POLITECNICO DI MILANO

INFORMATICA

Dal programma per tabelle con array al programma per tabelle dinamiche

```
#include <iostream>
using namespace std;
#include <iomanip.h>

const int FALSO = 0;
const int VERO = 1;
struct Data
  { int giorno,
        mese,
        anno;
  };
const int MAXIDENT = 10;
struct Studente
  { int matricola;
    char cognome[MAXIDENT],
         nome[MAXIDENT],
         sesso;
    Data dataNascita;
  };
const int DIM = 8;
struct InsStudenti
  { int num;
    Studente elenco[DIM];
  };
```

```
#include <iostream>
using namespace std;
#include <iomanip.h>

const int FALSO = 0;
const int VERO = 1;
struct Data
  { int giorno,
        mese,
        anno;
  };
const int MAXIDENT = 10;
struct Studente
  { int matricola;
    char cognome[MAXIDENT],
         nome[MAXIDENT],
         sesso;
    Data dataNascita;
  };
const int DIM = 8;
struct Nodo
  { Studente datiStud;
    Nodo *nextPtr;
  };
```

```
const int DIM = 8;
struct InsStudente
  { int num;
    Studente elenco[DIM];
  };

InsStudente inizializzaTabella();
void inserisciSeNonEsiste(Studente, InsStudente &);
void eliminaSeEsiste(int, InsStudente &);
void cercaSeEsisteEPosizione(int, InsStudente &,
                             int &, int &);
void stampaTabella(InsStudente &);

int main()
{
  const char INSERISCI = 'i';
  const char ELIMINA = 'e';
  const char FINE = 'f';
  char operazione ;
  Studente nuovoStudente;
```

```
const int DIM = 8;
struct InsStudente
  { int num;
    Studente elenco[DIM];
  };

Nodo *inizializzaTabella();
void inserisciSeNonEsiste(Studente, InsStudente &);
void eliminaSeEsiste(int, InsStudente &);
void cercaSeEsisteEPosizione(int, InsStudente &,
                             int &, int &);
void stampaTabella(InsStudente &);

int main()
{
  const char INSERISCI = 'i';
  const char ELIMINA = 'e';
  const char FINE = 'f';
  char operazione ;
  Studente nuovoStudente;
```

```
const int DIM = 8;
struct InsStudente
  { int num;
    Studente elenco[DIM];
  };

Nodo *inizializzaTabella();
void inserisciSeNonEsiste(Studente, InsStudente &);
void eliminaSeEsiste(int, InsStudente &);
void cercaSeEsisteEPosizione(int, InsStudente &,
                             int &, int &);
void stampaTabella(InsStudente &);

int main()
{
  const char INSERISCI = 'i';
  const char ELIMINA = 'e';
  const char FINE = 'f';
  char operazione ;
  Studente nuovoStudente;
```

```
const int DIM = 8;
struct InsStudente
  { int num;
    Studente elenco[DIM];
  };

Nodo *inizializzaTabella();
void inserisciSeNonEsiste(Studente, Nodo *&);
void eliminaSeEsiste(int, Nodo *&);
void cercaSeEsisteEPosizione(int, Nodo *&,
                             int &, Nodo *&);
void stampaTabella(Nodo *&);

int main()
{
  const char INSERISCI = 'i';
  const char ELIMINA = 'e';
  const char FINE = 'f';
  char operazione ;
  Studente nuovoStudente;
```

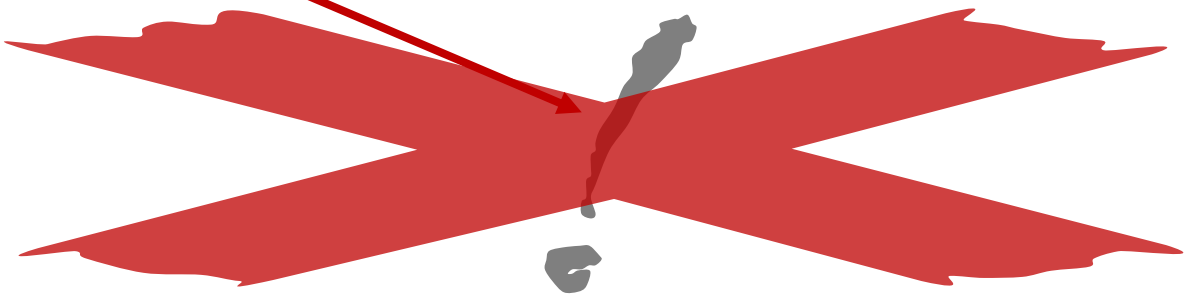
```
int main()
{
  const char INSERISCI = 'i';
  const char ELIMINA = 'e';
  const char FINE = 'f';
  char operazione ;
  Studente nuovoStudente;
  int matricola;
  InsStudenti classe;
  classe = inizializzaTabella();
  stampaTabella(classe);
  cout << setw(50) << "acquisizione operazioni"
       << endl << endl;
  cout << "operazione? (i per inserimento,"
       << " e per eliminazione,"
       << " f per fine" << " ): ";
  cin >> operazione;
  while (operazione != FINE)
    { if (operazione == INSERISCI)
        { cout << " dati del nuovo studente" << endl;
          cout << " matricola: ";
            cin >> nuovoStudente.matricola;
          cout << " cognome: "; cin >> nuovoStudente.cognome;
          cout << " nome: "; cin >> nuovoStudente.nome;
          cout << " sesso( M/F ): ";
```

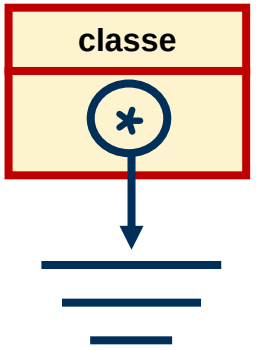
```
int main()
{
  const char INSERISCI = 'i';
  const char ELIMINA = 'e';
  const char FINE = 'f';
  char operazione ;
  Studente nuovoStudente;
  int matricola;
  InsStudenti classe;
  classe = inizializzaTabella();
  stampaTabella(classe);
  cout << setw(50) << "acquisizione operazioni"
       << endl << endl;
  cout << "operazione? (i per inserimento,"
       << " e per eliminazione,"
       << " f per fine" << " ): ";
  cin >> operazione;
  while (operazione != FINE)
    { if (operazione == INSERISCI)
        { cout << " dati del nuovo studente" << endl;
          cout << " matricola: ";
            cin >> nuovoStudente.matricola;
          cout << " cognome: "; cin >> nuovoStudente.cognome;
          cout << " nome: "; cin >> nuovoStudente.nome;
          cout << " sesso( M/F ): ";
```

```
int main()
{
  const char INSERISCI = 'i';
  const char ELIMINA = 'e';
  const char FINE = 'f';
  char operazione ;
  Studente nuovoStudente;
  int matricola;
  Nodo *classe;
  classe = inizializzaTabella();
  stampaTabella(classe);
  cout << setw(50) << "acquisizione operazioni"
       << endl << endl;
  cout << "operazione? (i per inserimento,"
       << " e per eliminazione,"
       << " f per fine" << " ): ";
  cin >> operazione;
  while (operazione != FINE)
    { if (operazione == INSERISCI)
        { cout << " dati del nuovo studente" << endl;
          cout << " matricola: ";
            cin >> nuovoStudente.matricola;
          cout << " cognome: "; cin >> nuovoStudente.cognome;
          cout << " nome: "; cin >> nuovoStudente.nome;
          cout << " sesso( M/F ): ";
```

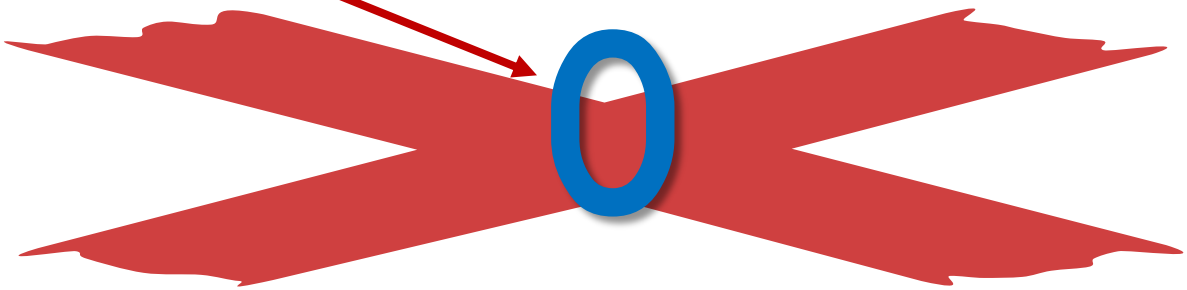
```
Nodo *inizializzaTabella()
{
  InsStudenti tab =
                  { 4,
                    { {7, "rossi", "marco",'M',{5,10,96}},
                      {48,"verdi", "anna", 'F',{4, 7,95}},
                      {63,"neri", "remo", 'M',{5, 8,96}},
                      {84,"gialli","carla",'F',{5,11,95}}
                    }
                  };
  return tab;
}
```

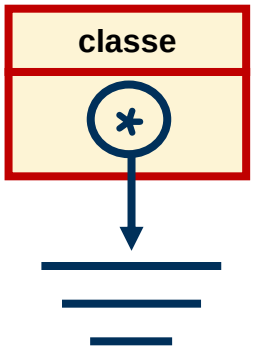
```
int main()
{  ...
   classe = inizializzaTabella();
   ...

Nodo *inizializzaTabella()
{
   InsStudenti tab =
                  {




                  };
   return tab;
}
```





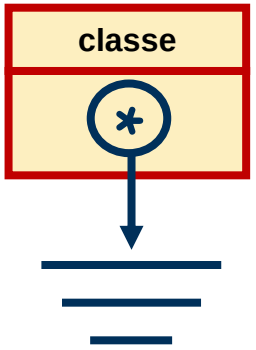
```
int main()
{  ...
   classe = inizializzaTabella();
   ...

Nodo *inizializzaTabella()
{
   InsStudenti tab =
                  {
                  };
   return tab;
}
```



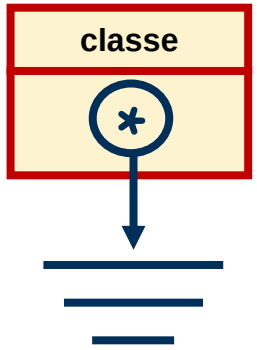


```
int main()
{  ...
   classe = inizializzaTabella();
   ...

Nodo *inizializzaTabella()
{
   InsStudenti tab =
                  {
                  };
   return tab;
}
```

null



```
int main()
{  ...
   classe = inizializzaTabella();
   ...

Nodo *inizializzaTabella()
{
   return 0;
}
```

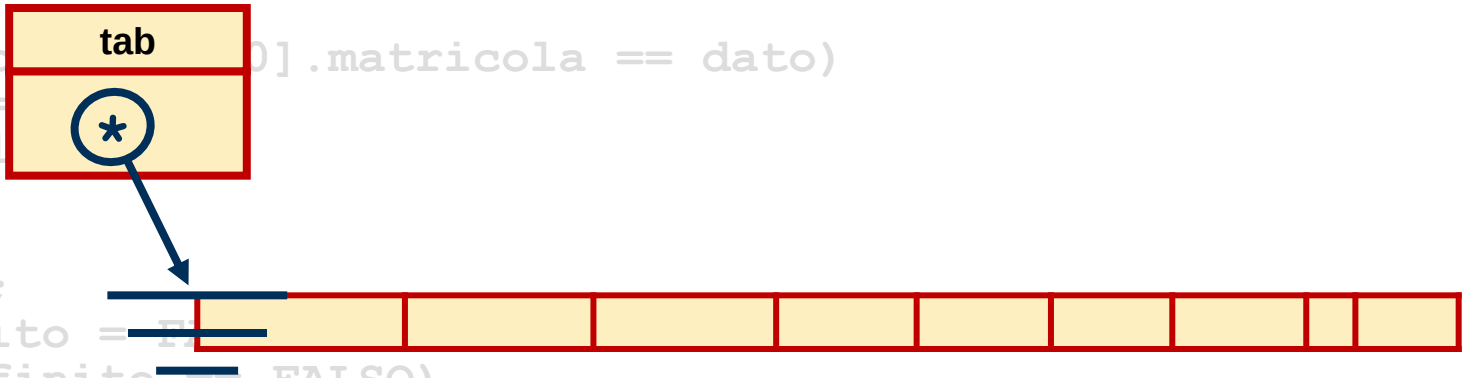


```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.num == 0) || (tab.elenco[0].matricola > dato))
    pos = -1;
  else if (tab.elenco[0].matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.num == 0) || (tab.elenco[0].matricola > dato))
    pos = -1;
  else if (tab.elenco[0].matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.num == 0) || (tab.elenco[0].matricola > dato))
    pos = -1;
  else if (tab.elenco[0].matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.num == 0) || (tab.elenco[0].matricola > dato))
    pos = -1;
  else if (tab.elenco[0].matricola == dato)
    { esiste = VERO;
      pos = 0;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```



```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab == 0) || (tab.elenco[0].matricola > dato))
    pos = -1;
  else if (tab.elenco[0].matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab == 0) || (tab.elenco[0].matricola > dato))
    pos = -1;
  else if (tab.elenco[0].matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab == 0) || (tab->datiStud.matricola > dato))
    pos = -1;
  else if (tab->datiStud.matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab == 0) || (tab->datiStud.matricola > dato))
    pos = -1;
  else if (tab->datiStud.matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab == 0) || (tab->datiStud.matricola > dato))
    pos = 0;
  else if (tab->datiStud.matricola == dato)
    { esiste = VERO;
      pos = 0;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab == 0) || (tab->datiStud.matricola > dato))
    pos = 0;
  else if (tab->datiStud.matricola == dato)
    { esiste = VERO;
      pos = 0;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab == 0) || (tab->datiStud.matricola > dato))
    pos = 0;
  else if (tab->datiStud.matricola == dato)
    { esiste = VERO;
      pos = 0;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab == 0) || (tab->datiStud.matricola > dato))
    pos = 0;
  else if (tab->datiStud.matricola == dato)
    { esiste = VERO;
      pos = 0;
    }
  else
    { pos = tab;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.num == 0) || (tab.elenco[0].matricola > dato))
    pos = -1;
  else if (tab.elenco[0].matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.num == 0) || (tab.elenco[0].matricola > dato))
    pos = -1;
  else if (tab.elenco[0].matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.nu
    pos = -1;
  else if (ta
    { esiste
      pos = -
    }
  else
    { pos = 0
      int fin
      while (
        { if
            f
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

classe

num	elenco						
4	matricola	cognome	nome	sesso	dataNascita		
					giorno	mese	anno
0	7	Rossi	Marco	M	5	10	95
1	48	Neri	Anna	F	4	7	95
2	63	Verdi	Remo	M	5	8	96
3	84	Gialli	Carla	F	5	11	95
4							

