Corso di Laurea in INFORMATICA
Algoritmi e Strutture Dati

MODULO N. 5

Strutture lineari di dati: Code

Le code: specifiche e realizzazioni attraverso rappresentazioni sequenziali e collegate.

Questi lucidi sono stati preparati da Oronzo Altamura, Maria Francesca Costabile e Floriana Esposito, Università degli Studi di Bari, per uso didattico. Essi contengono materiale originale di proprietà dell'Università degli Studi di Bari e/o figure di proprietà di altri autori, società e organizzazioni di cui e' riportato il riferimento. Tutto o parte del materiale può essere fotocopiato per uso personale o didattico ma non può essere distribuito per uso commerciale. Qualunque altro uso richiede una specifica autorizzazione da parte dell'Università degli Studi di Bari e degli altri autori coinvolti.

CODE

UNA CODA E' UN TIPO ASTRATTO CHE CONSENTE DI RAPPRESENTARE UNA SEQUENZA DI ELEMENTI IN CUI E' POSSIBILE AGGIUNGERE ELEMENTI AD UN ESTREMO ("IL FONDO") E TOGLIERE ELEMENTI DALL'ALTRO ESTREMO ("LA TESTA").

TALE DISCIPLINA DI ACCESSO E' DETTA FIFO (First In First Out).

E' PARTICOLARMENTE ADATTA A RAPPRESENTARE SEQUENZE NELLE QUALI L'ELEMENTO VIENE ELABORATO SECONDO L'ORDINE DI ARRIVO (lista d'attesa, insieme di dispositivi in attesa di assegnazione di risorse, etc.)

CODE

SPECIFICA SINTATTICA

TIPI: coda, boolean, tipoelem

Operatori:

CREACODA :	()	→ coda
CODAVUOTA :	(coda)	→ boolean
LEGGICODA :	(coda)	→ tipoelem
FUORICODA :	(coda)	→ coda
INCODA :	(tipoelem, coda)	→ coda

SPECIFICA SEMANTICA

Tipi: CODA: insieme delle sequenze Q = $a_1, a_2, \ldots, a_n$ di elementi di tipo TIPOELEM;

BOOLEAN: insieme dei valori di verità;

Operatori:

CREACODA = Q'

Post: Q' = Λ sequenza vuota

CODAVUOTA(Q) = b

Post: b = VERO SE Q = Λ
b = FALSO ALTRIMENTI

LEGGICODA(Q) = a

Pre: Q = $a_1, a_2, \ldots, a_n$ E n ³ 1
Post: a = a_1

FUORICODA(Q) = Q'

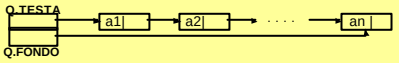
Pre: Q = $a_1, a_2, \ldots, a_n$ E n ³ 1
Post: Q' = $a_2, a_3, \ldots, a_n$ SE n > 1
Q' = Λ SE n = 1

INCODA (a,Q) = Q'

Pre: Q = $a_1, a_2, \ldots, a_n$ E n ³ 0
Post: Q' = $a_1, a_2, \ldots, a_n, a$

COME RAPPRESENTARE LE CODE

IN GENERALE LE POSSIBILI RAPPRESENTAZIONI DELLE CODE SONO ANALOGHE A QUELLE DELLE PILE CON L'ATTENZIONE CHE E' CONVENIENTE CONSENTIRE L'ACCESSO SIA ALL'ELEMENTO INSERITO PER PRIMO SIA ALL'ELEMENTO INSERITO PER ULTIMO.



REALIZZAZIONE CON PUNTATORI

LA CODA E' REALIZZATA CON n CELLE, LA PRIMA DELLE QUALI E' INDIRIZZATA DA UN PUNTATORE "TESTA" E L'ULTIMA DA UN PUNTATORE "FONDO".

LA CODA VUOTA E' INDIVIDUATA DAL VALORE NULLO Nil DEL PUNTATORE DI TESTA.

