Caratteristiche Linguaggio

Generali

- Paradigma imperativo
- Il linguaggio prevede che il programma sia costituito da due sezioni:
  - Dichiarazione di variabili globali
  - Corpo del programma (**inizio** [..] **fine**)
- Tipi: **intero**, **decimale**, **stringa**, **boolean**
- Commenti: ***//*** **commento**

Tipi di Dati e Costanti

- **intero**

```
intero: numero;
[..]
numero = 22;
```

- **decimale**

```
decimale: numeroDec;
[..]
numeroDec = 22.22;
```

- **stringa**

```
stringa: str;
[..]
str = “Hello world!”;
```

- **boolean**

```
boolean: a, b;      // è possibile dichiarare più di una variabile dello stesso tipo
[..]                // nello stesso statement
a = vero;
b = falso;
```

Espressioni Booleane

- Operatori logici, applicati solo ai **boolean**: **e**, **o**
- Operatori relazionali: **<**, **<=**, **>**, **>=**, **==**, **!=**
- Utilizzate per la condizione del **se** statement

Operatori relazionali su stringhe

“Ciao**” == “**Ciao**” → vero**

“abcd**” < “**abc**” →** strlen(“abcd”) < strlen(“abc”) **→ falso**

“abcd**” <=** 4 **→** strlen(“abcd”) <= 4 **→ vero**

Operatori relazionali su interi e decimali

- Significato canonico

Operatori logici su boolean

- Significato canonico
- Le espressioni logiche sono valutate in corto circuito

Precedenza, ordine di valutazione e associatività nelle espressioni booleane

- Ordine di valutazione degli operandi: da sinistra a destra

Operatore	Tipo	Associatività
e, **o**	Binario	Sinistra
<, **<=**, **>**, **>=**, **==**, **!=**	Binario	No

N.B. Precedenza crescente dall’alto verso il basso

Espressioni Aritmetiche

- Operatori: **+**, **-**, ***, **/**
- Operatori overloaded applicabili ad **intero** e **decimale**
- Espressioni miste permesse
- Coercizione permessa

- **Coercizione**

```
intero: a;
decimale: b,c;
[..]
b=a+c;
```

⇨

```
(float(a) + c);
```

- **Precedenza, ordine di valutazione e associatività nelle espressioni aritmetiche**
  - Ordine di valutazione degli operandi: da sinistra a destra

Operatore	Tipo	Associatività
+, **-**	Binario	Sinistra
***, /,radice**	Binario	Sinistra

N.B. Precedenza crescente dall'alto verso il basso

Istruzione: se

- **se (** *logical-expr* **) vero fai :** *stat-list* **[se falso fai :** *stat-list* **] fine**

```
intero: numero;
[..]
numero = 22;
se (numero==10 o numero>20 ) vero fai:
        numero = 3;
se falso fai:
        numero = 1;
fine;
```

- Vincolo: type(*logical-expr*) = **boolean**

- Semantica espressa in linguaggio C:

```
if(logical-expr){
        stat-list
}
else{
        stat-list
}
```

Istruzione: ripeti

- **ripeti intconst volte :** *stat-list* **fine**

```
intero: a;
[..]
a=10;
ripeti 3 volte:
        a=a-2;
fine;
```

- Semantica espressa in linguaggio C:

```
for( i=0; i<intconst; i++ ){
        stat-list
}
```

Istruzione: stop

- Vincolo: **stop** può essere utilizzata solo all'interno di un'istruzione **ripeti**

```
intero: a;
[..]
a=10;
ripeti 3 volte:
        a=a-2;
        stop;
fine;
```

- Semantica espressa in linguaggio C:

```
for( i=0; i<intconst; i++ ){
        stat-list
        break;
}
```

Istruzione: scrivi

- **scrivi(** *scrivi-arg* **)**

```
intero: a;
[..]
a=10;
scrivi( a );
```

- Semantica espressa in linguaggio C:

```
printf("%d", a);
```

Istruzione: inserisci

- **Inserisci (id)**

```
intero: a;
[..]
Inserisci(a);
fine;
```

- Vincolo: **id** deve esser stato dichiarato
- Semantica espressa in linguaggio C:

```
int id;
scanf("%d", &id);
```

Istruzione di incremento/decremento

- **id++ | id--**

```
intero: a;
[..]
a=10;
a++;
a--;
```

- Vincolo: type(**id**)**==intero**
- Semantica espressa in linguaggio C:

```
int id = intconst;
id = id+1;
id = id-1;
```

Istruzione: radice

- **radice(** *math-expr* **)**

```
decimale: a;
[..]
a=10;
a=radice(a);
```

- Vincolo: type(*math-expr*)**==intero ||** type(*math-expr*)**==decimale**
- Semantica espressa in linguaggio C:

float(sqrt(double(*math-expr* **)));**