Corso di Laurea in INFORMATICA
Algoritmi e Strutture Dati

MODULO N. 4

Strutture lineari di dati: Pile

Le pile: specifiche e realizzazioni attraverso rappresentazioni sequenziali e collegate. Pile e procedure ricorsive

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.

PILE

UNA PILA È UNA SEQUENZA DI ELEMENTI DI UN CERTO TIPO IN CUI È POSSIBILE AGGIUNGERE O TOGLIERE ELEMENTI SOLO DA UN ESTREMO DELLA SEQUENZA (LA "TESTA").
PUÒ ESSERE VISTA COME UN CASO SPECIALE DI LISTA IN CUI L'ULTIMO ELEMENTO INSERITO È IL PRIMO AD ESSERE RIMOSSO (LIFO) E NON È POSSIBILE ACCEDERE AD ALCUN ELEMENTO CHE NON SIA QUELLO IN TESTA.

SPECIFICA SINTATTICA

Tipi: *pila, boolean, tipoelem*
Operatori:
CREAPILA: ( ) → *pila*
PILAVUOTA: (*pila*) → *boolean*
LEGGIPILA: (*pila*) → *tipoelem*
FUORIPILA: (*pila*) → *pila*
INPILA: (*tipoelem*, *pila*) → *pila*

SPECIFICA SEMANTICA

Tipi:
pila=insieme delle sequenze P=a_1, a_2, …, a_n di elementi di tipo tipoelem; solo l'elemento in testa alla squenzae' accessibile;
boolean=insieme dei valori di verità.
Operatori:
CREAPILA=P *EMPTY STACK*
POST: P=Λ (LA SEQUENZA VUOTA)
PILAVUOTA(P)=b *EMPTY*
POST: b=VERO SE P=Λ
b=FALSO, ALTRIMENTI
LEGGIPILA(P)=a *TOP*
PRE: P=a_1, a_2, …, a_n n≥1
POST: a=a_1
FUORIPILA(P)=P' *POP*
PRE: P=a_1, a_2, …, a_n n≥1
POST: P'= a_2, …, a_n SE n >1
P=Λ SE n =1
INPILA(A, P)= P' *PUSH*
PRE: P=a_1, a_2, …, a_n n≥0
POST: P'=a, a_1, a_2, …, a_n

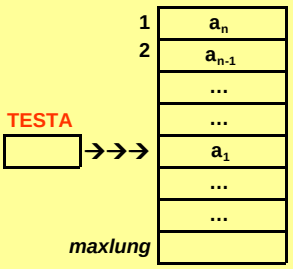
REALIZZAZIONI

LA PILA È' UN CASO PARTICOLARE DI LISTA E OGNI REALIZZAZIONE DESCRITTA PER LA LISTA FUNZIONA ANCHE PER LA PILA. POSSIAMO DEFINIRE LA CORRISPONDENZA TRA GLI OPERATORI.

CREAPILA	→	CREALISTA
PILAVUOTA(P)	→	LISTAVUOTA(P)
LEGGIPILA(P)	→	LEGGILISTA(PRIMOLISTA(P),P)
FUORIPILA(P)	→	CANCLISTA(PRIMOLISTA(P),P)
INPILA(a,P)	→	INSLISTA(a,PRIMOLISTA(P),P)

REALIZZAZIONE CON VETTORE

VANNO MEMORIZZATI GLI *n* ELEMENTI DELLA PILA, *IN ORDINE INVERSO*, NELLE PRIME *n* POSIZIONI DEL VETTORE, MANTENENDO UN CURSORE ALLA TESTA DELLA PILA.



COME ESEMPIO SEGUE LA CODIFICA IN PASCAL DEGLI OPERATORI SECONDO LA REALIZZAZIONE PROPOSTA.

```
const MAXLUNG=50;
type pila= record
     testa:0..MAXLUNG;
     elementi: array[1..MAXLUNG] of tipoelem
     end;

procedure creapila(var p:pila);
begin
     p.testa:=0;
end;

function pilavuota(var p:pila):boolean;
begin
     pilavuota:=(p.testa=0);
end;

procedure leggipila(var p:pila; var a:tipoelem);
begin
     if not pilavuota(p) then
          a:=p.elementi[p.testa];
          else
          writeln('LEGGIPILA A PILA VUOTA!!');
end;
```

```
procedure inpila(a:tipoelem; var p:pila);
begin
     if p.testa=MAXLUNG then
          writeln('VETTORE PIENO!!')
          else
          begin
          p.testa:=p.testa+1;
          p.elementi[p.testa]:=a;
          end;
end;

procedure fuoripila(var p:pila);
begin
     if not pilavuota(p) then
          p.testa:=p.testa-1
          else
          writeln('FUORIPILA A PILA VUOTA!!');
end;
```

7

NOTE ALLA REALIZZAZIONE CON VETTORE

CON QUESTA REALIZZAZIONE, OGNI OPERATORE RICHIEDE TEMPO COSTANTE PER ESSERE ESEGUITO.