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.nu
    pos = -1;
  else if (ta
    { esiste
      pos = -
    }
  else
    { pos = 0
      int fin
      while (
        { if
            f
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1]          a > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

classe

num		elenco						
		matricola	cognome	nome	sesso	dataNascita		
						giorno	mese	anno
4	0	7	Rossi	Marco	M	5	10	95
	1	48	Neri	Anna	F	4	7	95
	2	63	Verdi	Remo	M	5	8	96
	3	84	Gialli	Carla	F	5	11	95
	4							

pos
x

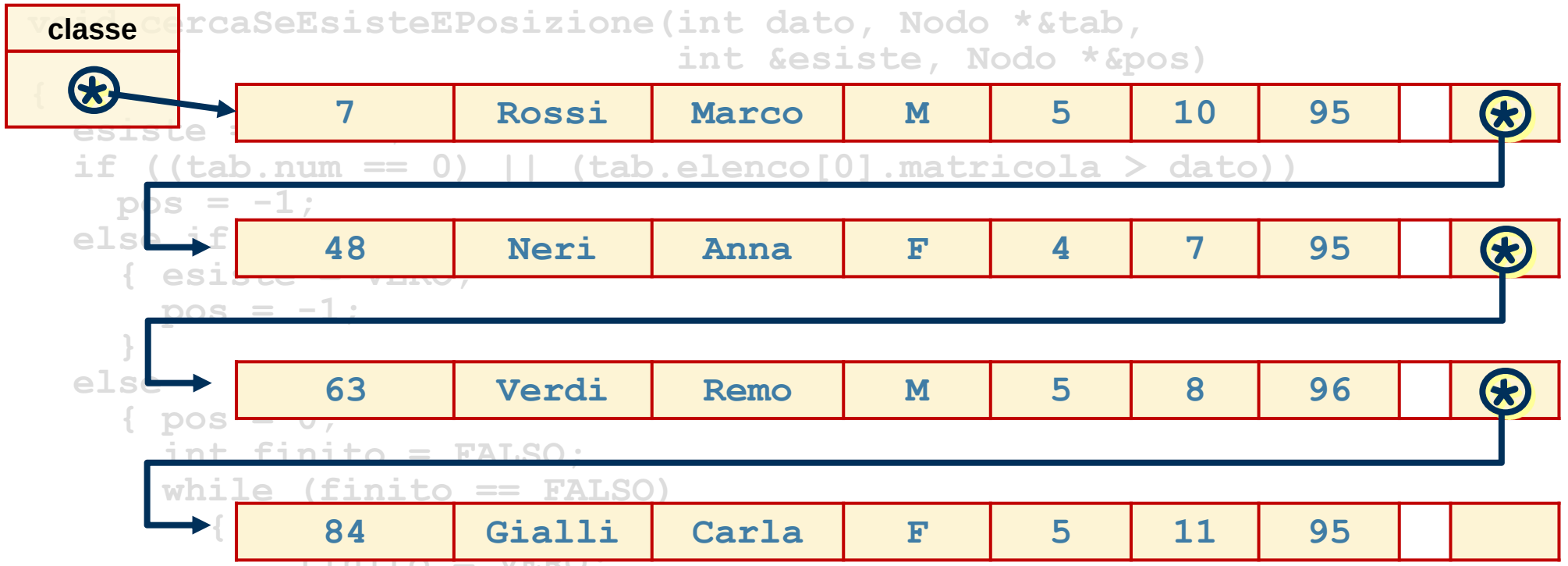
```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.nu
    pos = -1;
  else if(ta
    { esiste
      pos =
    }
  else
    { pos = 0
      int fin
      while (
        { if
            f
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1]          a > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

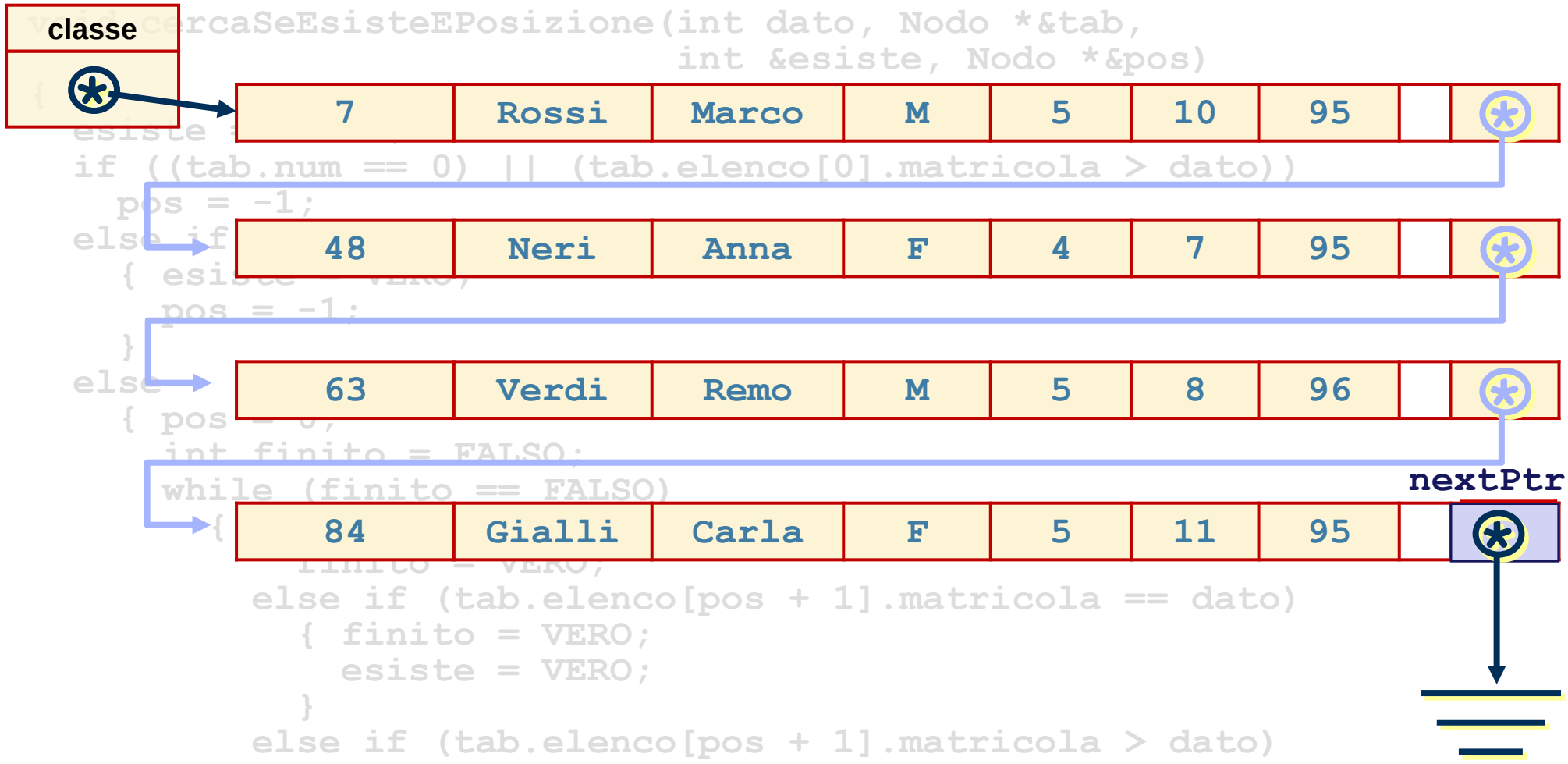
tab.num

classe							
num	elenco						
4	matricola	cognome	nome	sesso	dataNascita		
					giorno	mese	anno
0	7	Rossi	Marco	M	5	10	95
1	48	Neri	Anna	F	4	7	95
2	63	Verdi	Remo	M	5	8	96
3	84	Gialli	Carla	F	5	11	95
4							

pos

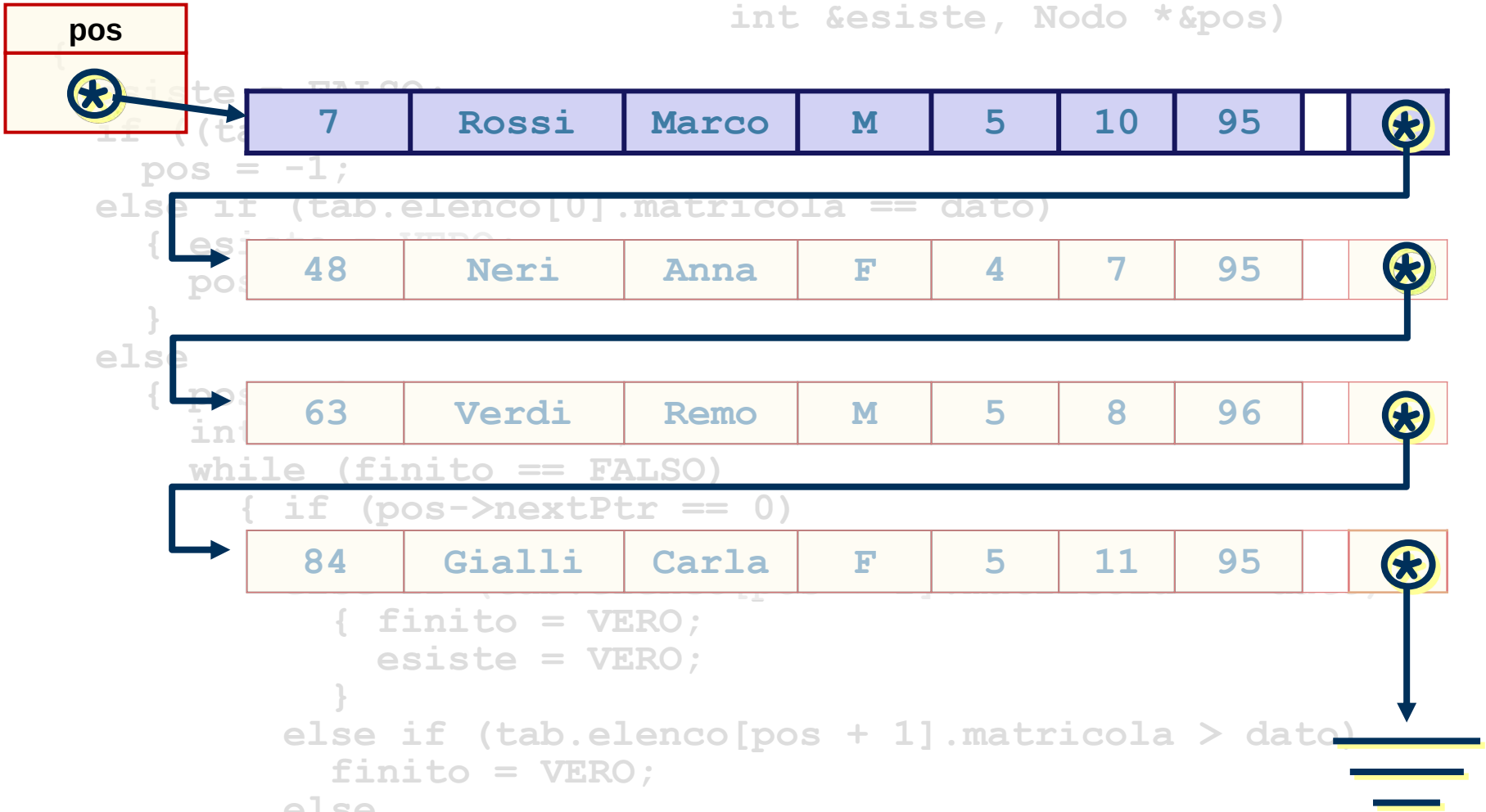
```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.num == 0) || (tab.elenco[0].matricola > dato))
    pos = -1;
  else if (tab.elenco[0].matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos == tab.num -1)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

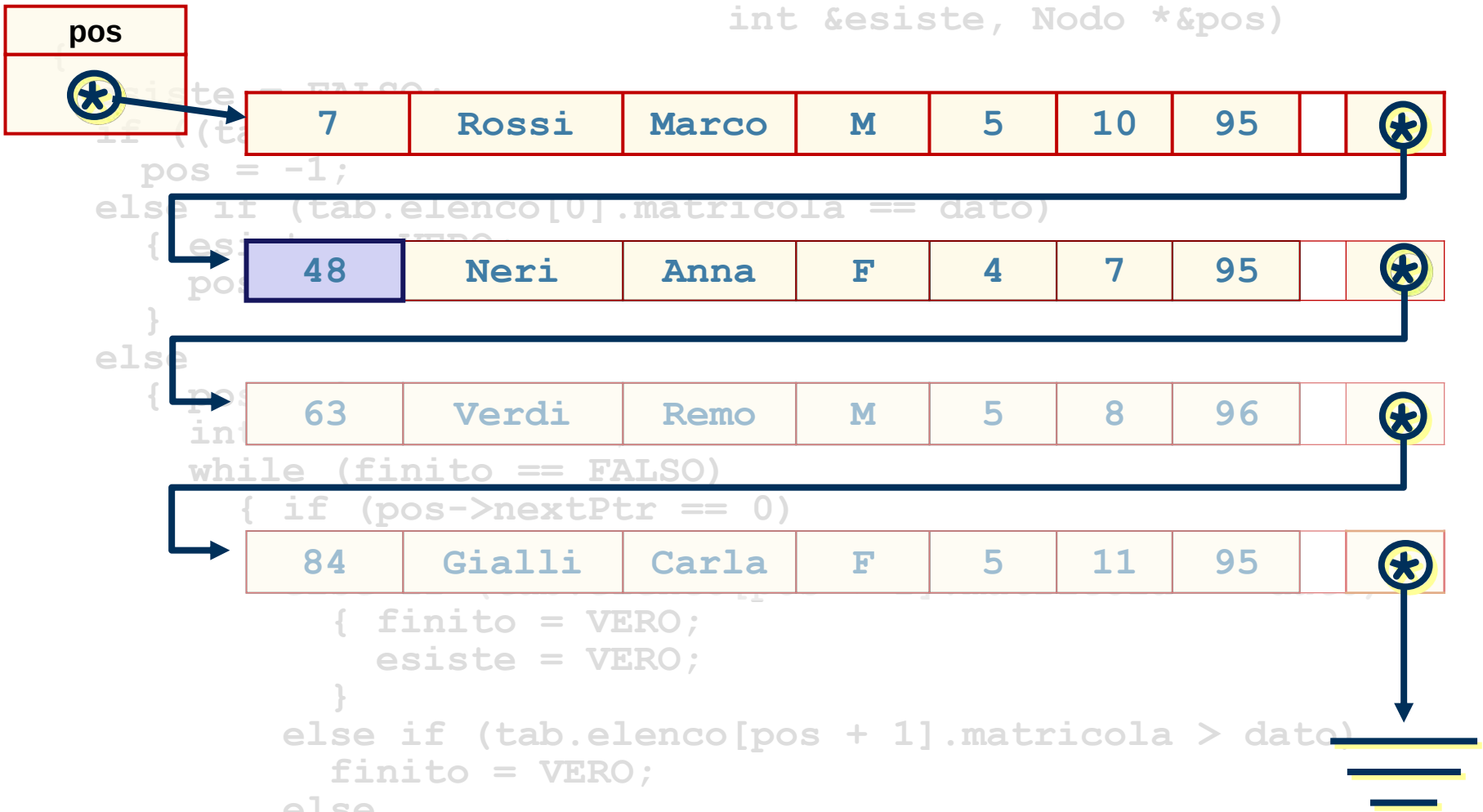

classe
7 Rossi Marco M 5 10 95
48 Neri Anna F 4 7 95
63 Verdi Remo M 5 8 96
84 Gialli Carla F 5 11 95


classe
7
Rossi
Marco
M
5
10
95
48
Neri
Anna
F
4
7
95
63
Verdi
Remo
M
5
8
96
84
Gialli
Carla
F
5
11
95
nextPtr

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.num == 0) || (tab.elenco[0].matricola > dato))
    pos = -1;
  else if (tab.elenco[0].matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos->nextPtr == 0)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

