioni e corpo del programma
- Funzioni non innestabili (solo globali)
- Variabili globali e locali
- Passaggio dei parametri per valore
- Corpo della funzione = sequenza di istruzioni
- Ricorsione
- Tipi: **`integer`**, **`real`**, **`string`**, **`boolean`**, **`void`** (solo per funzioni senza valore di ritorno)
- Commenti: `# Questo è un commento`

Tipi di Dati e Costanti

- **integer**

```
i : integer;
i = 25;
```

- **real**

```
r : real;
r = 64.15
```

- **string**

```
s : string;
s = "alfa";
```

- **boolean**

```
ok : boolean;
ok = true;
```

Dichiarazioni di variabili multiple

```
i, j, k: integer;
nome, cognome: string;
media, tempo: real;
ok, flag: boolean;
```

Struttura del Programma

```
numero: integer;

func fattoriale(n: integer): integer
  fact: integer;
body
  if n == 0 then
    fact = 1;
  else
    fact = n * fattoriale(n-1);
  end;
  return fact;
end;

func stampaFattoriali(tot: integer): void
  i, f: integer;
body
  for i=0 to tot do
    f = fattoriale(i);
    writeln("Il fattoriale di ", i, "è ", f);
  end;
end;

body
  read(numero);
  if numero < 0 then
    writeln("Il numero ", numero, "non è valido");
  else
    stampaFattoriali(numero);
  end;
end.
```

- Corpo del programma = sequenza di istruzioni finali racchiuse tra **`body`** ed **`end.`**
- Non necessaria la dichiarazione della funzione prima della sua chiamata

Espressioni Aritmetiche

- +, -, *, / : operatori (overloaded) applicabili a $\left\langle \begin{array}{l} \texttt{integer} \\ \texttt{real} \end{array} \right.$
- Espressioni miste non permesse (coercizione vietata)
- Operatori di cast $\left\langle \begin{array}{l} \texttt{integer : real} \rightarrow \texttt{integer} \\ \texttt{real : integer} \rightarrow \texttt{real} \end{array} \right.$

```
i,j: integer;
x, y, r: real;
...
x = real(i+j)*(r-real(i));
j = integer(x+y-1.25);
```

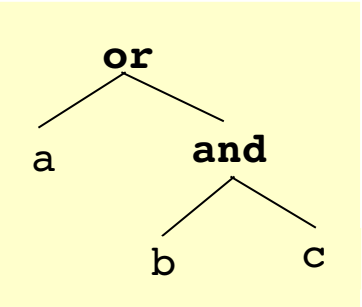
Espressioni Booleane

- Operatori relazionali (applicabili a tutti i tipi): **==**, **!=**, **>**, **>=**, **<**, **<=**
- Operatori logici (applicabili solo a **boolean)**: **and**, **or**, **not**
- Valutazione espressioni logiche: in corto circuito

```
a, b, c, d: boolean;
...
d = a or (b and c);
```

⟺

```
if a then
  d = true
else
  if not b then
    d = false
  else
    d = c
  endif
endif
```



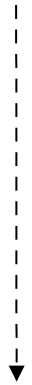
- Integrazione con operazioni relazionali

```
i, j: integer;

cognome: string;
a, b: boolean;
…
b = (i == j+2 or a) and (cognome == "Rossi");
```

Precedenza, Associatività, Ordine di Valutazione

Operatore	*Tipo*	*Associatività*
`and`, **`or`**	binario	sinistra
`==`, **`!=`**, **`>`**, **`>=`**, **`<`**, **`<=`**	binario	no
`+`, **`-`**	binario	sinistra
`*`, **`/`**	binario	sinistra
`-`, **`not`**	unario	destra



precedenza crescente

- Ordine di valutazione degli operandi: da sinistra a destra

Espressione Condizionale

- **if** *expr* **then** *expr* **else** *expr* **end**

```
a, b, c: integer;

a = if b>c then b+c else (a+1)*b end;
```

Istruzione Condizionale

- **if** *expr* **then** *stat-list* [**else** *stat-list*] **end**

```
a, b, c: integer;
...
if a == b then
    c = a + 1;
else
  if d > a then
    b = b - a;
  end;
end;
```

Ciclo While

- **`while`** *expr* **`do`** *stat-list* **`end`**

```
a, b, res: integer;
...
res = 0;
while a >= b do
    res = res + 1;
    a = a – b;
end;
```

Ciclo For

- **`for`** **id =** *expr* **to** *expr* **`do`** *stat-list* **`end`**

```
i, tot, sum: integer;
...
sum = 0;
for i=1 to tot do
  sum = sum + i;
end;
```

Break

- **`break`**: per uscire dal ciclo (while o for) corrispondente

```
i, tot, sum: integer;
...
sum = 0;
for i=1 to tot do
  sum = sum + i;
  if sum > 100 then
    break;
  end;
end;
```

Return

- **`return`** [*expr*]

```
func media(n: integer, m: integer): real
  somma: integer;
body
  somma = n + m;
  return real(somma) / 2.0;
end;
```

```
func fattoriale(n: integer): integer
body
  if n <= 0 then
    return 1;
  else
    return n * fattoriale(n-1);
  end;
end;
```

oppure:

```
func fattoriale(n: integer): integer
body
  return if n <= 0 then 1 else n * fattoriale(n-1) end;
end;
```

- Se funzione senza valore di ritorno (**`void`**), (eventuali) **`return`** senza argomento

Write

- **`write(`***expr-list***`)`**

```
write("Hello world!");
```

```
n, f: integer;
...
f = fattoriale(n);
write("Il fattoriale di ", n, " è ", f);
```

```
n, m: integer;
avg: real;
...
write("La media di ", n, " e ", m, " è ", media(n, m));
```

- Variante: **`writeln`**, come **`write`**, ma stampa anche un fine linea (newline)

Read

- **read(***id-list***)**

```
n, f: integer;

write("Inserisci il numero: ");
read(n);
f = fattoriale(n);
writeln("Il fattoriale di ", n, " è ", f);
```

```
n, m: integer;
avg: real;
...
write("Inserisci i due numeri di cui calcolare la media: ");
read(n, m);
avg = media(n, m);
writeln("La media di ", n, " e ", m, " è ", avg);
```