COME SEMPRE, DI SEGUITO E' DATA COME ESEMPIO UNA POSSIBILE IMPLEMENTAZIONE PASCAL:

```
TYPE
    cella = RECORD
              elemento : tipoelem;
              successivo : ^cella;
    END;
    coda = RECORD
              testa, fondo : ^cella;
    END;
```

```
PROCEDURE Creacoda (var Q: coda);
BEGIN
Q.testa : = NIL;
END;
FUNCTION Codavuota (var Q : coda): boolean;
BEGIN
Codavuota : = (Q.testa = NIL);
END;

FUNCTION Leggicoda (var Q: coda) : tipoelem;
BEGIN
   IF NOT Codavuota (Q) THEN
        Leggicoda : = Q.testa^.elemento
END;

PROCEDURE Fuoricoda (vat Q: coda);
   VAR    temp : ^cella;
BEGIN
  IF NOT Codavuota (Q) THEN BEGIN
      temp : = Q.testa;
      Q.testa := Q.testa^.successivo;
      DISPOSE (temp);
  END;
END;
```

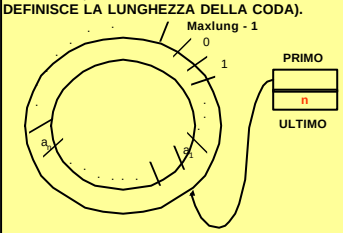
```
PROCEDURE Incoda(a: tipoelem ; var Q:coda);
BEGIN
  IF Codavuota (Q) THEN
    BEGIN
      new (Q.fondo);
      Q.testa : = Q.fondo;
    END
    ELSE BEGIN
      new (Q.fondo^.successivo);
      Q.fondo : = Q.fondo^.successivo;
    END;
   Q.fondo^.elementoo := a;
   Q.fondo^.successivo := NIL;
END;
```

PER LE CODE LA RAPPRESENTAZIONE SEQUENZIALE NON E' AGEVOLE COME PER LE PILE. E' UTILE GESTIRE L'ARRAY IN MODO CIRCOLARE.

REALIZZAZIONE CON VETTORE CIRCOLARE

IL VETTORE CIRCOLARE E' INTESO COME UN ARRAY DI maxlung ELEMENTI, CON INDICE DA 0 A maxlung – 1, IN CUI CONSIDERIAMO L'ELEMENTO DI INDICE 0 COME SUCCESSORE DI QUELLO DI INDICE maxlung - 1.

SI UTILIZZANO DUE VARIABILI PRIMO E ULTIMO. IL VALORE DI PRIMO INDICA LA POSIZIONE DELL'ARRAY IN CUI E' MEMORIZZATO L'ELEMENTO INSERITO PER PRIMO, ULTIMO SI RIFERISCE ALL'ULTIMO ELEMENTO INSERITO (OPPURE DEFINISCE LA LUNGHEZZA DELLA CODA).



UNA POSSIBILE IMPLEMENTAZIONE IN PASCAL

```
TYPE
Coda = RECORD
  elementi: ARRAY [ 0..maxlung -1]     OF tipoelem;
  PRIMO : 0..maxlung-1;
  ULTIMO: 0..maxlung;
END;
```

```
PROCEDURE Creacoda (var Q : coda);
BEGIN
        Q.primo := 0;
        Q.ultimo := 0;
END;
```

```
FUNCTION Codavuota (var Q: coda): boolean;
BEGIN
        Codavuota:= (Q.ultimo = 0);
END;
```

```
PROCEDURE Fuoricoda(var Q: coda);

BEGIN
        IF NOT Codavuota (Q) THEN BEGIN
                Q.primo := (Q.primo +1) mod maxlung;
                Q.ultimo := Q.ultimo –1;
        END;
END;
```

```
FUNCTION Leggicoda ( var Q: coda): tipoelem;
BEGIN
        IF NOT Codavuota (Q) THEN
                Leggicoda := Q.elementi[ Q.primo ];
END;
```

```
PROCEDURE Incoda (a: tipoelem; var Q: coda);
BEGIN
        IF Q.ultimo = maxlung THEN
                { 'CODA PIENA' }
        ELSE BEGIN
        Q.elementi [ (Q.primo + Q.ultimo) mod maxlung ] := a;
        Q.ultimo := Q.ultimo + 1;
        END;
END;
```

ESERCIZI SU CODE E PILE (PASCAL)

PROBLEMA

SI VUOLE REALIZZARE UN PROGRAMMA CHE PRENDE UNA CODA DI INTERI E RESTITUISCE UN'ALTRA CODA OTTENUTA DALLA PRIMA CONSIDERANDO SOLO VALORI POSITIVI.