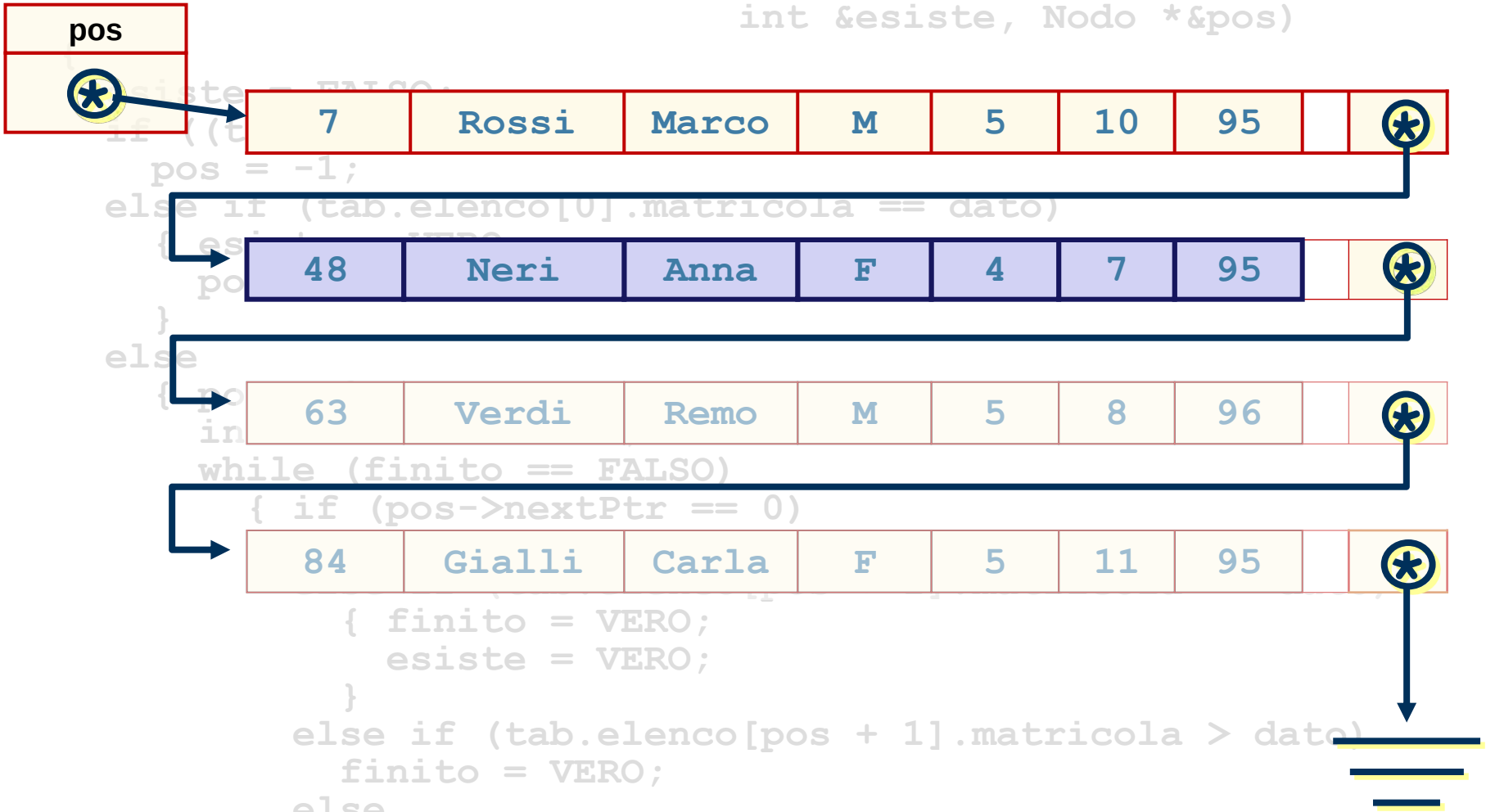
```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.num == 0) || (tab.elenco[0].matricola > dato))
    pos = -1;
  else if (tab.elenco[0].matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos->nextPtr == 0)
            finito = VERO;
          else if (tab.elenco[pos + 1].matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if (tab.elenco[pos + 1].matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```


pos
7
Rossi
Marco
M
5
10
95
48
Neri
Anna
F
4
7
95
63
Verdi
Remo
M
5
8
96
84
Gialli
Carla
F
5
11
95

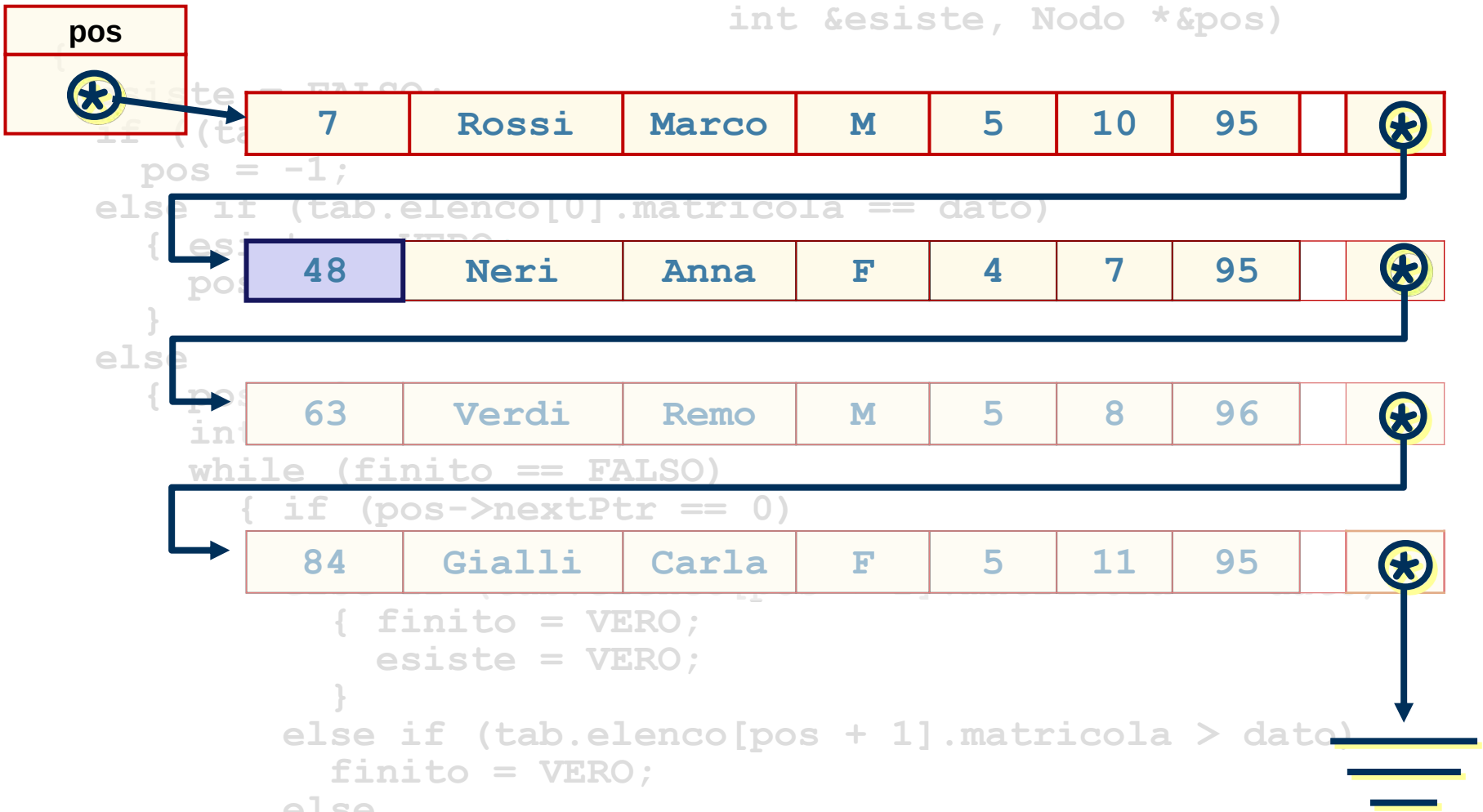

pos
7
Rossi
Marco
M
5
10
95
48
Neri
Anna
F
4
7
95
63
Verdi
Remo
M
5
8
96
84
Gialli
Carla
F
5
11
95


pos
7
Rossi
Marco
M
5
10
95
48
Neri
Anna
F
4
7
95
63
Verdi
Remo
M
5
8
96
84
Gialli
Carla
F
5
11
95

pos->nextPtr



pos->nextPtr ->datiStud



pos->nextPtr ->datiStud .matricola

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.num == 0) || (tab.elenco[0].matricola > dato))
    pos = -1;
  else if (tab.elenco[0].matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos->nextPtr == 0)
            finito = VERO;
          else if ((pos->nextPtr)->datiStud.matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if ((pos->next.Ptr)->datiStud.matricola > dato)
            finito = VERO;
          else
            pos++;
        }
    }
}
```

```
void cercaSeEsisteEPosizione(int dato, Nodo *&tab,
                             int &esiste, Nodo *&pos)
{
  esiste = FALSO;
  if ((tab.num == 0) || (tab.elenco[0].matricola > dato))
    pos = -1;
  else if (tab.elenco[0].matricola == dato)
    { esiste = VERO;
      pos = -1;
    }
  else
    { pos = 0;
      int finito = FALSO;
      while (finito == FALSO)
        { if (pos->nextPtr == 0)
            finito = VERO;
          else if ((pos->nextPtr)->datiStud.matricola == dato)
            { finito = VERO;
              esiste = VERO;
            }
          else if ((pos->next.Ptr)->datiStud.matricola > dato)
            finito = VERO;
          else
            pos = pos->nextPtr;
        }
    }
}
```

```
void inserisciSeNonEsiste(Studente nuovoStudente, Nodo *&tab)
{
  int esiste, posizione;

  cercaSeEsisteEPosizione(nuovoStudente.matricola,
                          tab, esiste, posizione);
  if ((esiste == FALSO) && (tab.num <= DIM))
    { cout << "inserimento di studente con matricola"
           << setw(5) << nuovoStudente.matricola << endl;
      int temp = posizione + 1;
      for (int i = tab.num -1; i >= temp; i--)
        tab.elenco[i + 1] = tab.elenco[i];
      tab.elenco[temp] = nuovoStudente;
      tab.num++;
    }
}
```

```
void inserisciSeNonEsiste(Studente nuovoStudente, Nodo *&tab)
{
  int esiste, posizione;

  cercaSeEsisteEPosizione(nuovoStudente.matricola,
                          tab, esiste, posizione);
  if ((esiste == FALSO) && (tab.num <= DIM))
    { cout << "inserimento di studente con matricola"
           << setw(5) << nuovoStudente.matricola << endl;
      int temp = posizione + 1;
      for (int i = tab.num -1; i >= temp; i--)
        tab.elenco[i + 1] = tab.elenco[i];
      tab.elenco[temp] = nuovoStudente;
      tab.num++;
    }
}
```

```
void inserisciSeNonEsiste(Studente nuovoStudente, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(nuovoStudente.matricola,
                          tab, esiste, posizione);
  if ((esiste == FALSO) && (tab.num <= DIM))
    { cout << "inserimento di studente con matricola"
           << setw(5) << nuovoStudente.matricola << endl;
      int temp = posizione + 1;
      for (int i = tab.num -1; i >= temp; i--)
        tab.elenco[i + 1] = tab.elenco[i];
      tab.elenco[temp] = nuovoStudente;
      tab.num++;
    }
}
```

```
void inseris                                    tab)
{
  int esiste;
  Nodo *posiz
  cercaSeEsis

  if ((esiste
    { cout <<
            <<
      int tem
      for (in
        tab.e
      tab.elenco[temp] = nuovoStudente;
      tab.num++;
    }
}
```

classe

num		elenco						
4		matricola	cognome	nome	sesso	dataNascita		
						giorno	mese	anno
	0	7	Rossi	Marco	M	5	10	95
	1	48	Neri	Anna	F	4	7	95
	2	63	Verdi	Remo	M	5	8	96
	3	84	Gialli	Carla	F	5	11	95
	4							

```
void inserisciSeNonEsiste(Studente nuovoStudente, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(nuovoStudente.matricola,
                          tab, esiste, posizione);
  if ((esiste == FALSO) && (tab.num <= DIM))
    { cout << "inserimento di studente con matricola"
           << setw(5) << nuovoStudente.matricola << endl;
      int temp = posizione + 1;
      for (int i = tab.num -1; i >= temp; i--)
        tab.elenco[i + 1] = tab.elenco[i];
      tab.elenco[temp] = nuovoStudente;
      tab.num++;
    }
}
```

```
void inserisciSeNonEsiste(Studente nuovoStudente, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(nuovoStudente.matricola,
                          tab, esiste, posizione);
  if (esiste == FALSO)
    { cout << "inserimento di studente con matricola"
           << setw(5) << nuovoStudente.matricola << endl;
      int temp = posizione + 1;
      for (int i = tab.num -1; i >= temp; i--)
        tab.elenco[i + 1] = tab.elenco[i];
      tab.elenco[temp] = nuovoStudente;
      tab.num++;
    }
}
```

```
void inserisciSeNonEsiste(Studente nuovoStudente, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(nuovoStudente.matricola,
                          tab, esiste, posizione);
  if (esiste == FALSO)
    { cout << "inserimento di studente con matricola"
           << setw(5) << nuovoStudente.matricola << endl;
      Nodo *temp;
      for (int i = tab.num -1; i >= temp; i--)
        tab.elenco[i + 1] = tab.elenco[i];
      tab.elenco[temp] = nuovoStudente;
      tab.num++;
    }
}
```

classe

num		elenco						
4		matricola	cognome	nome	sesso	dataNascita		
						giorno	mese	anno
	0	7	Rossi	Marco	M	5	10	95
	1	48	Neri	Anna	F	4	7	95
	2	63	Verdi	Remo	M	5	8	96
	3	84	Gialli	Carla	F	5	11	95
	4							
		12	Violi	Aldo	M	10	5	96

```
void inseris                                    tab)
{
  int esiste;
  Nodo *posiz
  cercaSeEsis

  if ((esiste
    { cout <<
            <<
      Nodo *t
      for (in
        tab.e
      tab.elenco[temp] = nuovoStudente;
      tab.num++;
    }
}
```

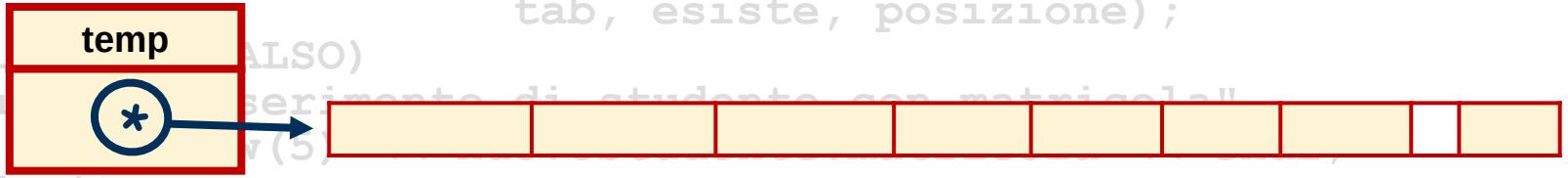
classe

num		elenco						
4		matricola	cognome	nome	sesso	dataNascita		
						giorno	mese	anno
	0	7	Rossi	Marco	M	5	10	95
	1	12	Violi	Aldo	M	10	5	96
	2	48	Neri	Anna	F	4	7	95
	3	63	Verdi	Remo	M	5	8	96
	4	84	Gialli	Carla	F	5	11	95

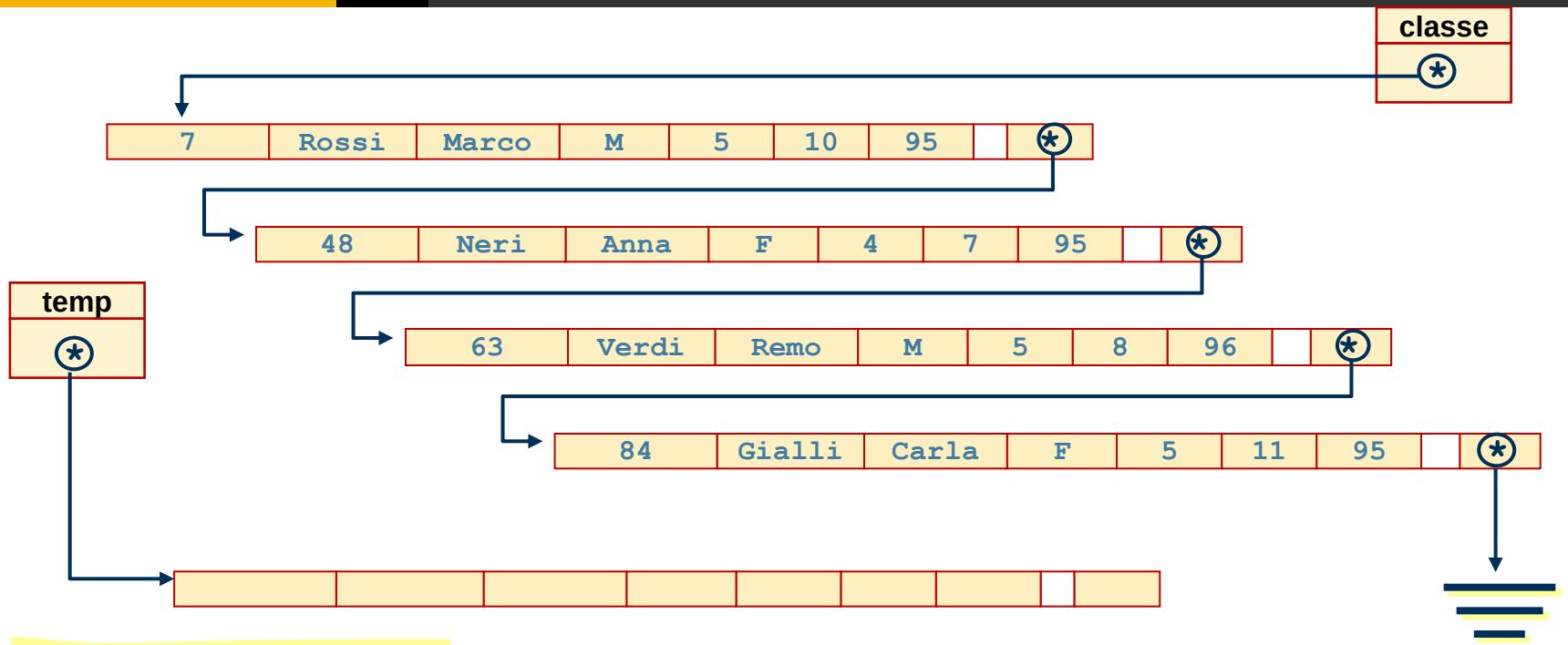
```
void inserisciSeNonEsiste(Studente nuovoStudente, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(nuovoStudente.matricola,
                          tab, esiste, posizione);
  if (esiste == FALSO)
    { cout << "inserimento di studente con matricola"
           << setw(5) << nuovoStudente.matricola << endl;
      Nodo *temp;
      for (int i = tab.num -1; i >= temp; i--)
        tab.elenco[i + 1] = tab.elenco[i];
      tab.elenco[temp] = nuovoStudente;
      tab.num++;
    }
}
```

```
void inserisciSeNonEsiste(Studente nuovoStudente, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(nuovoStudente.matricola,
                          tab, esiste, posizione);
  if (esiste == FALSO)
    { cout << "inserimento di studente con matricola"
           << setw(5) << nuovoStudente.matricola << endl;
      Nodo *temp;
      temp = new Nodo;
      for (int i = tab.num -1; i >= temp; i--)
        tab.elenco[i + 1] = tab.elenco[i];
      tab.elenco[temp] = nuovoStudente;
      tab.num++;
    }
}
```