LA RAPPRESENTAZIONE MEDIANTE ARRAY PRESENTA DUE SVANTAGGI:

- RICHIEDE DI DETERMINARE A PRIORI UN LIMITE AL NUMERO MASSIMO DI ELEMENTI DELLA PILA;
- LO SPAZIO DI MEMORIA UTILIZZATO È' INDIPENDENTE DAL NUMERO EFFETTIVO DI ELEMENTI.

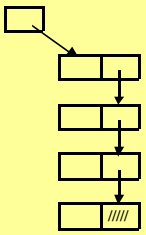
PER CONTRO:

+ AL CONTRARIO DELLE LISTE, GLI INSERIMENTI E LE CANCELLAZIONI NON RICHIEDONO SPOSTAMENTI PERCHÉ EFFETTUATI AD UNA ESTREMITÀ DELL'ARRAY.

8

REALIZZAZIONE CON PUNTATORI

CI RIFERIAMO ALLA PILA CON UN PUNTATORE ALLA CELLA CHE SI TROVA IN CIMA.



9

COME ESEMPIO SEGUE LA CODIFICA IN PASCAL DEGLI OPERATORI SECONDO LA REALIZZAZIONE PROPOSTA.

```
type posizione=^cella;
     cella= record
          elementi:tipoelem;
          successivo:posizione
          end;
     pila=posizione;

procedure creapila(var p:pila);
begin
     p:=nil;
end;

function pilavuota(var p:pila):boolean;
begin
     pilavuota:=(p=nil);
end;

procedure leggipila(var p:pila; var a:tipoelem);
begin
     if not pilavuota(p) then
          a:=p^.elementi
          else
          writeln('LEGGIPILA A PILA VUOTA!!');
end;
```

10

```
procedure inpila(a:tipoelem; var p:pila);
var temp: posizione;
begin
     new(temp);
     temp^.elementi:=a;
     temp^.successivo:=p;
     p:=temp;
end;

procedure fuoripila(var p:pila);
var temp: posizione;
begin
     if not pilavuota(p) then
          begin
          temp:=p;
          p:=p^.successivo;
          dispose(temp);
          end
          else
          writeln('FUORIPILA A PILA VUOTA!!');
end;
```

11

PILE E PROCEDURE RICORSIVE

UNA DELLE APPLICAZIONI PIÙ INTERESSANTI DELLE PILE RIGUARDA L'ESECUZIONE DI PROGRAMMI RICORSIVI.
L'ESECUZIONE DI UNA PROCEDURA RICORSIVA PREVEDE IL SALVATAGGIO DEI DATI SU CUI LAVORA LA PROCEDURA AL MOMENTO DELLA CHIAMATA RICORSIVA. TALI DATI VENGONO "RIPRISTINATI" QUANDO LA COMPUTAZIONE "INTERNA" TERMINA (MECCANISMO FIFO).

LE DIVERSE CHIAMATE ATTIVE SONO ORGANIZZATE IN UNA PILA. LA CHIAMATA PIÙ RECENTE È QUELLA CHE SI CONCLUDE PER PRIMA.

NELLA PILA VANNO SALVATI I PARAMETRI (E LE EVENTUALI VARIABILI LOCALI), IL PUNTO DI RITORNO, CIOÈ L'ETICHETTA DELLA ISTRUZIONE DA CUI RIPARTIRE AL TERMINE DELLA COMPUTAZIONE "INTERNA".

12

GRAZIE ALLE PILE È SEMPRE POSSIBILE, DATO UN PROGRAMMA RICORSIVO, TRASFORMARLO IN UNO ITERATIVO.

COME È NOTO, I PROGRAMMI RICORSIVI CORRISPONDONO A METODI SOLUTIVI PARTICOLARMENTE ADATTI A PROBLEMI PER I QUALI:

- LA SOLUZIONE DEL PROBLEMA DI RANGO *N* È DEFINIBILE IN TERMINI DELLA SOLUZIONE DEL PROBLEMA DI RANGO INFERIORE A *N;*
- È DEFINIBILE UNA SOLUZIONE PER ASSIOMA SUL PROBLEMA DI RANGO MINIMO (1 O 0).