Trascuriamo le dichiarative relative alla implementazione della coda

```
PROCEDURE ESTRAI (VAR Q: CODA; VAR Q1: CODA);

        var   e: TIPOELEM;
         BEGIN
           CREACODA(Q1);
           WHILE NOT CODAVUOTA (Q)
             BEGIN
                e := LEGGICODA (Q);
                IF e > 0
                      THEN INCODA (e, Q1)
                FUORICODA (Q);
              END;
        END;
```

SE VOLESSIMO CONSERVARE LA CODA ORIGINALE DOVREMMO USARE UNA CODA AUSILIARIA.

```
PROCEDURE ESTRAI1 (var Q:CODA ; var Q1:CODA);
VAR     QAUX: CODA;
        e: TIPOELEM;
BEGIN
        CREACODA (Q1);
        CREACODA (QAUX);
        WHILE NOT CODAVUOTA(Q) DO
        BEGIN
                e := LEGGICODA (Q);
                IF e > 0
                        THEN INCODA (e,Q1);
                FUORICODA (Q);
                INCODA (e, QAUX);
        END;
```

```
CREACODA (Q);
        WHILE NOT CODAVUOTA (QAUX) DO
        BEGIN
                e := LEGGICODA (QAUX);
                INCODA (e,Q);
                FUORICODA (QAUX);
        END;
          END;
```

RIPRISTINO DI Q

ESEMPIO

Aritmetica postfissa

LE ESPRESSIONI ARITMETICHE SONO SCRITTE IN NOTAZIONE **INFISSA**, CIOE' I SIMBOLI DELLE OPERAZIONI APPAIONO TRA GLI OGETTI SU CUI OPERANO.

NELLA NOTAZIONE **POSTFISSA** GLI OPERATORI SI PONGONO **DOPO** GLI OGGETTI SU CUI OPERANO

35 + 17 * (40 – 9) – 7

35 17 40 9 - * + 7 –

QUESTA NOTAZIONE E' DEFINITA POLACCA (SI DEVE AL MATEMATICO LUKASIEWICZ)

Aritmetica postfissa

Definizione

UNA NOTAZIONE POLACCA E' UNA QUALUNQUE SERIE DI OPERANDI ARITMETICI x, y, … E OPERATORI BINARI op (+,-,*,/) CHE SI PUO' FORMARE MEDIANTE LE REGOLE SEGUENTI:

- ***OGNI OPERANDO x E' UNA NOTAZIONE POLACCA***
- ***SE p1 E p2 SONO NOTAZIONI POLACCHE ALLORA p1p2 op E' UNA NOTAZIONE POLACCA***

ES:

48+63 corrisponde a p1 48 63 +

52*4 corrisponde a p2 52 4 *

p1 p2 - è una notazione polacca

Aritmetica postfissa

LE REGOLE PER CALCOLARE L'ESPRESSIONE POSTFISSA

- SCANDISCI L'ESPRESSIONE DA SINISTRA A DESTRA FINO A CHE RAGGIUNGI IL PRIMO OPERATORE
- APPLICA L'OPERATORE AI DUE OPERANDI CHE SONO ALLA SUA SINISTRA, OTTIENI IL RISULTATO CHE SOSTITUIRAI NELL'ESPRESSIONE AL POSTO DI OPERANDI E OPERATORE

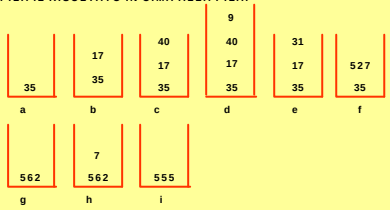
35 + 17 * (40 – 9) – 7

35 17 40 9 - * + 7 –

- 35 17 40 9 - * + 7 -
- 35 17 31 * + 7 -
- 35 527 + 7 -
- 562 7 -
- 555

ALGORITMO PER VALUTARE UNA ESPRESSIONE IN NOTAZIONE POLACCA

1. SCANDIRE L'ESPRESSIONE DA SINISTRA A DESTRA.
   APPENA E' RAGGIUNTO UN OPERANDO COMPI IL PASSO 2.
   APPENA E' RAGGIUNTO UN OPERATORE COMPI IL PASSO 3.
2. IN-PILA L'OPERANDO NELLA PILA DEGLI OPERANDI
3. RIMUOVI DALLA PILA I PRIMI DUE OPERANDI, APPLICA L'OPERATORE A QUESTI E IN-PILA IL RISULTATO IN CIMA ALLA PILA.



IL PROGRAMMA DI SEGUITO REALIZZA L'ALGORITMO, MA RIMANDA ALCUNE FUNZIONALITA' (VERIFICA CHE IL SIMBOLO SIA UN NUMERO) AD UN ULTERIORE SFORZO DI PROGRAMMAZIONE.