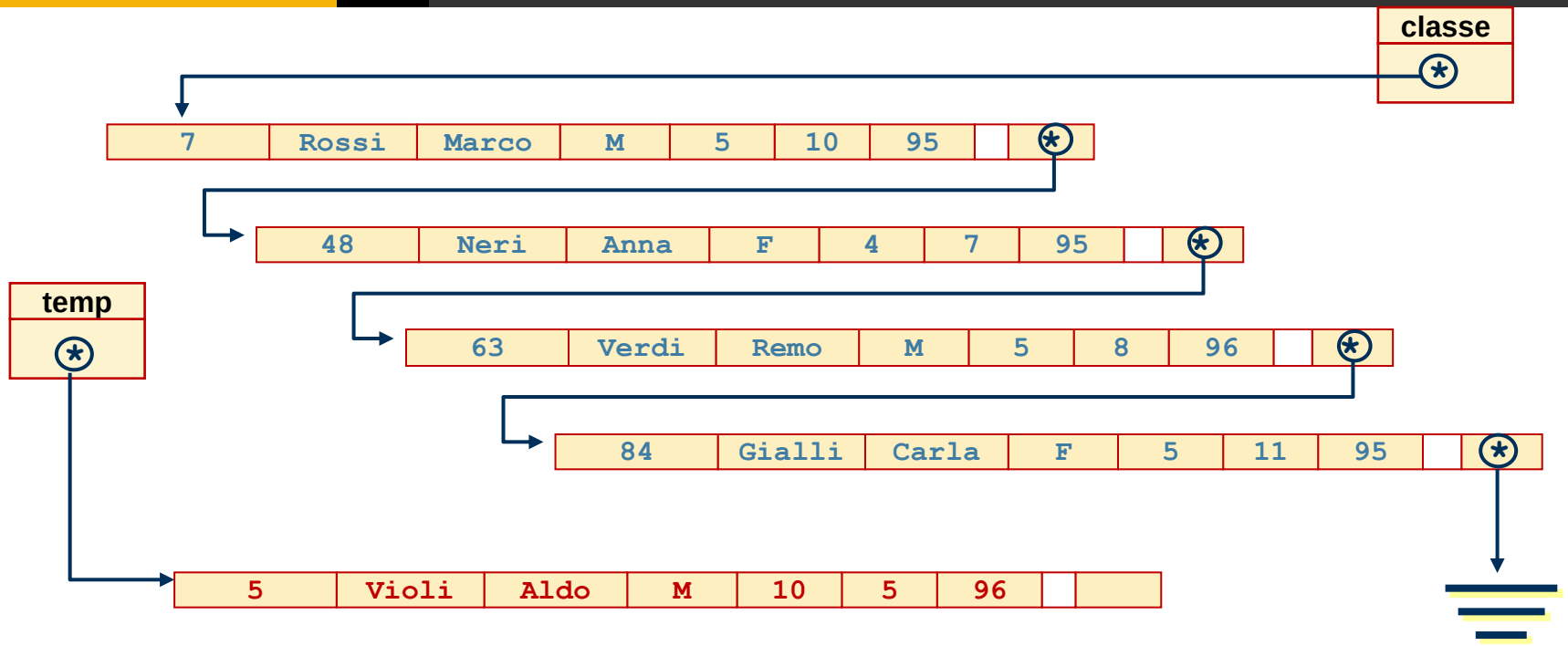
```
void inserisciSeNonEsiste(Studente nuovoStudente, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(nuovoStudente.matricola,
                          tab, esiste, posizione);
  if (esi                 ALSO)
    { cou                 serimento di studente con matricola"
                          v(5,
      Nodo *temp;
      temp = new Nodo;
      for (int i = tab.num -1; i >= temp; i--)
        tab.elenco[i + 1] = tab.elenco[i];
      tab.elenco[temp] = nuovoStudente;
      tab.num++;
    }
}
```



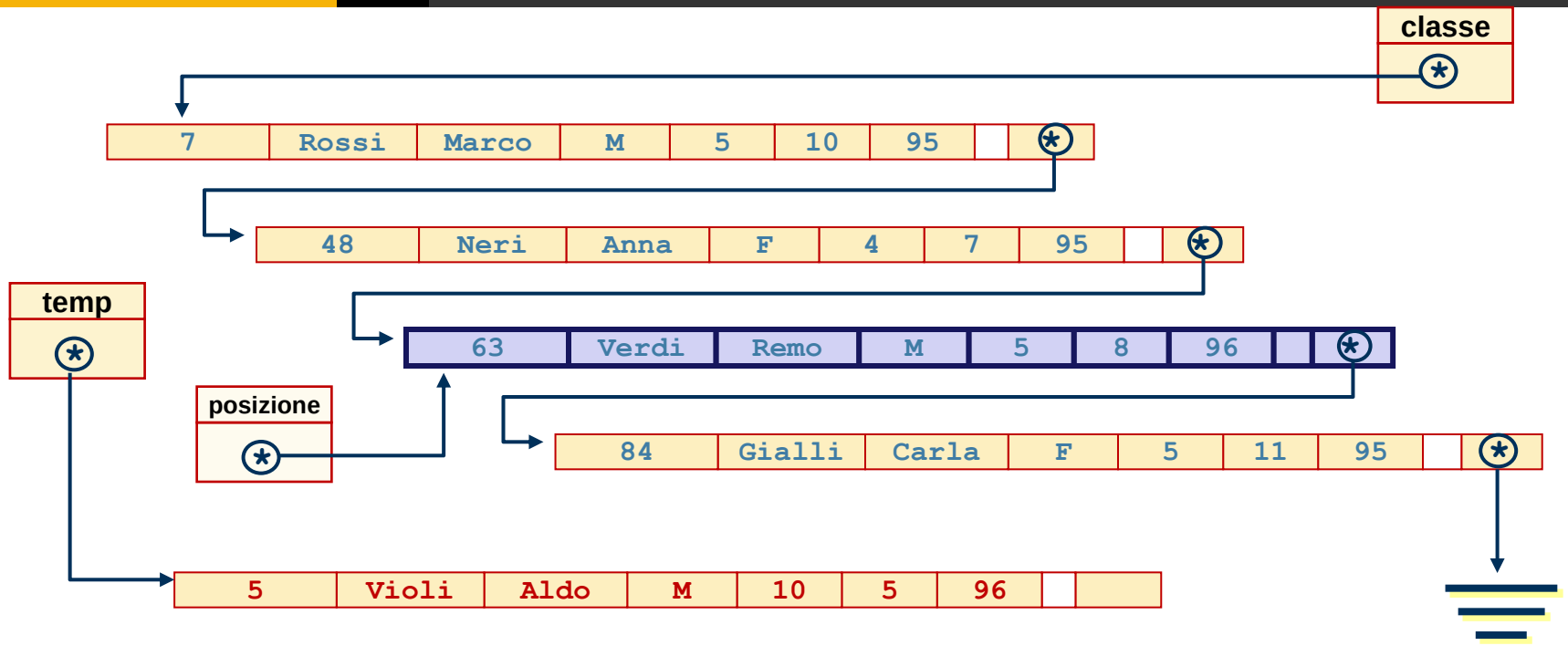
```
void inserisciSeNonEsiste(Studente nuovoStudente, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(nuovoStudente.matricola,
                          tab, esiste, posizione);
  if (esiste == FALSO)
    { cout << "inserimento di studente con matricola"
           << setw(5) << nuovoStudente.matricola << endl;
      Nodo *temp;
      temp = new Nodo;
      for (int i = tab.num -1; i >= temp; i--)
        tab.elenco[i + 1] = tab.elenco[i];
      tab.elenco[temp] = nuovoStudente;
      tab.num++;
    }
}
```



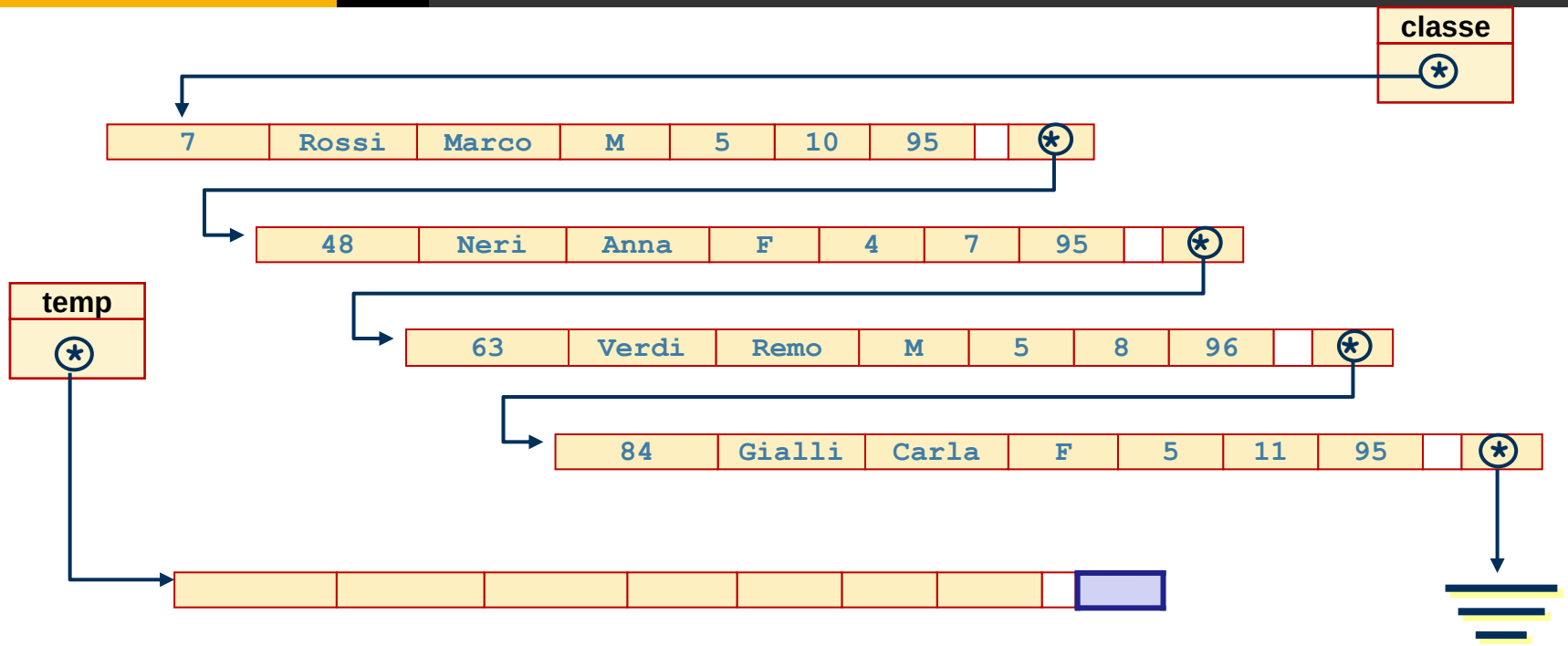

```
temp = new Nodo;
```




```
temp = new Nodo;
if (posizione == 0)
```



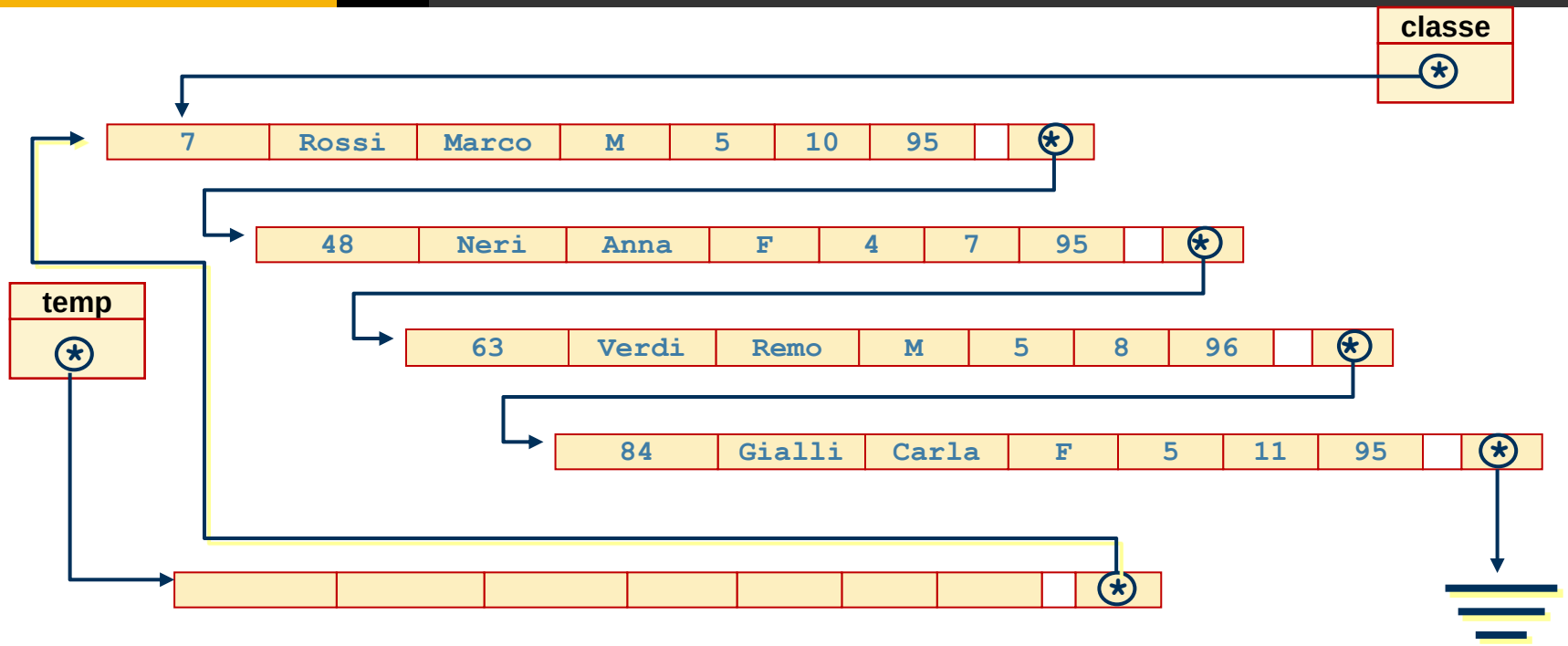

```
temp = new Nodo;
```




```
temp = new Nodo;
```