ESEMPIO

LA SUCCESSIONE DI FIBONACCI È DEFINITA COME UNA SUCCESSIONE IL CUI *K-ESIMO* ELEMENTO È UGUALE ALLA SOMMA DEI DUE CHE LO PRECEDONO:

FIB(1)=1 FIB(2)=1 *livello assiomatico o di base*

FIB(K)=FIB(K-1)+FIB(K-2) PER K>2

13

LA SEQUENZA DI FIBONACCI SI PUÒ CREARE MEDIANTE UNA FUNZIONE RICORSIVA CHE CALCOLA IL *K-ESIMO* NUMERO DELLA SUCCESSIONE.

```
function FIB(K:integer):integer;
        var F:integer;
begin
        if (K=1) or (K=2) then
                F:=1
                else
                F:=FIB(K-1)+FIB(K-2); chiamata ricorsiva
        FIB(K):=F;
end;
```

GLI STATI DELLA PILA per K=4

1 FIB(4)
2 FIB(3), FIB(4)
3 FIB(2)=1, FIB(3), FIB(4)
4 FIB(3), FIB(4) → RIS. PARZ.=1
5 FIB(1)=1, FIB(3), FIB(4) → RIS. PARZ.=1
6 FIB(3)=2, FIB(4) → RIS. PARZ.=2
7 FIB(4)
8 FIB(2)=1, FIB(4) → RIS. PARZ.=2
9 FIB(4)=3

14

LA RICORSIONE PUÒ ESSERE USATA PER FORMALIZZARE UN'AMPIA CLASSE DI STRUTTURE DI DATI CHE MOSTRANO CARATTERISTICHE RICORSIVE NELLA STRUTTURA.

IN GENERALE, SE INTENDIAMO PER *SEQUENZA* DI DATI UN AGGREGATO IN CUI SIA RICONOSCIBILE UN PRIMO ELEMENTO, UN SECONDO… UN SUCCESSIVO, POSSIAMO DEFINIRLA RICORSIVAMENTE COME:

- UN AGGREGATO DI DATI EVENTUALMENTE VUOTO; *livello assiomatico*
- UN AGGREGATO NON VUOTO, IN CUI È INDIVIDUATO UN PRIMO ELEMENTO CHE INSIEME AI SUCCESSIVI COSTITUISCE ANCORA UNA SEQUENZA.

SE INDICHIAMO LA SEQUENZA DI n ELEMENTI CON $S_n=a_1, a_2, \ldots, a_n$ ALLORA S_n SARÀ:

PER N=0 $S_0=\{\}$ *livello assiomatico*

PER N>0 $\{S_{n-1}, A_n\}$

15

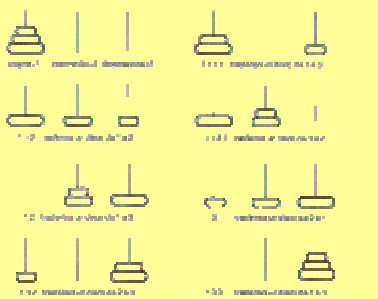
UN CLASSICO PROBLEMA RICORSIVO:
LA TORRE DI HANOI

IN UN MONASTERO TIBETANO CI SONO *N* DISCHI D'ORO, FORATI AL CENTRO, TUTTI DI DIAMETRO DIVERSO. I DISCHI SONO INFILATI IN UN PIOLO VERTICALE, ACCATASTATI PER DIAMETRO DECRESCENTE A PARTIRE DAL BASSO. I MONACI DEVONO SPOSTARE TUTTI I DISCHI DAL PIOLO IN CUI SI TROVANO IN UN ALTRO PIOLO, FORMANDO UNA CATASTA IDENTICA A QUELLA DI PARTENZA, SPOSTANDO UN DISCO ALLA VOLTA SENZA MAI SOVRAPPORRE UN DISCO PIU' GRANDE SU UN DISCO PIU' PICCOLO. È POSSIBILE USARE UN TERZO PIOLO DI APPOGGIO PER EFFETTUARE TRASFERIMENTI.

REGOLE

- SI PUÒ SPOSTARE UN SOLO DISCO PER VOLTA DA UN PIOLO AD UN ALTRO;
- DOPO OGNI MOSSA, PER OGNI PIOLO, I DISCHI DEVONO AVERE DIAMETRO DECRESCENTE DAL BASSO VERSO L'ALTO.

