Modulo: Strutture dati dinamiche lineari [P2_08]

Unità didattica: Implementazione C di una lista lineare: organizzazione dei dati e funzioni per gestirli [6-C]

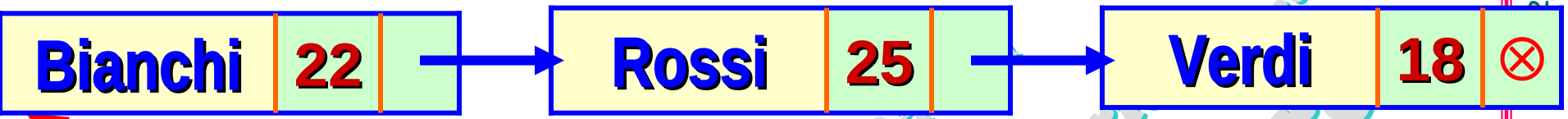
Titolo: Implementazione C di una lista lineare: organizzazione dei dati e funzioni per gestirli

Argomenti trattati:

- ✓ Organizzazione dei dati per la gestione di una lista
- ✓ Inserimento, eliminazione di un nodo alla testa di una lista
- ✓ Inserimento, eliminazione di un nodo dopo il nodo corrente
- ✓ Algoritmo ricorsivo per la costruzione di una lista

Prerequisiti richiesti: fondamenti della programmazione C, struttura dati autoriferente dinamica

In C: operazioni sulle liste (con funzioni)



testa

organizzazione dei dati

```
typedef struct{
                    char nome[20];
                    short eta;
                    } INFO_FIELD;
struct PERSONA
      {INFO_FIELD info;
       struct PERSONA *p_next;
      };
...
void main()
{struct PERSONA *head, *punt;
```

esempio di dati

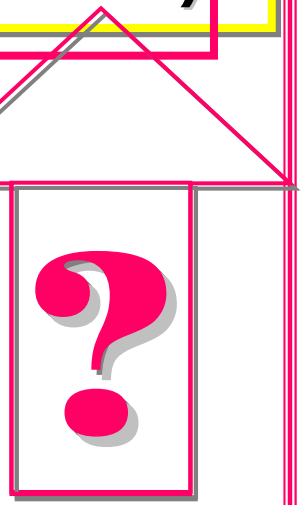
```
char *name[]={"Bianchi Roberto",…};
short  age[]={22,25,18,…};
```

prototipo delle funzioni

```
struct PERSONA *crea_lista()
void insl_testa(INFO_FIELD ,struct PERSONA **)
void insl_nodo(INFO_FIELD ,struct PERSONA **)
```

oppure, equivalentemente, ...

```
struct PERSONA * insl_testa(INFO_FIELD ,struct PERSONA *)
struct PERSONA * insl_nodo(INFO_FIELD ,struct PERSONA *)
```



```
/* crea lista vuota */
struct PERSONA *crea_lista()
   {struct PERSONA *testa;
     testa=NULL;
     return testa;
   }
```

chiamata:

```
head=crea_lista();
```

```
/* inserisce dato in testa alla lista */
void insl_testa(INFO_FIELD dato,
                          struct PERSONA **p_head)
   {struct PERSONA *ptr;
    ptr=calloc(1, sizeof(struct PERSONA));
    ptr->info=dato;
    ptr->p_next=*p_head;
    *p_head=ptr;/*aggiorna head al nuovo nodo*/
   }
```

il puntatore è un parametro di uscita della function

puntatore a puntatore
(punta a head – head è puntatore)

chiamata:

```
insl_testa(nuovodato,&head);
```

```
/* inserisce dato dopo nodo corrente */
void insl_nodo(INFO_FIELD dato,
                              struct PERSONA **p_punt )
   {struct PERSONA *ptr;
    ptr=calloc(1, sizeof(struct PERSONA));
    ptr->info=dato;
    ptr->p_next=(*p_punt)->p_next;
    (*p_punt)->p_next=ptr;
    *p_punt=ptr;/*aggiorna punt a nodo corrente*/
   }
```

il puntatore è un parametro di uscita della function

puntatore a puntatore

(punta a punt – punt è puntatore)

chiamata:

```
insl_nodo(nuovodato, &punt);
```

Esercizio: realizzare le seguenti *function C* per la gestione di una lista

prototipo delle funzioni

```
INFO_FIELD legge_infonodo(struct PERSONA *)
```

restituisce il campo informazione del nodo corrente

chiamata: `info=legge_infonodo(pt_nodo);`

```
void elim_testa(struct PERSONA **)
```

elimina nodo di testa

chiamata: `elim_testa(&p_head);`

```
void elim_nodo(struct PERSONA *)
```

elimina nodo successore

chiamata: `elim_nodo(pt_prec);`

Esercizio

1

Realizzare la gestione di una lista lineare mediante menù (voci: visualizzazione mediante visita, inserimento in testa, inserimento in mezzo, eliminazione in testa, eliminazione in mezzo) implementando la lista lineare con una struttura autoriferente dinamica.

Programma ricorsivo per la costruzione di una lista

A partire da un array contenente le informazioni da inserire nei nodi, si costruisce una lista lineare.

In particolare il seguente programma, a partire da una stringa, crea ricorsivamente una lista dove ogni nodo contiene un carattere.

```
#include <stdio.h>
#include <stdlib.h>
typedef char DATA;
struct linked_list
     { DATA d;
       struct linked_list *next;
     };
typedef struct linked_list ELEMENT;

typedef ELEMENT *LINK;

LINK array_to_list(DATA []);  ...
```

definisce il tipo dei nodi della lista

prototipo

```
...    void main()
       { DATA c_arr[]="questa e` una stringa di prova!";
         LINK head_list, p_list;
         head_list = array_to_list(c_arr);
         p_list=head_list;
         while (p_list != NULL)
            { putchar(p_list->d);
              p_list=p_list->next;
            }
         puts("");
       }

LINK array_to_list(DATA s[])
       {
         LINK head;
         if (s[0] == '\0') return NULL;
         else
            { head = malloc(sizeof(ELEMENT));
              head->d = s[0];
              head->next = array_to_list(s+1);
              return head;
            }
       }
```

visita

se “fine stringa”

Esercizi

2

Riscrivere, in versione iterativa, la precedente versione ricorsiva per la costruzione di una lista.

[liv. 2]

3

Scrivere una *function C* per costruire una lista ordinata in ordine alfabetico a partire da un elenco di nomi in ordine casuale, come nell'esempio a fianco.

[liv. 2]

Anna
Mario
Giuseppe
Angela
Valeria
Fabrizio
Marianna
Giovanni
Patrizia
Valentina
Sara