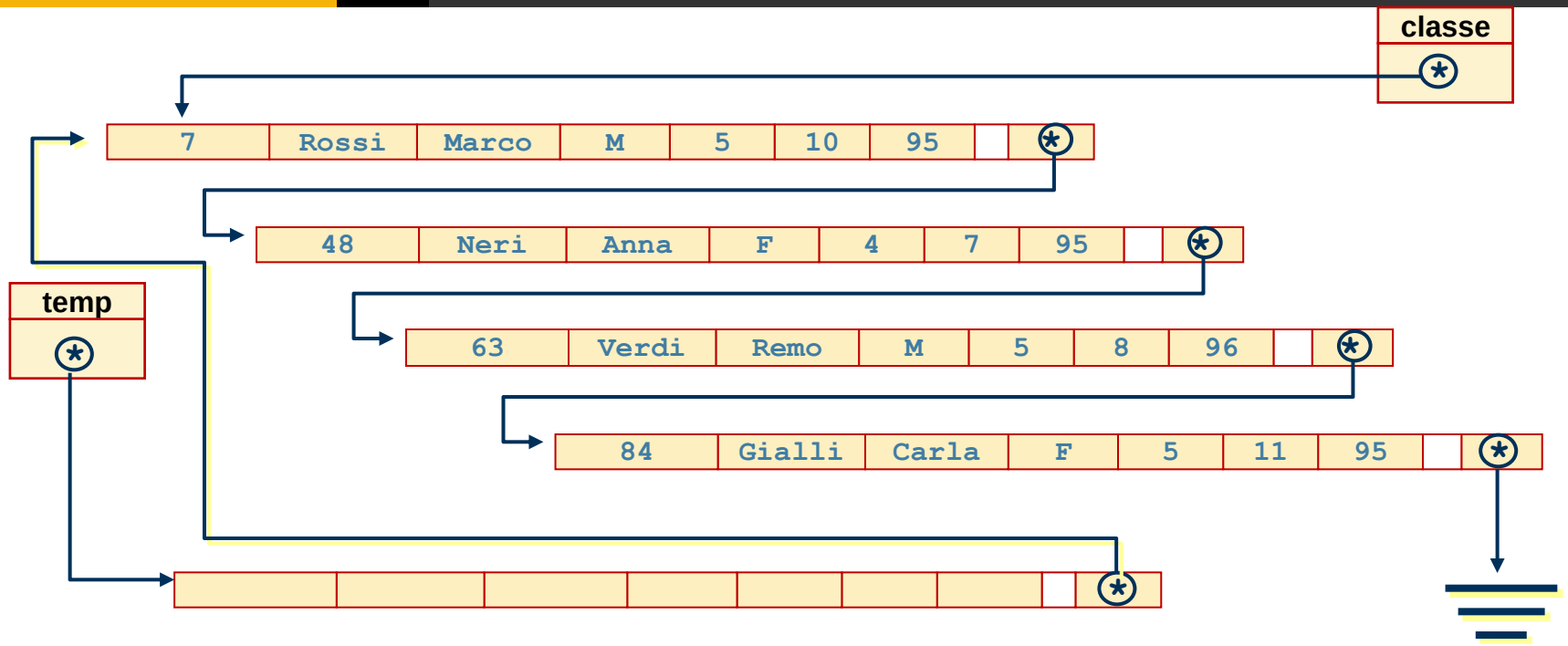
nextPtr

```
if (posizione == 0)
```



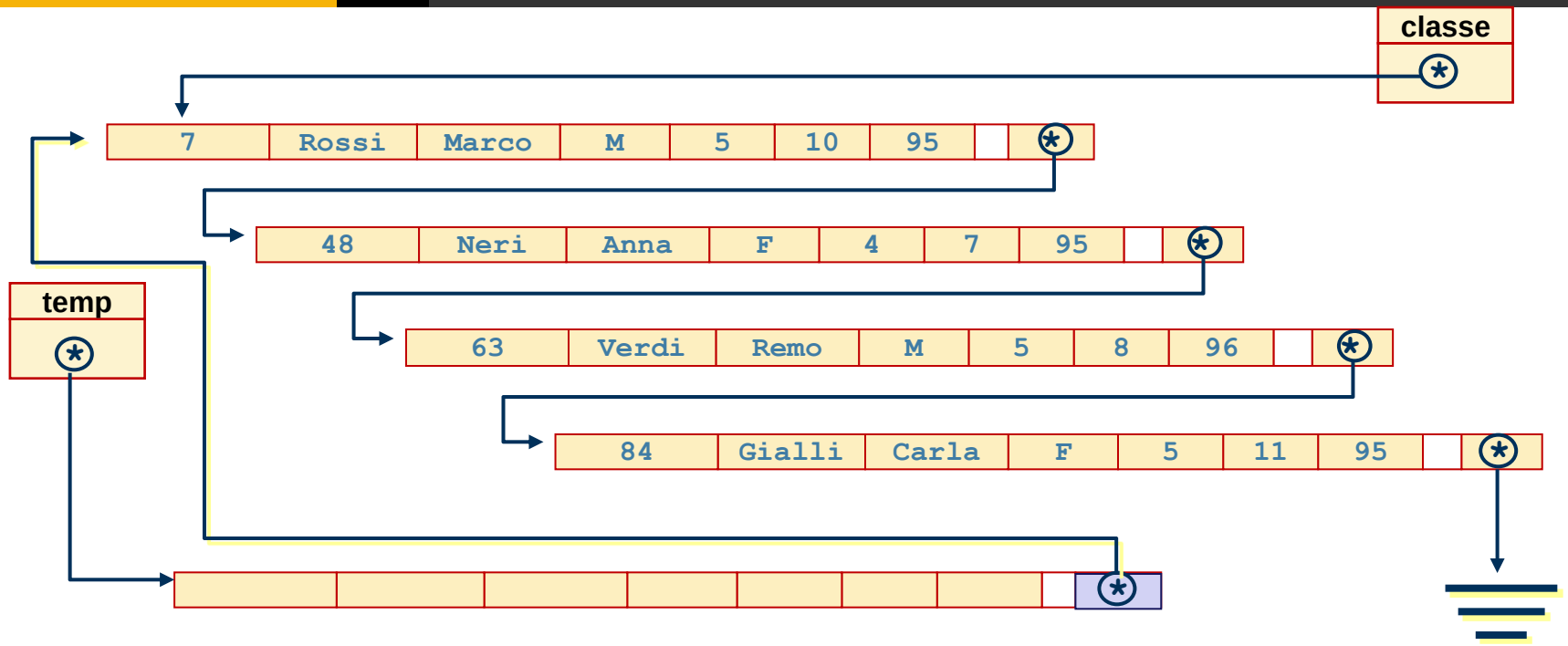

```
temp = new Nodo;
if (posizione == 0)
```

nextPtr

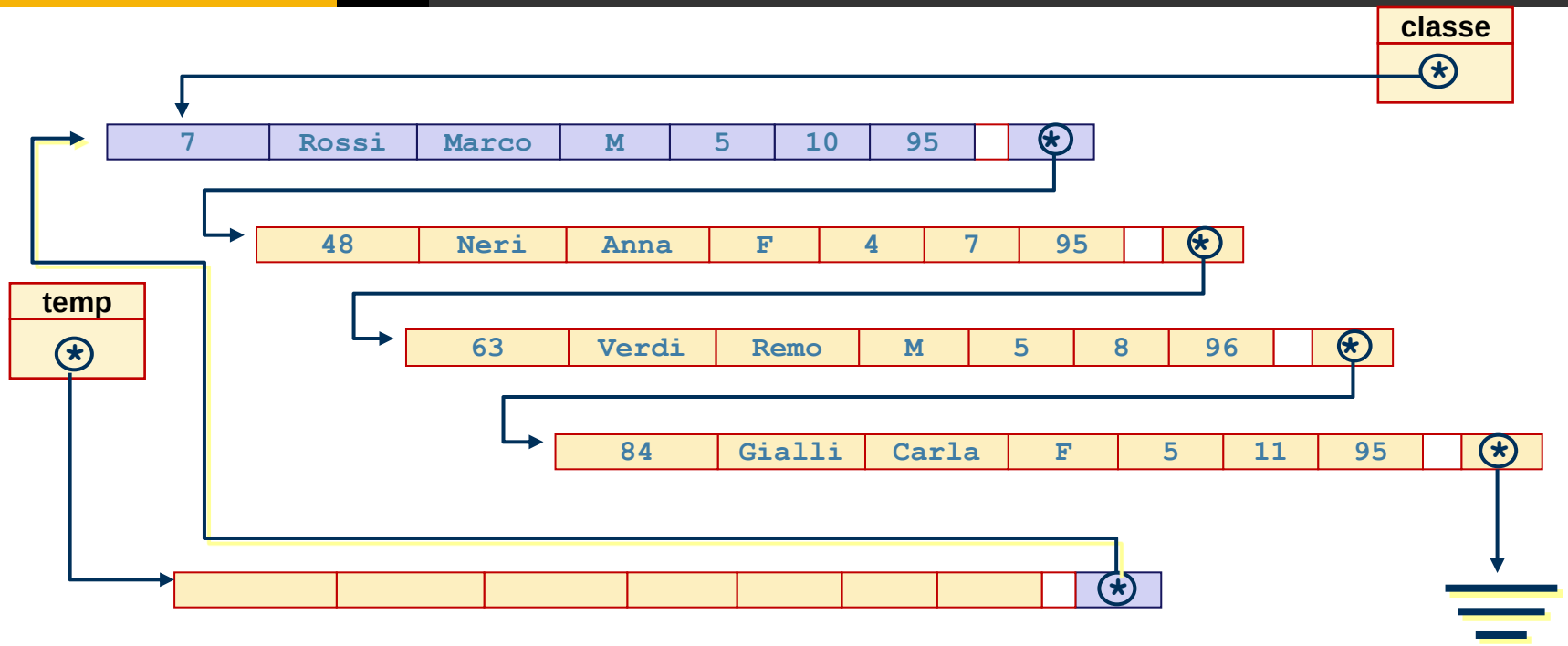



```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
```

nextPtr

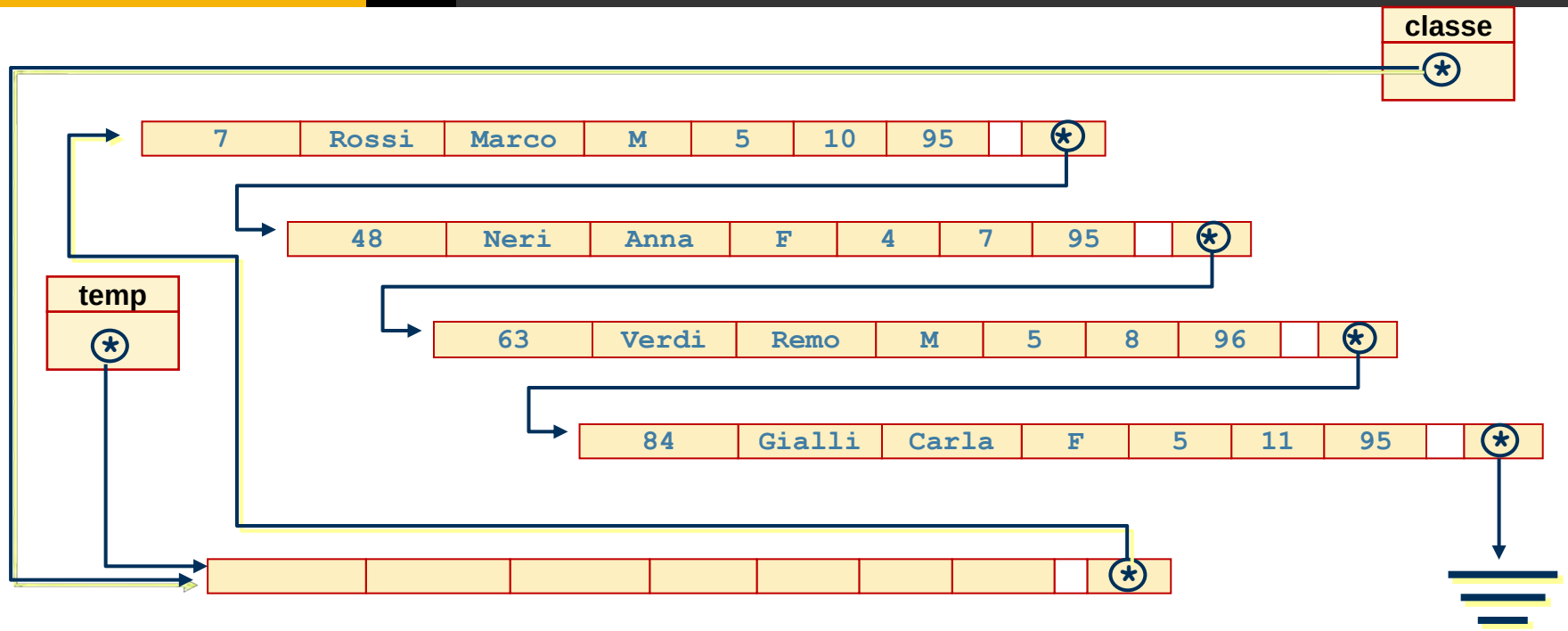



```
temp = new Nodo;                                    nextPtr
if (posizione == 0)
   { temp->nextPtr = tab;
```

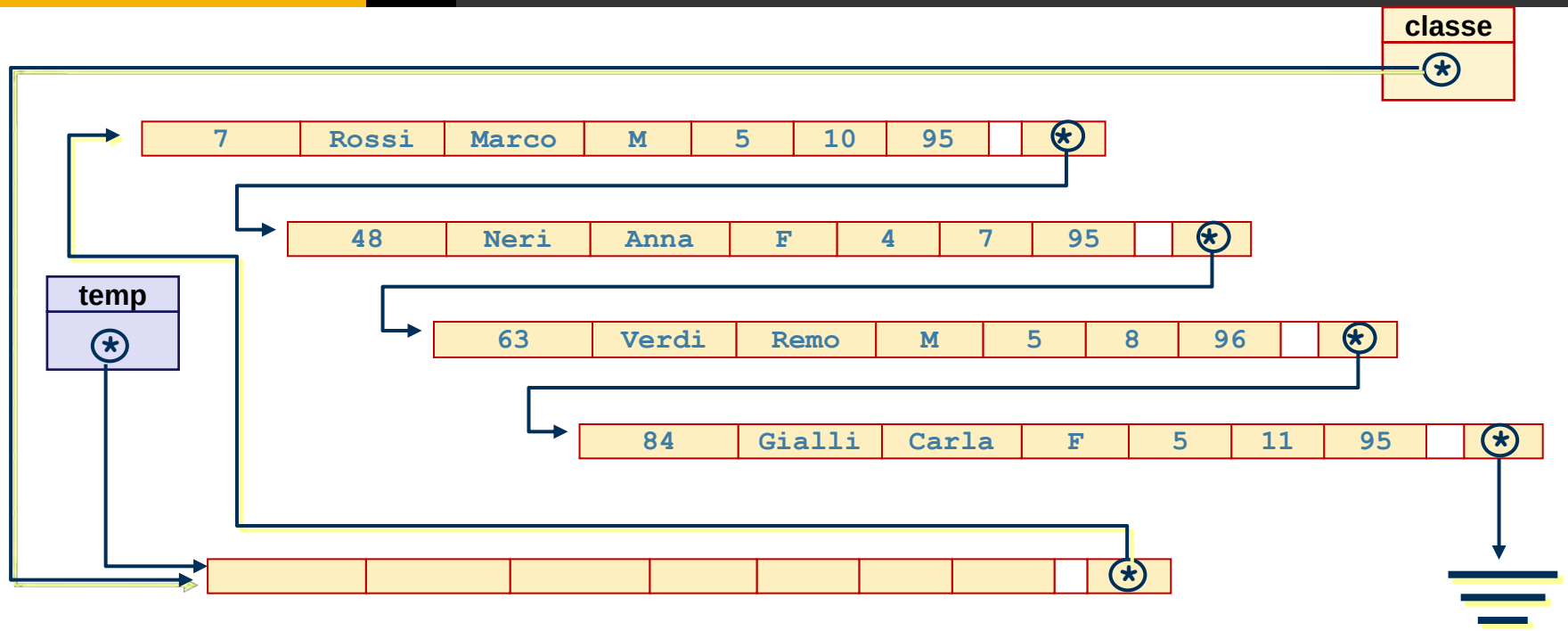



```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
```