16



17

PER DIRE *SPOSTARE N DISCHI DAL PIOLO SORGENTE AL PIOLO DESTINAZIONE USANDO IL PIOLO AUSILIARIO* USIAMO LA NOTAZIONE

P(*NDISCHI*; *SORGENTE*, *AUSILIARIO*, *DESTINAZIONE*)

LA FORMALIZZAZIONE DELLA TORRE DI HANOI DI ORDINE N, CIOÈ SPOSTARE N DISCHI DAL PIOLO A AL PIOLO C, USANDO IL PIOLO B, PUÒ ESSERE:

P(N; A, B,C)

E' POSSIBILE UTILIZZARE LA SCOMPOSIZIONE RICORSIVA:
spostare N-1dischi da A a B usando C come ausiliario e spostare 1 disco da A a C usando B come ausiliario
LA SCOMPOSIZIONE DI ORDINE N È RIPORTABILE NEI TERMINI DELLO STESSO PROBLEMA DI ORDINE N-1 E LO SPOSTAMENTO DI UN SOLO DISCO COSTITUISCE IL LIVELLO ASSIOMATICO.

P_1 P(N-1; A, C, B)
P_2 P(1; A, B, C)
P_3 P(N-1; B, A, C)

18

NEL CASO DI N=3 I SOTTOPROBLEMI SONO

P_1 P(2; A, C, B) DECOMPONIBILE IN

P_{11} P(1; A, B, C)

P_{12} P(1; A, C, B)

P_{13} P(1; C, A, B)

P_2 P(1; A, B, C)

P_3 P(2; B, A, C)

IN DEFINITIVA IL PROBLEMA PER N=3 SARÀ SCOMPOSTO NEI SEGUENTI PROBLEMI PRIMITIVI:

P_{11} P(1; A, B, C)

P_{12} P(1; A, C, B)

P_{13} P(1; C, A, B)

P_2 P(1; A, B, C)

P_{31} P(1; B, C, A)

P_{32} P(1; B, A, C)

P_{33} P(1; A, B, C)

UN PROGRAMMA IN PASCAL PER LA TORRE DI HANOI

STRATEGIA

A) MUOVI GLI N1 DISCHI DAL PIOLO A SINISTRA AL PIOLO DI MEZZO USANDO COME PIOLO AUSILIARIO QUELLO DI DESTRA;

B) MUOVI IL DISCO CHE RIMANE DAL PIOLO DI SINISTRA A QUELLO DI DESTRA;

C) MUOVI GLI N-1 DISCHI DAL PIOLO DI CENTRO A QUELLO DI DESTRA USANDO COME PIOLO AUSILIARIO QUELLO DI SINISTRA.

```
procedure MOVE(N: intero; SORGENTE,
        AUSILIARIO, DESTINAZIONE:pole);
begin
      if N=1 then
        muovi_un_disco_da_sorgente_a_destinazione
        else
        begin
        MOVE(N-1,SORGENTE, DESTINAZIONE, AUSILIARIO);
        muovi_un_disco_da_sorgente_a_destinazione;
        MOVE(N-1, AUSILIARIO, SORGENTE, DESTINAZIONE);
        end;
end;
```



```
program TORRE(input, output);
type pole=(sinistra, centro, destra);
     interopos=1..MAXINT;
var NUMERODISCHI: interopos;

procedure MOVE(N: interopos; SORGENTE,
        AUSILIARIO, DESTINAZIONE:pole);

    procedure MUOVISORG_A_DESTIN;
        procedure PRINTPOLE(P:pole);
        begin
            case P of
                sinistra: write('SINISTRA');
                centro: write('CENTRO');
                destra: write('DESTRA');
            end;
        end;    (*PRINTPOLE*)
    begin
        write('MUOVI UN DISCO DA ');
        PRINTPOLE(SORGENTE);
        write(' A ');
        PRINTPOLE(DESTINAZIONE);
        write('USANDO COME AUSILIARIO ');
```



```
        PRINTPOLE(AUSILIARIO);
        writeln;
    end;    (* MUOVISORG_A_DESTIN*)

begin
    if N=1 then
        MUOVISORG_A_DESTIN
        else
        begin
        MOVE(N-1,SORGENTE, DESTINAZIONE, AUSILIARIO);
        MUOVISORG_A_DESTIN;
        MOVE(N-1, AUSILIARIO, SORGENTE, DESTINAZIONE);
        end;
end;    (*MOVE*)


begin    (*MAIN*)
    read(NUMERODISCHI);
    write('PER ',NUMERODISCHI:3,' DISCHI I MOVIMENTI ');
    writeln('RICHIESTI SONO:'); writeln;
    MOVE(NUMERODISCHI,SINISTRA,CENTRO,DESTRA);
end.
```


PER TRASFORMARE UNA PROCEDURA RICORSIVA IN UNA ITERATIVA BISOGNA:

A) CREARE UNA PILA DOPO IL BEGIN INIZIALE;

B) SOSTITUIRE OGNI CHIAMATA RICORSIVA CON UNA SEQUENZA DI ISTRUZIONI CHE:

- *SALVANO NELLA PILA I VALORI DEI PARAMETRI DELLE VARIABILI LOCALI E L'ETICHETTA DELLA ISTRUZIONE SEGUENTE ALLA CHIAMATA RICORSIVA;*
- *ASSEGNANO AI PARAMETRI GLI OPPORTUNI VALORI;*
- *EFFETTUANO UN SALTO ALL'ISTRUZIONE CHE SEGUE LA CREAZIONE DELLA PILA.*

C) INTRODURRE PRIMA DELL'END FINALE ISTRUZIONI CHE, NEL CASO LA PILA NON SIA VUOTA, ESTRAGGONO DALLA PILA I VALORI SALVATI E SALTANO ALLA ISTRUZIONE LA CUI ETICHETTA È UGUALE AL PUNTO DI RITORNO.

TALI REGOLE SONO VALIDE SE I PARAMETRI SI INTENDONO PASSATI PER VALORE.

QUANDO PARAMETRI E VARIABILI LOCALI SONO DI TIPO DIVERSO NELLA PILA È CONVENIENTE MEMORIZZARE UN

RECORD DI ATTIVAZIONE

CHE CONTIENE LO STATO DELLA COMPUTAZIONE SOSPESA.

25

LA TORRE DI HANOI IN VERSIONE ITERATIVA

```
type tipoelem=record     (*RECORD DI ATTIVAZIONE*)
       NUMERODISCHI: interopos;
       PIOLORIG, PIOLODEST, PIOLOAUS:pole;
       RITORNO: integer
       end;

procedure MOVE(N: interopos; SORGENTE,
          AUSILIARIO, DESTINAZIONE:pole);
label 1,2,3,4
var S:pila; TEMP:pole; STATO:tipoelem;
begin
     creapila(S);
1:   if N=1 then
          begin
          MUOVISORG_A_DESTIN;
          goto 3;
          end;
     with STATO do
          begin
          NUMERODISCHI:=N;
          PIOLORIG:=SORGENTE;
```

26

```
          PIOLOAUS:=AUSILIARIO;
          PIOLODEST:=DESTINAZIONE;
          RITORNO:=2
          end;
     inpila(STATO,S);
     N:=N-1;
     TEMP:=DESTINAZIONE;
     DESTINAZIONE:=AUSILIARIO;
     AUSILIARIO:=TEMP;
     goto1;
2:   MUOVISORG_A_DESTIN;
     with STATO do
          begin
          NUMERODISCHI:=N;
          PIOLORIG:=SORGENTE;
          PIOLOAUS:=AUSILIARIO;
          PIOLODEST:=DESTINAZIONE;
          RITORNO:=3
          end;
     inpila(STATO,S);
     N:=N-1;
     TEMP:=SORGENTE;
     SORGENTE:=AUSILIARIO;
```

27

```
     AUSILIARIO:=TEMP;
     goto1;
3:   if not pilavuota(S) then
          begin
          STATO:=leggipila(S);
          fuoripila(S);
          N:=STATO.NUMERODISCHI;
          SORGENTE:=STATO.PIOLORIG;
          DESTINAZIONE:=STATO.PIOLODEST;
          AUSILIARIO:=STATO.PIOLOAUS;
          case STATO.RITORNO of
               2: goto 2;
               3: goto 3;
               end;
          end;
end;
```

28

ESERCIZI

1. **I DATI ANAGRAFICI RELATIVI AGLI STUDENTI DEL CORSO DI ASD SONO MEMORIZZATI IN UNA PILA.**
   - **DEFINIRE L'ALGEBRA DELLA STRUTTURA DI DATI PILA.**
   - **SCRIVERE UN PROGRAMMA CHE RICERCA I DATI RELATIVI AD UNO SPECIFICO STUDENTE E, SE PRESENTE, LO CANCELLA DALLA PILA RICOMPATTANDOLA.**

2. **SCRIVERE UN PROGRAMMA CHE ACQUISISCA E VALUTI UN'ESPREDSSIONE ALGEBRICA IN NOTAZIONE POSTFISSA (NOTAZIONE POLACCA INVERSA). GLI OPERATORI AMMESSI SONO SOLO +, -, *, / E POSSONO ESSERE SOLO INTERI.**
   - **ESEMPIO: 125 3 7 / +**

```
                  7
            3     3     2
125  125  125   125   127
```

29