Il tipo Text

- Una variabile **text** in Simula è un array che contiene un puntatore ad una catena di carattere e un indicatore **pos** di posizioni all'interno della catena.
- Le catene di caratteri sono gestite *dinamicamente* e le procedure di accesso e il modo di trattare una catena favoriscono il trattamento sequenziale dei caratteri.
- Ad ogni catena si associa un campo descrittore che contiene l'indirizzo della catena la sua lunghezza e l'indice del carattere da trattare.
  Una variabile di tipo **text** inizializza questo campo.
  I parametri iniziali sono settati a **notext** (catena vuota).

Il tipo text

- Per creare una catena si utilizzano le funzioni: **blanks**(n) che crea una catena di n caratteri bianchi, **copy**(T) crea una copia di T.
- Le operazioni di assegnazione e di confronto tra gli elementi di una catena si possono fare mediante i puntatori (utilizzando quindi :-,= =, =/=) o direttamente sul contenuto della catena (utilizzando quindi :=, =, NE (notequal)).

Il tipo text

- L'operatore di concatenazione di una catena è **&** per esempio:

```
text T;
.................
OutText("scrivi" & T & "fine");
```

- La manipolazione del contenuto di una catena si fa con l'aiuto di procedure e funzioni del Simula e la notazione usata per accedere agli elementi del campo è la "dot notation".

Proprietà di una catena

- T.**length** fornisce la lunghezza della catena T,
- T.**pos** fornisce l'indice del carattere da trattare,
- T.**more** vero se T.**pos** > T.**length** falso altrimenti,
- T.**getchar** restituisce il carattere successivo e incrementa T.**pos**,
- T.**putchar**(C) inserisce il carattere C nella posizione **pos** e incrementa la **pos**,
- T.**setpos**(n) dà il valore n a T.**pos,**

Proprietà di una catena

- T.**getint** e T.**putint**(I) equivalenti a **InInt** e **OutInt,**
- T.**sub**(I,n) designa una sottocatena di T di lunghezza n inseribile in posizione I,
- T.**strip** restituisce la sottocatena
- T.**sub**(1,n) oppure tutti i caratteri dopo l'i-esimo sono bianchi.

Procedure e funzioni

Il modello più semplice di procedura è il seguente:

(TipoVariabile) **procedure** NomeProcedura (Parametro1,..., ParametroN,...);

TipoVariabile Parametro1,..., ParametroN,...;

begin ...;

end

Esempio di procedura

! Procedura che somma due numeri interi A e B e restituisce il risultato in C.;

```
begin
procedure Add(A,B,C); name C; integer A, B, C;
begin
C := A + B end of Add; integer ValA, ValB,
ValC; ValA := 1; ValB := 2; ValC := 0;
Add(ValA,ValB,ValC);
OutImage;OutInt(ValA,1); OutText(" + ");
OutInt(ValB,1); OutText(" = "); OutInt(ValC,1)
end
```

Esempio di funzione

```
!Funzione che somma due numeri interi A e B.;
    begin
    integer procedure Sum(A,B); integer A, B;
!i parametri sono passati per valore;
    begin
       Sum := A + B
    end of Sum;
    integer ValA, ValB, ValC;
    ValA := 1; ValB := 2; ValC := 0;
    ValC := Sum(ValA,ValB);
    OutImage; OutInt(ValA,1); OutText(" + "); OutInt(ValB,1);
    OutText(" = ");
    OutInt(ValC,1);
 end
```

Selezioni e Iterazioni

- Le principali istruzioni di selezione e iterazione sono riassunte nella seguente tabella:
- **If**
- **While**
- **For**

Istruzione IF

- Il comando **if** del Simula si serve della parola chiave **if** più o meno come si usa in italiano la parola "se". La frase "se il bar è aperto mi porti una birra, altrimenti un caffè" ha la stessa stesura in Simula:

if BAR = APERTO **then** BEV := BIRRA **else** BEV := CAFFE';

- La sintassi del commando **if** a cascata è

```
if Condizione1 then
begin
    if  Condizione2 then Statement
end
```

Istruzione While

In un programma Simula possiamo specificare un ciclo mediante un istruzione **while**.

La forma del comando **while** è la seguente:

while Condizione **do** Statement;

Statement è il corpo del ciclo **while;** viene valutata prima la condizione, se questa è falsa, l'esecuzione del comando **while** è terminata mentre se è vera Stantement viene eseguito un'altra volta; la condizione controllata nuovamente, e cosi via.

Istruzione For

Quando vogliamo eseguire più volte un'istruzione e il numero di ripetizioni non dipende dal risultato di istruzioni interne al ciclo, la costruzione più adatta è il ciclo **For**.

La sintassi del comando **For** è la seguente:

for I := 1 **step** +1 **until** 100 **do** Statement;

L’istruzione For

- **for** I := 100 **step** -1 **until** 1 **do** Statement;
- **for** I := 1, I + 3 **step** -1 **until** 1 **do** Statement;
- **for** I := 2, 3, 5, 7, 11 **step** 4 **until** 51 **do** Statement;
- **for** C := 'A', 'E', 'I', 'O', 'U', 'Y' **do** Statement;
- **for** P :- P1, P2, P3 **do** Statement