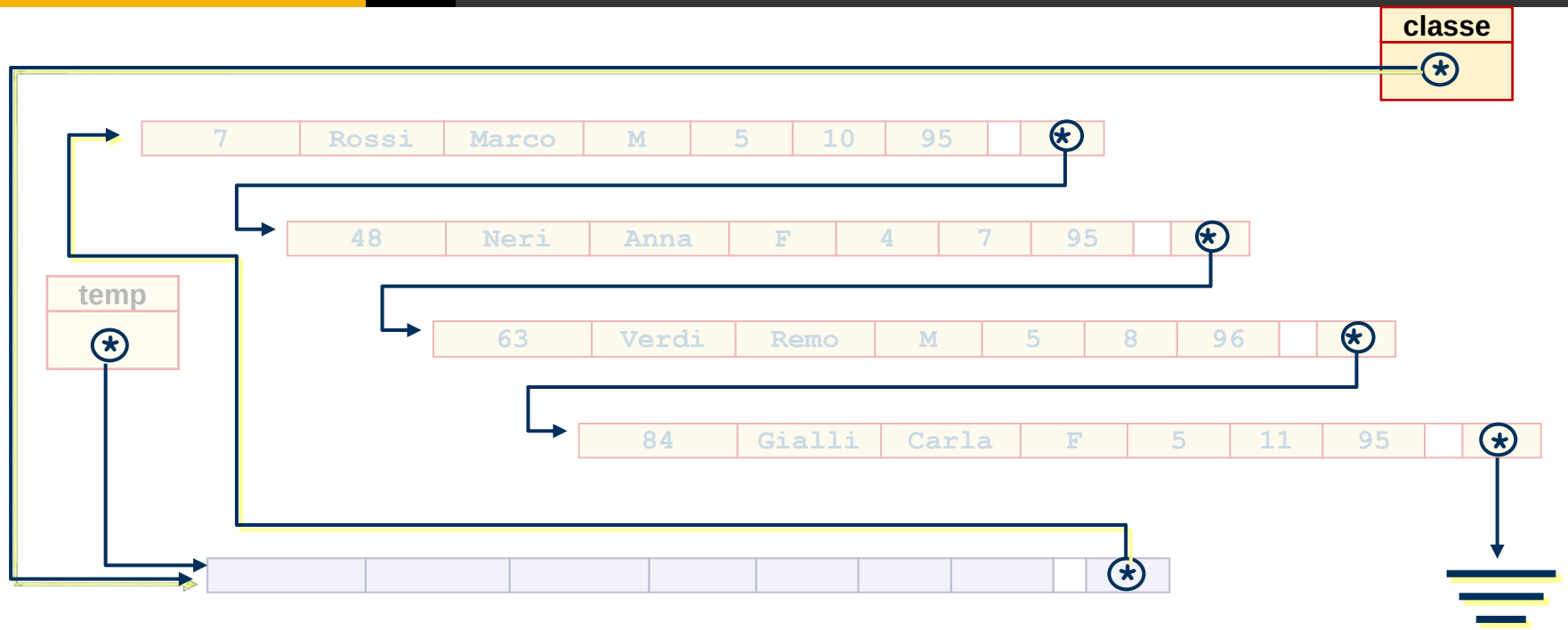
nextPtr



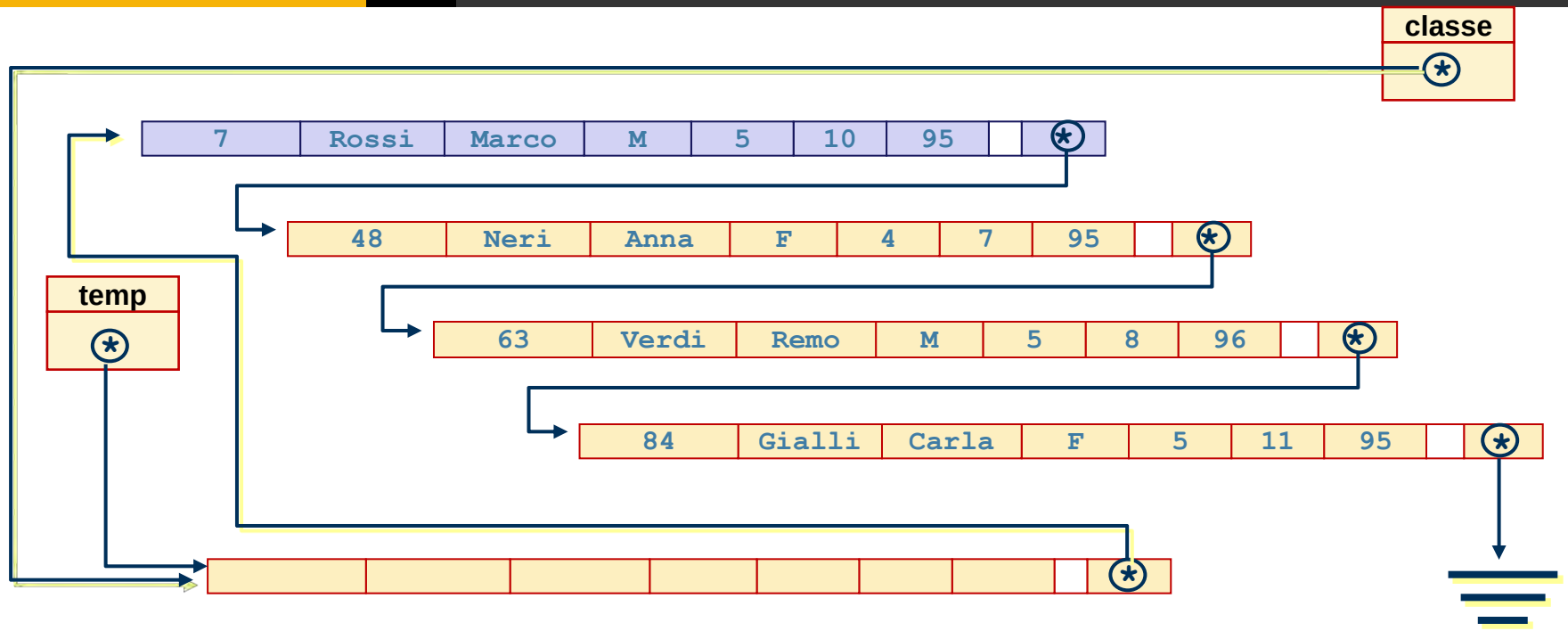

```
temp = new Nodo;                                    nextPtr
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
```



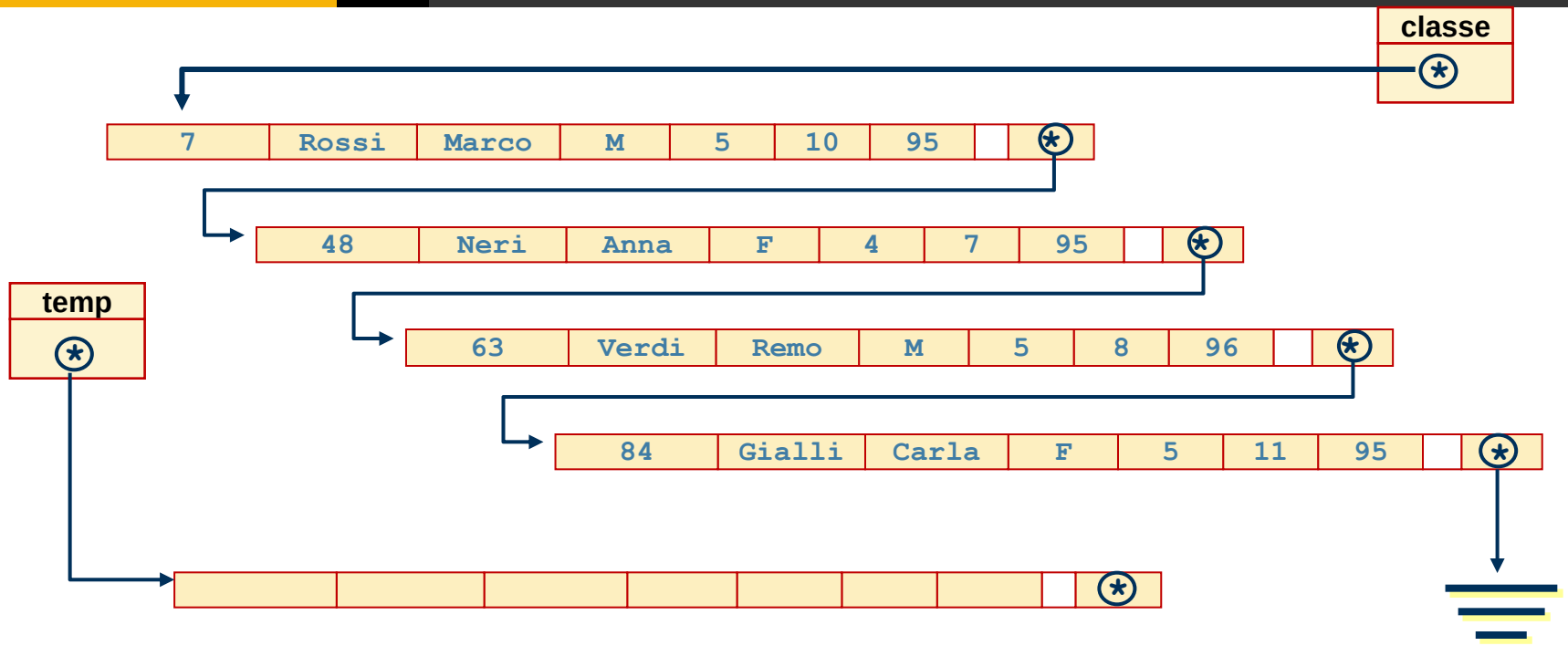

```
temp = new Nodo;                                    nextPtr
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
```



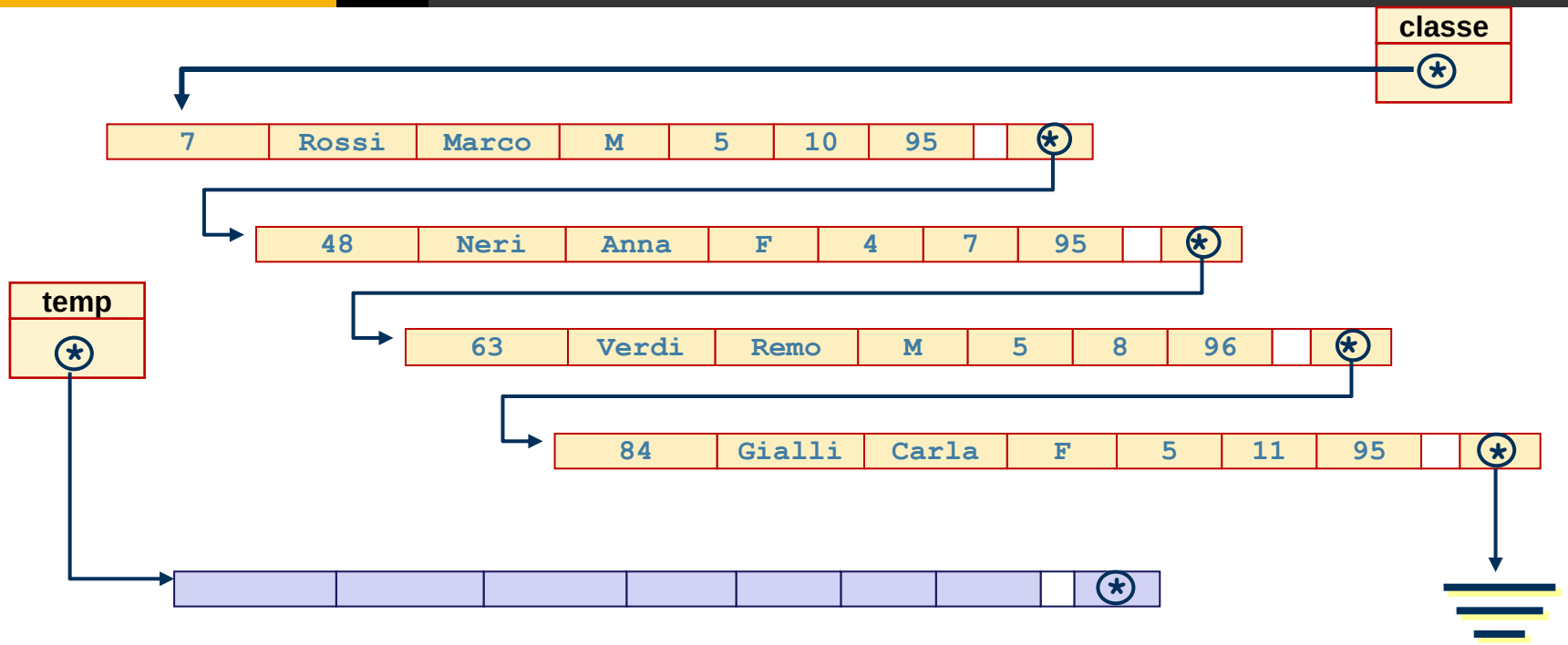

```
temp = new Nodo;                                    nextPtr
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
```



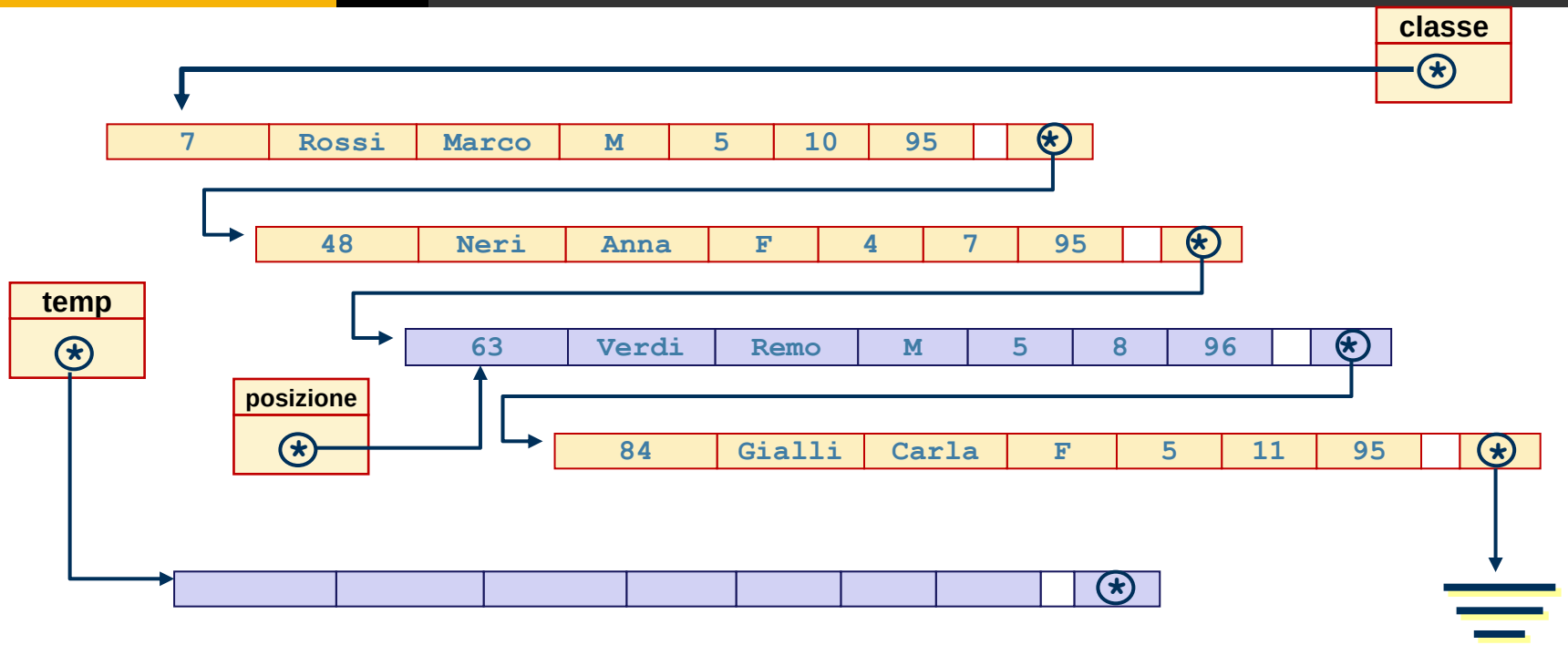

```
temp = new Nodo;                                    nextPtr
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
```



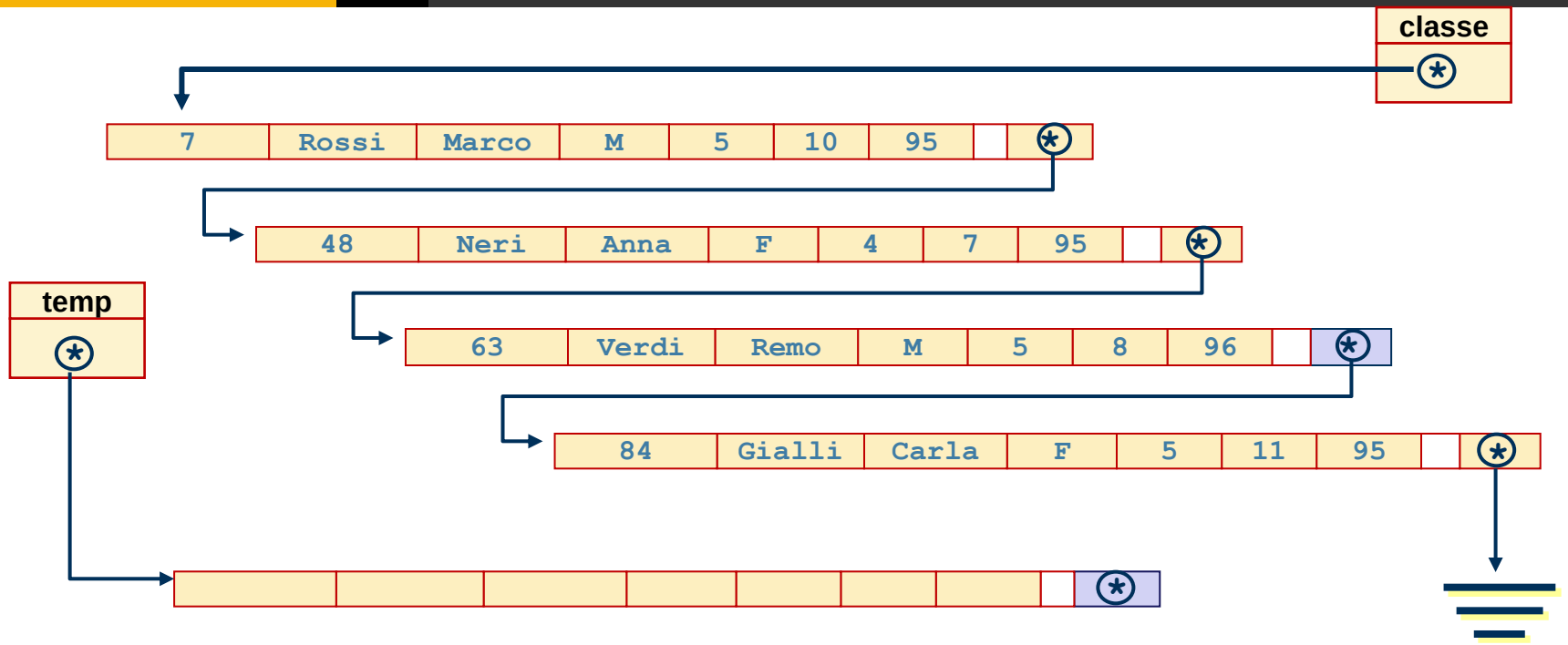

```
temp = new Nodo;                                    nextPtr
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
   }
else
```