SI DISPONE DI UNA LISTA DI SIMBOLI E SI USA UNA PILA PER LA VALUTAZIONE DELLA NOTAZIONE POLACCA MEDIANTE PILA DI NUMERI REALI.

```
FUNCTION VALUTA_POLACCA (POST : LISTA_SIMBOLI) :real;
var
        VAL : PILA;
        NUMTOP, NUMSUC, RIS :real;
        t :char; tnum : real;

BEGIN
        CREAPILA(VAL);
        WHILE NOT LISTAVUOTA (POST) DO
        BEGIN
                t := LEGGILISTA (POST);
                IF ( t E' UN NUMERO) THEN INPILA(tnum, VAL)
                        ELSE BEGIN
                                NUMTOP := LEGGIPILA (VAL);
                                FUORIPILA(VAL);
                                NUMSUC := LEGGIPILA (VAL);
                                FUORIPILA(VAL);
                                RIS := NUMSUC 't' NUMTOP;
                                        INPILA (RIS, VAL);
                                END;
                        END;
        VALUTA_POLACCA := LEGGIPILA (VAL);
END;
```

DISPORRE DI UN MODO PER CALCOLARE ESPRESSIONI POSTFISSE SERVE A POCO SE NON SI DISPONE DI UN METODO CHE SIA IN GRADO DI CONVERTIRE LA LISTA DI SIMBOLI DI UNA ESPRESSIONE INFISSA IN QUELLI DELLA CORRISPONDENTE ESPRESSIONE POSTFISSA. IN QUESTO CASO E' CONVENIENTE UTILIZZARE UNA CODA COME RISULTATO DELLA CONVERSIONE.

Esempio ***Conversione da infissa a postfissa***

UTILIZZIAMO UNA PILA PER MEMORIZZARE I SIMBOLI DEGLI OPERATORI "IN SOSPESO" E UNA CODA PER IMMAGAZZINARE LA ESPRESSIONE POSTFISSA CHE VIENE COSTRUITA MAN MANO. AD OGNI OPERATORE NELLA ESPRESSIONE INFISSA VIENE ASSEGNATO UN ORDINE DI PRECEDENZA (MOLTIPLICAZIONE E DIVISIONE PRECEDENZA MASSIMA). ATTRIBUIAMO ALLA PARENTESI (LA PRECEDENZA MINIMA.

LISTA SIMBOLI	PILA	CODA
35 + 17 * (40 – 9) – 7	Æ	Æ
+ 17 * (40 – 9) – 7	Æ	35
17 * (40 – 9) – 7	+	35
* (40 – 9) – 7	+	17 35
(40 – 9) – 7	+ *	17 35
40 – 9) – 7	+ * (	17 35
– 9) – 7	+ * (	40 17 35
9) – 7	+ * (–	40 17 35
) – 7	+ * (–	9 40 17 35
– 7	+ * (	– 9 40 17 35
7	–	+*– 9 40 17 35
Æ	-	7+*- 9 40 17 35
Æ	Æ	-7+*- 9 40 17 35

```
PROCEDURE CONVERTI (INFISSA : LISTA; var CODA_POST : CODA );
Var     PILA_OP : PILA;
        top_elem : tipoelem;
PROCEDURE TRASFERISCI (var S: PILA; var C: CODA);
BEGIN
        top_elem := LEGGIPILA (S);
        FUORIPILA (S);
        INCODA (top_elem, C);
END;
BEGIN
        CREAPILA (PILA_OP);
        CREACODA(CODA_POST);
        WHILE NOT LISTAVUOTA (INFISSA) DO
        BEGIN
                t := LEGGILISTA (INFISSA);
                IF ( t è un numero ) THEN     INCODA (t , CODA_POST)
                ELSE
                        IF PILAVUOTA (PILA_OP) THEN
```

```
                INPILA (t, PILA_OP)
        ELSE
                IF (t = "'(" ) THEN   INPILA ( t , PILA_OP)
        ELSE
                IF (t = ")" ) THEN
                BEGIN
                        Top_elem := LEGGIPILA (PILA_OP);
                        WHILE (top_elem <> "(" ) DO
                                TRASFERISCI (PILA_OP , CODA_POST);
                        FUORIPILA (PILA_OP);
                END
        ELSE
                BEGIN
                        WHILE (PRIORITA' t <= PRIORITA' DI top_elem ) DO
                                TRASFERISCI (PILA_OP, CODA_POST);
                        INPILA ( t , PILA_OP);
                END;
        END;
        WHILE NOT FINEPILA (PILA_OP) DO
                TRASFERISCI (PILA_OP , CODA_POST);
END;
```