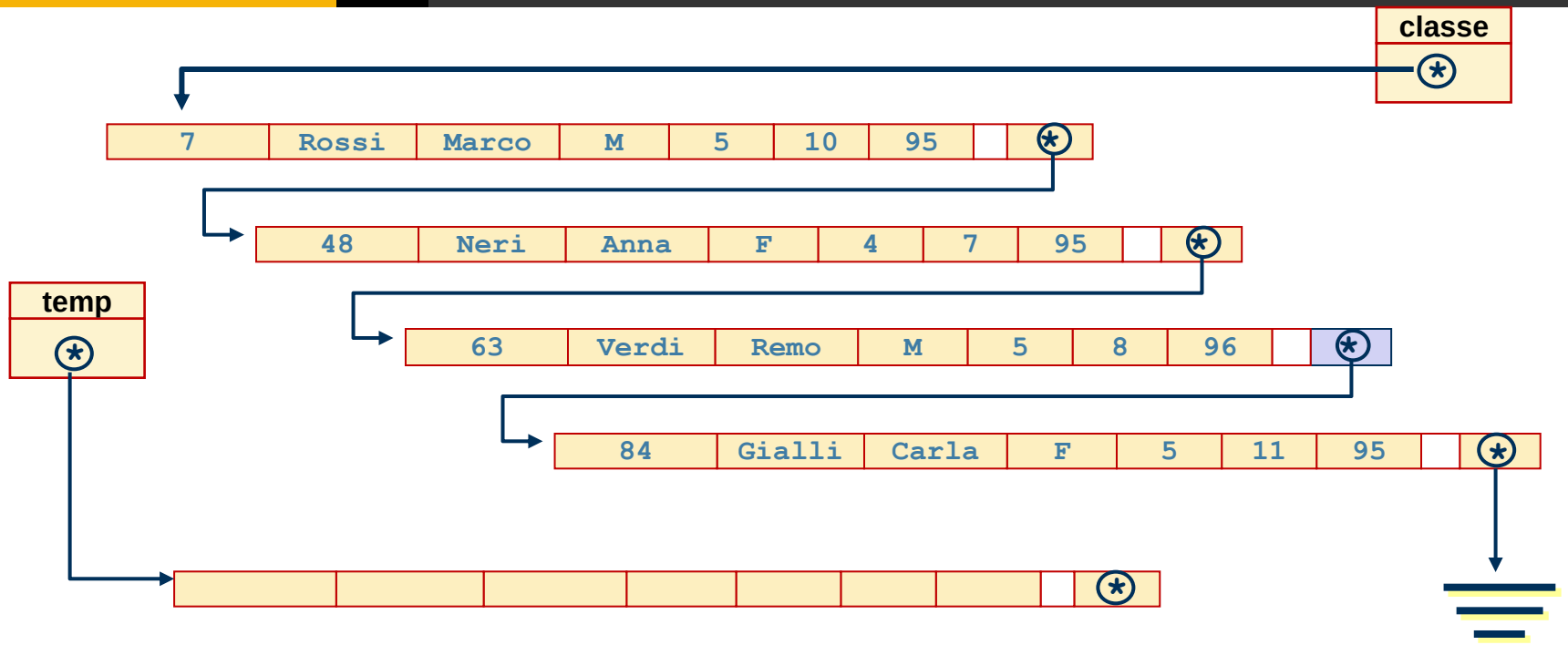

```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
   }
else
```



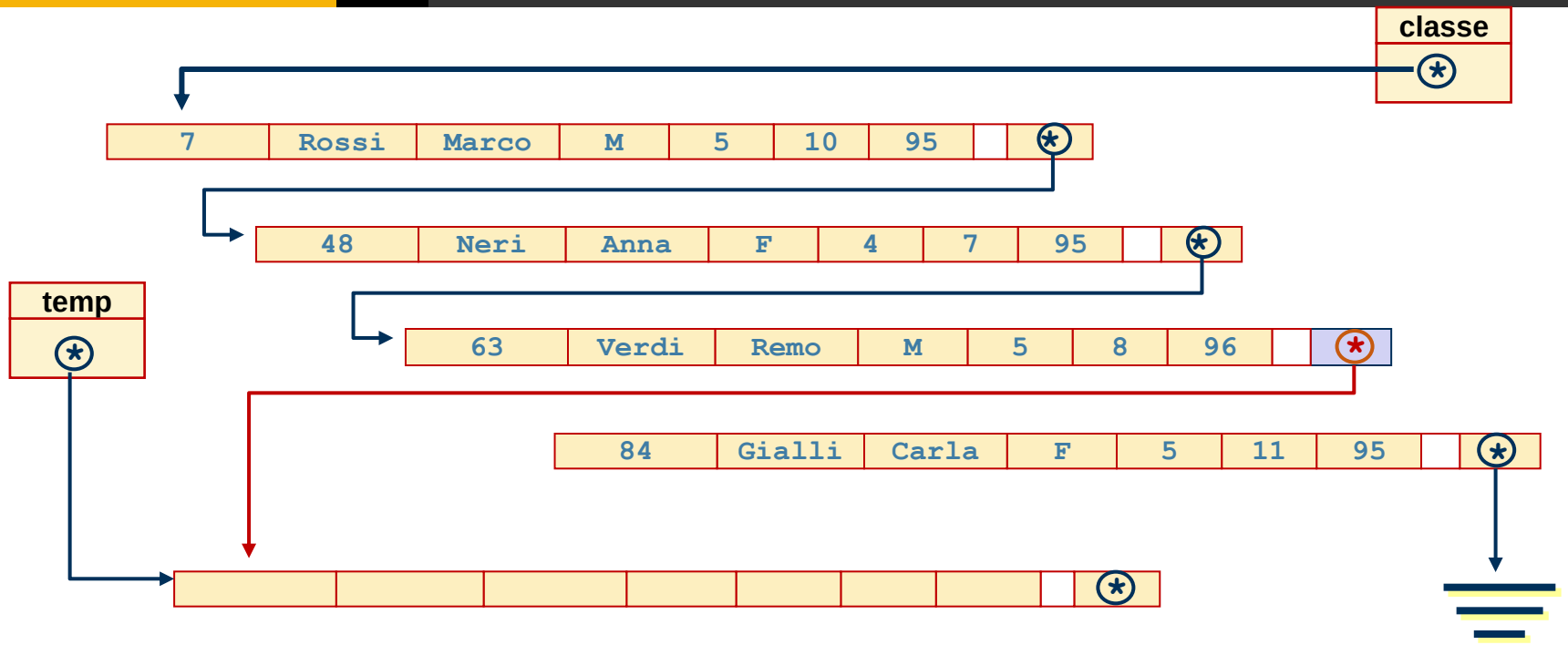

```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
   }
else
```



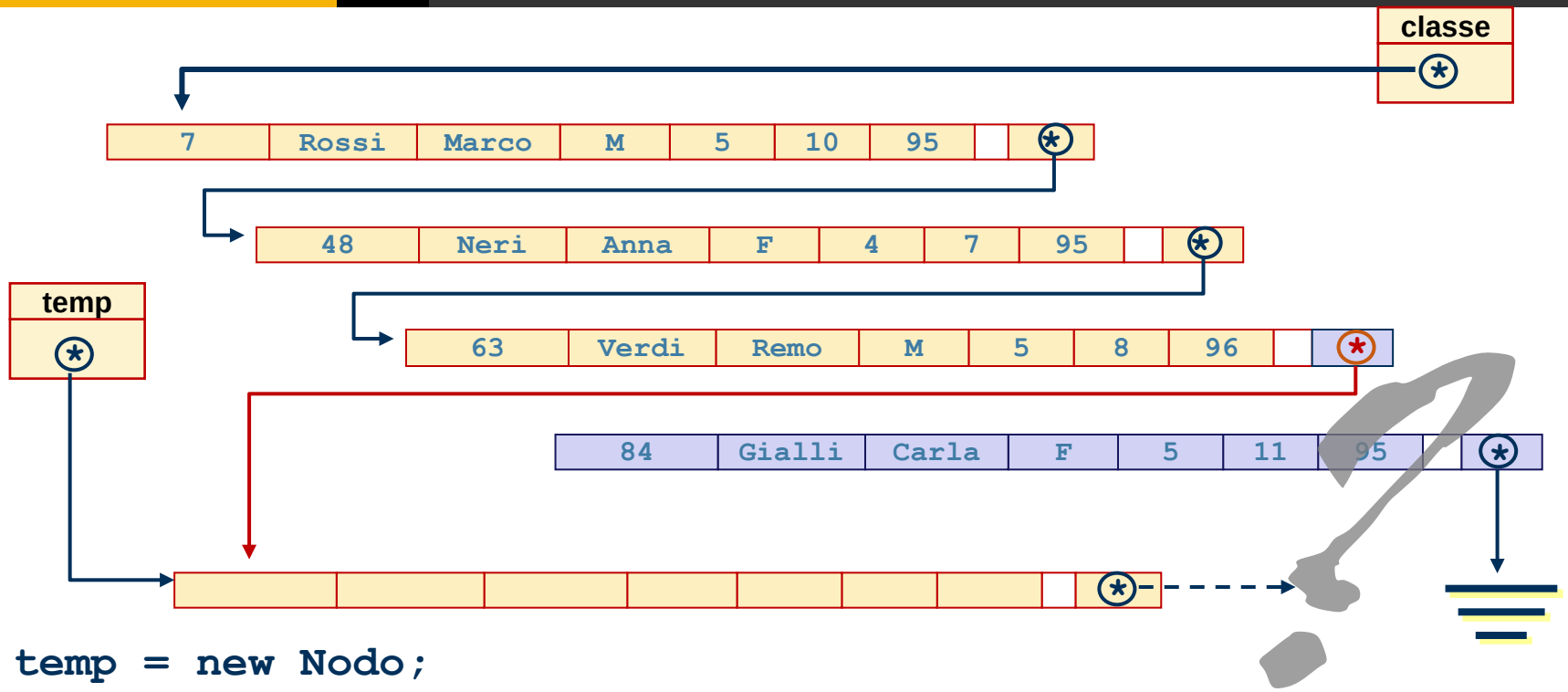

```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
   }
else
```



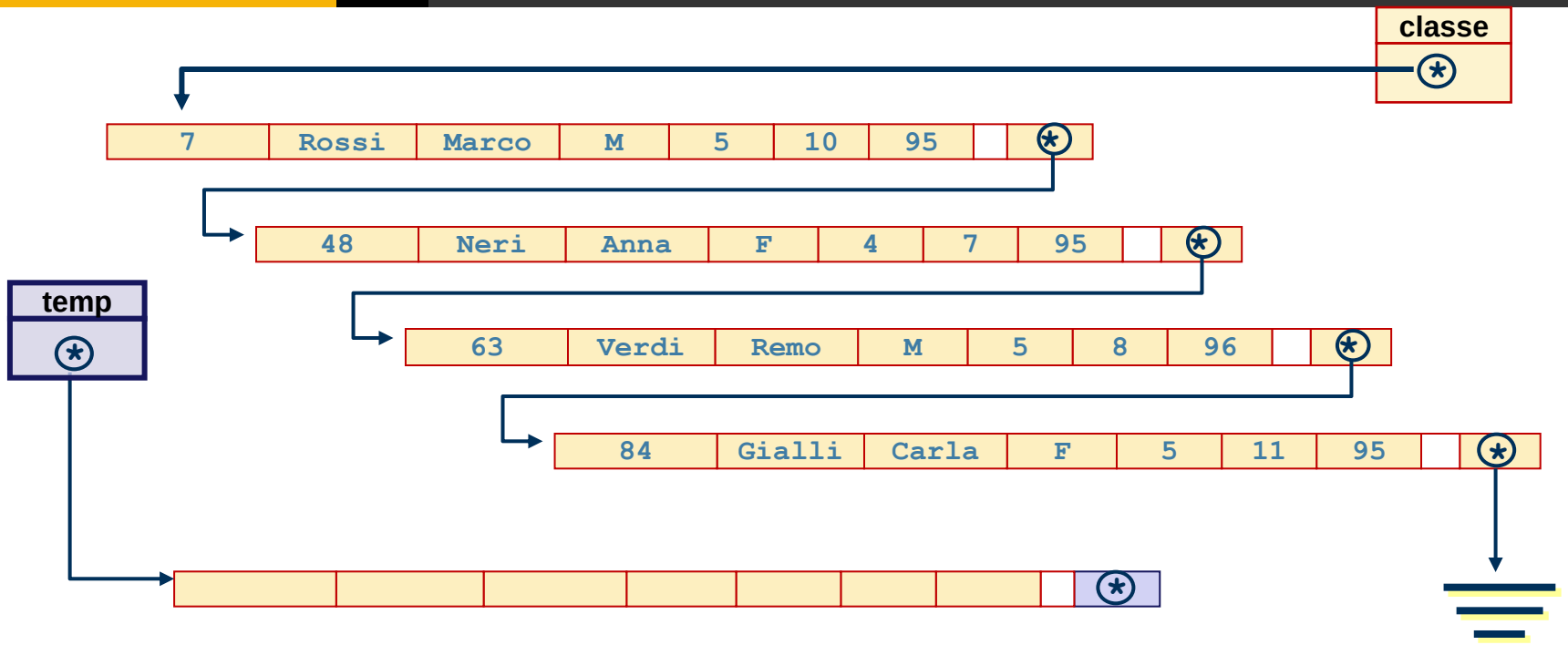

```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
   }
else
```



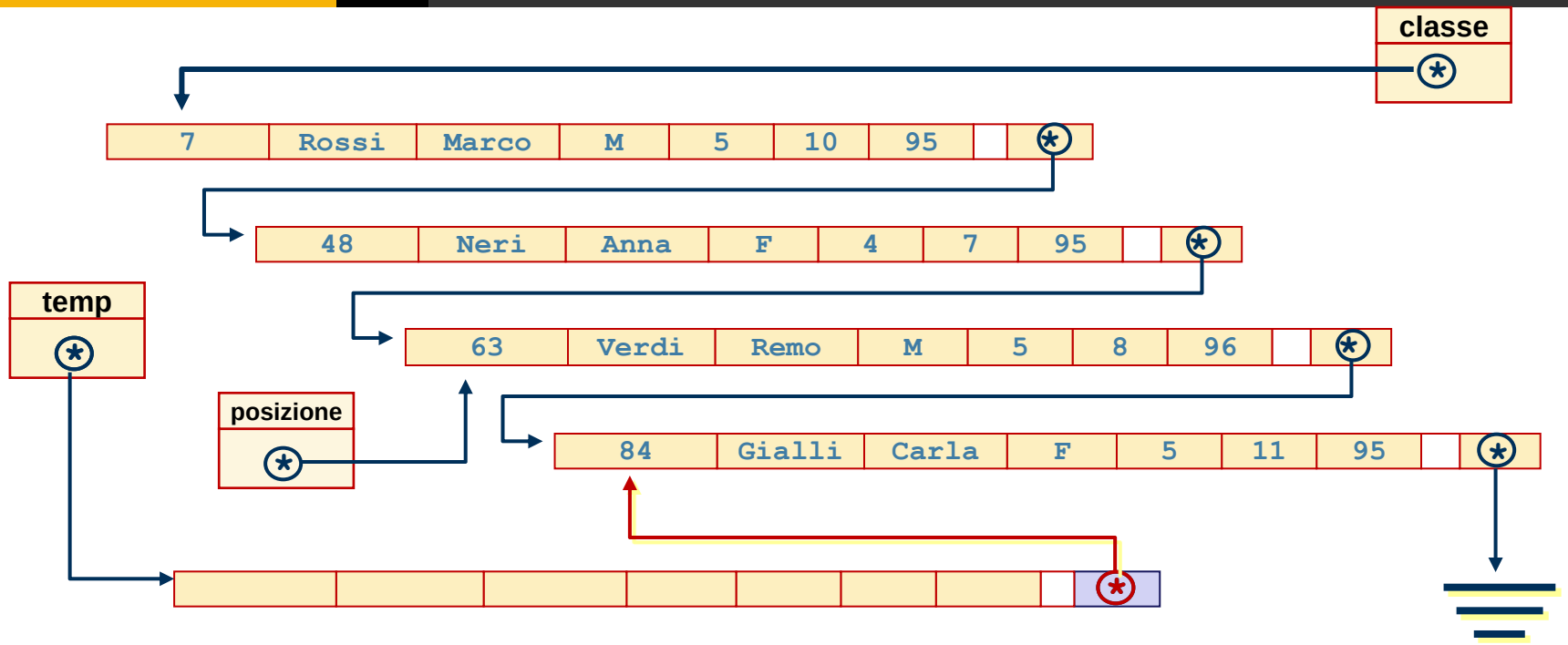

```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
   }
else
```



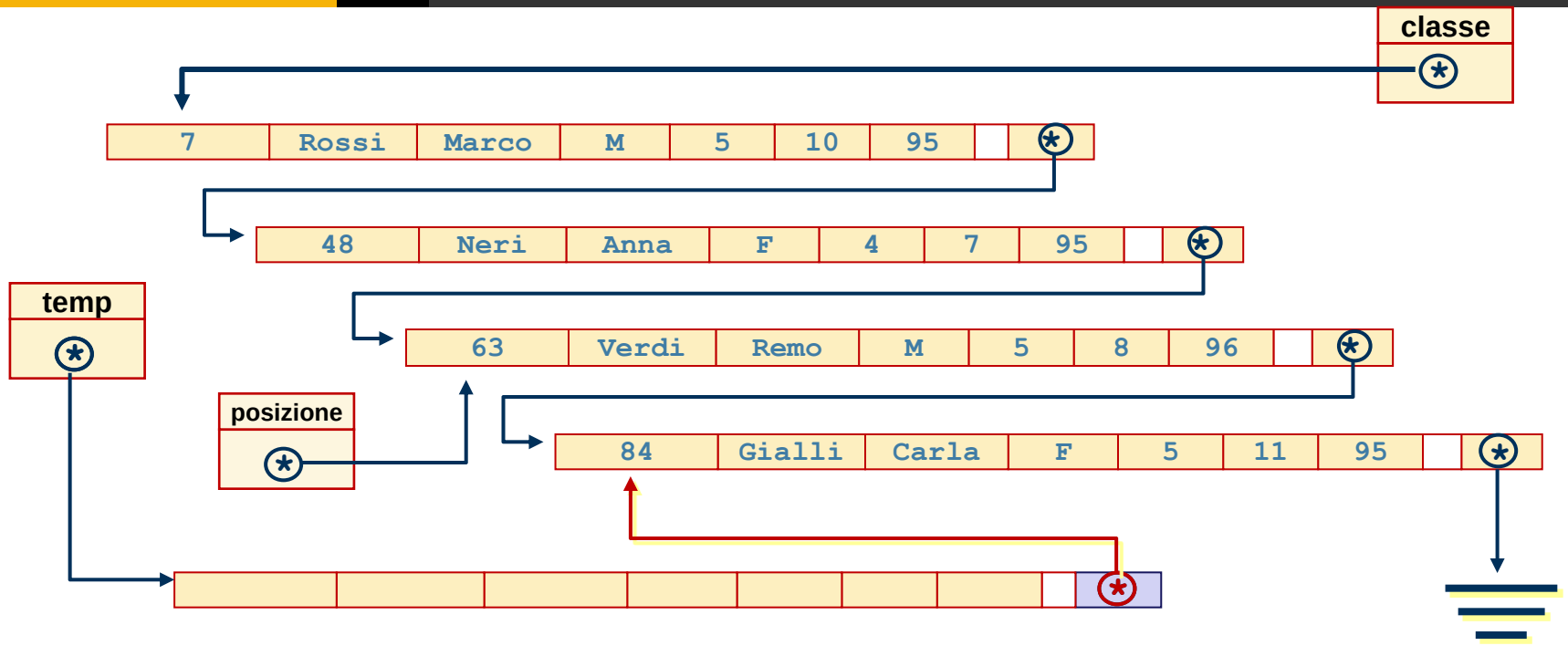

```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
   }
else
```



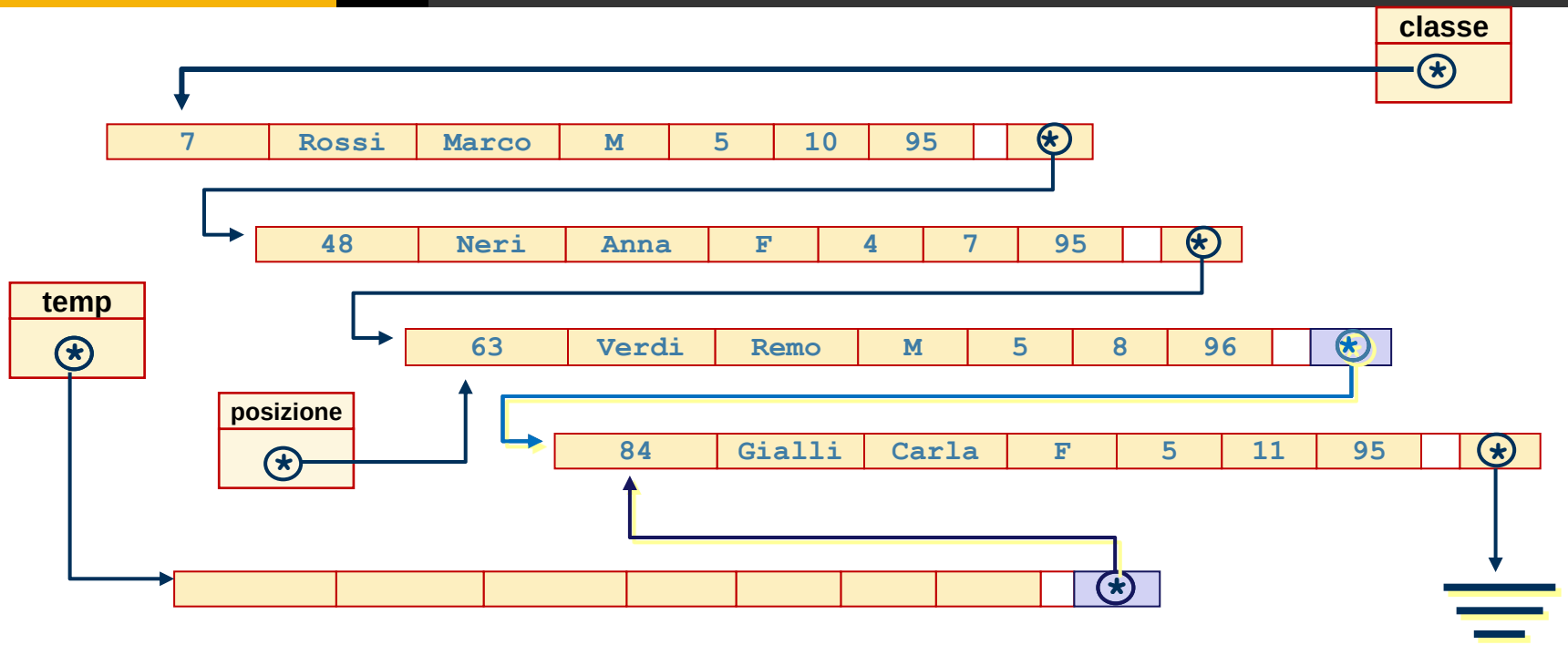

```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
   }
 else
   { temp->nextPtr = posizione->nextPtr;
```



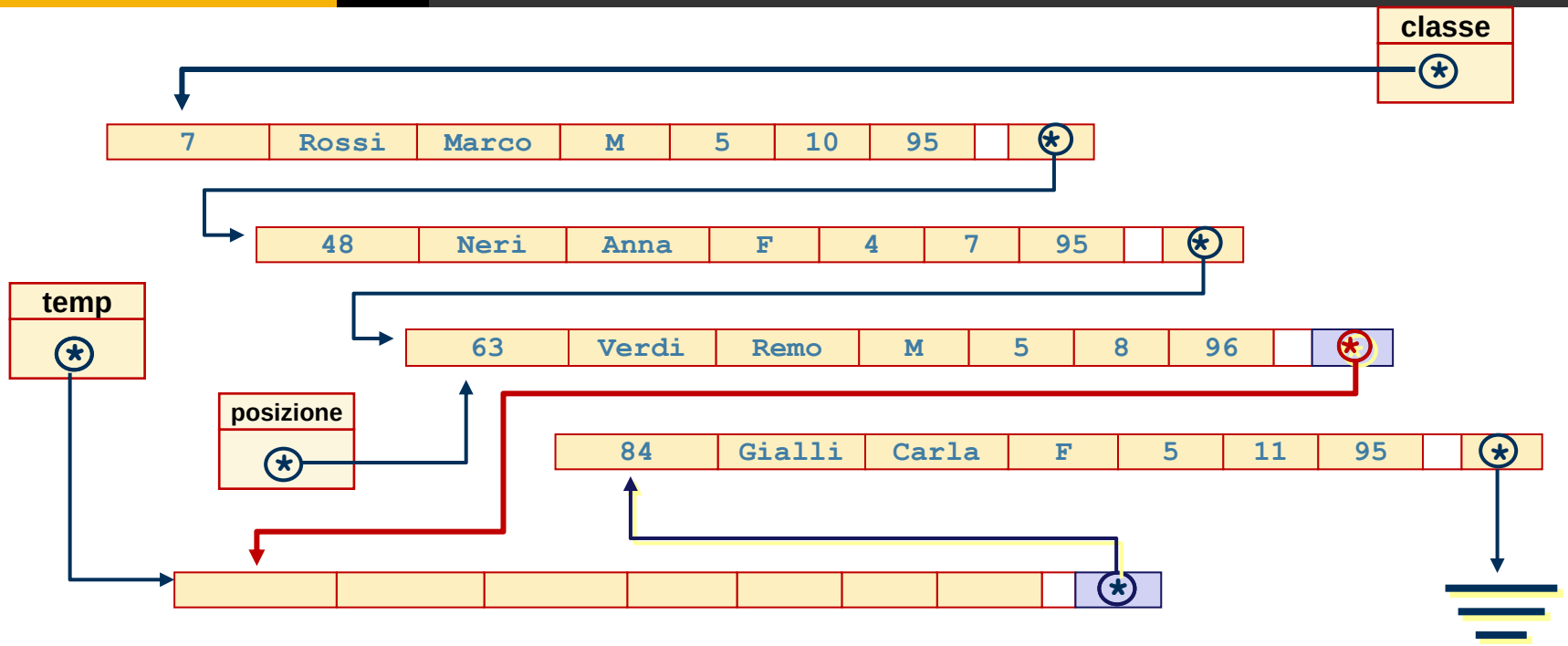

```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
   }
 else
   { temp->nextPtr = posizione->nextPtr;
```



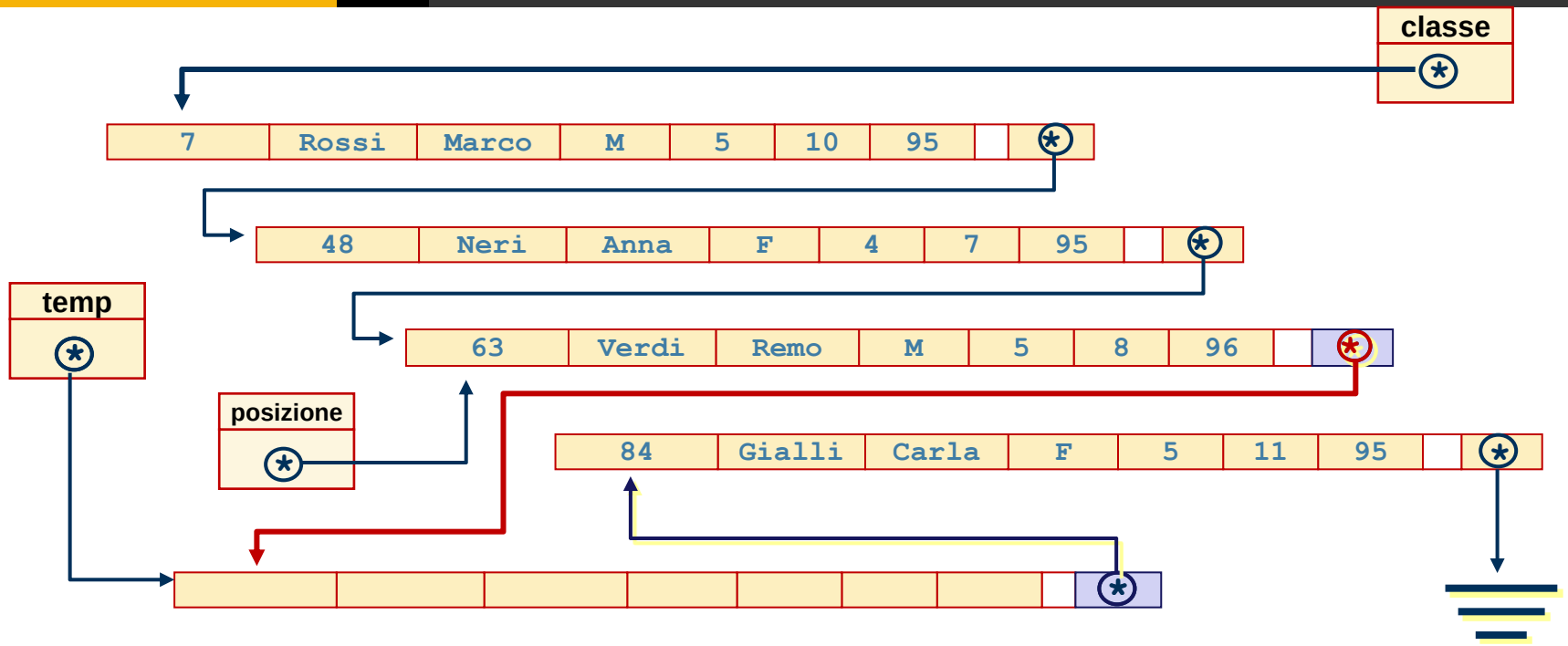

```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
   }
 else
   { temp->nextPtr = posizione->nextPtr;
```




```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
   }
 else
   { temp->nextPtr = posizione->nextPtr;
```




```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
   }
 else
   { temp->nextPtr = posizione->nextPtr;
     posizione->nextPtr = temp;
   }
```




```
temp = new Nodo;
if (posizione == 0)
   { temp->nextPtr = tab;
     tab = temp;
   }
 else
   { temp->nextPtr = posizione->nextPtr;
     posizione->nextPtr = temp;
   }
```

```
void inserisciSeNonEsiste(Studente nuovoStudente, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(nuovoStudente.matricola,
                          tab, esiste, posizione);
  if (esiste == FALSO)
    { cout << "inserimento di studente con matricola"
           << setw(5) << nuovoStudente.matricola << endl;
      Nodo *temp;
      temp = new Nodo;
      if (posizione == 0)
        { temp->nextPtr = tab;
          tab = temp;
        }
      else
        { temp->nextPtr = posizione->nextPtr;
          posizione->nextPtr = temp;
        }
      temp->datiStud = nuovoStudente;
    }
}
```


```
void inserisciSeNonEsiste(Studente nuovoStudente, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(nuovoStudente.matricola,
                          tab, esiste, posizione);
  if (esiste == FALSO)
    { cout << "inserimento di studente con matricola"
           << setw(5) << nuovoStudente.matricola << endl;
      Nodo *temp;
      temp = new Nodo;
      if (posizione == 0)
        { temp->nextPtr = tab;
          tab = temp;
        }
      else
        { temp->nextPtr = posizione->nextPtr;
          posizione->nextPtr = temp;
        }
      temp->datiStud = nuovoStudente;
    }
}
```


```
void inserisciSeNonEsiste(Studente nuovoStudente, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(nuovoStudente.matricola,
                          tab, esiste, posizione);
  if (esiste == FALSO)
    { cout << "inserimento di studente con matricola"
           << setw(5) << nuovoStudente.matricola << endl;
      Nodo *temp;
      temp = new Nodo;
      if (posizione == 0)
        { temp->nextPtr = tab;
          tab = temp;
        }
      else
        { temp->nextPtr = posizione->nextPtr;
          posizione->nextPtr = temp;
        }
      temp->datiStud = nuovoStudente;
    }
}
```

```
void eliminaSeEsiste(int matricola, Nodo *&tab)
{
  int esiste, posizione;
  cercaSeEsisteEPosizione(matricola, tab, esiste, posizione);
  if (esiste == VERO)
    { cout << "eliminazione di studente con matricola"
           << setw(5) << matricola << endl;
      int temp = posizione +1;
      for (int i = temp +1; i < tab.num; i++)
        tab.elenco[i-1] = tab.elenco[i];
      tab.num--;
    }
}
```

```
void eliminaSeEsiste(int matricola, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(matricola, tab, esiste, posizione);
  if (esiste == VERO)
    { cout << "eliminazione di studente con matricola"
           << setw(5) << matricola << endl;
      int temp = posizione +1;
      for (int i = temp +1; i < tab.num; i++)
        tab.elenco[i-1] = tab.elenco[i];
      tab.num--;
    }
}
```

```
void eliminaSeEsiste(int matricola, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(matricola, tab, esiste, posizione);
  if (esiste == VERO)
    { cout << "eliminazione di studente con matricola"
           << setw(5) << matricola << endl;
      int temp = posizione +1;
      for (int i = temp +1; i < tab.num; i++)
        tab.elenco[i-1] = tab.elenco[i];
      tab.num--;
    }
}
```

```
void eliminaSeEsiste(int matricola, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(matricola, tab, esiste, posizione);
  if (esiste == VERO)
    { cout << "eliminazione di studente con matricola"
           << setw(5) << matricola << endl;
      int temp = posizione +1;
      for (int i = temp +1; i < tab.num; i++)
        tab.elenco[i-1] = tab.elenco[i];
      tab.num--;
    }
}
```

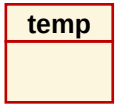
pos = -1

```
void eliminaSeEsiste(int matricola, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(matricola, tab, esiste, posizione);
  if (esiste == VERO)
    { cout << "eliminazione di studente con matricola"
           << setw(5) << matricola << endl;
      Nodo *temp;
      if (posizione == 0)
        { temp = tab;
          tab = tab->nextPtr;
        }
      else
        { temp = posizione->nextPtr;
          posizione->nextPtr = temp->nextPtr;
        }
      delete temp;
    }
}
```

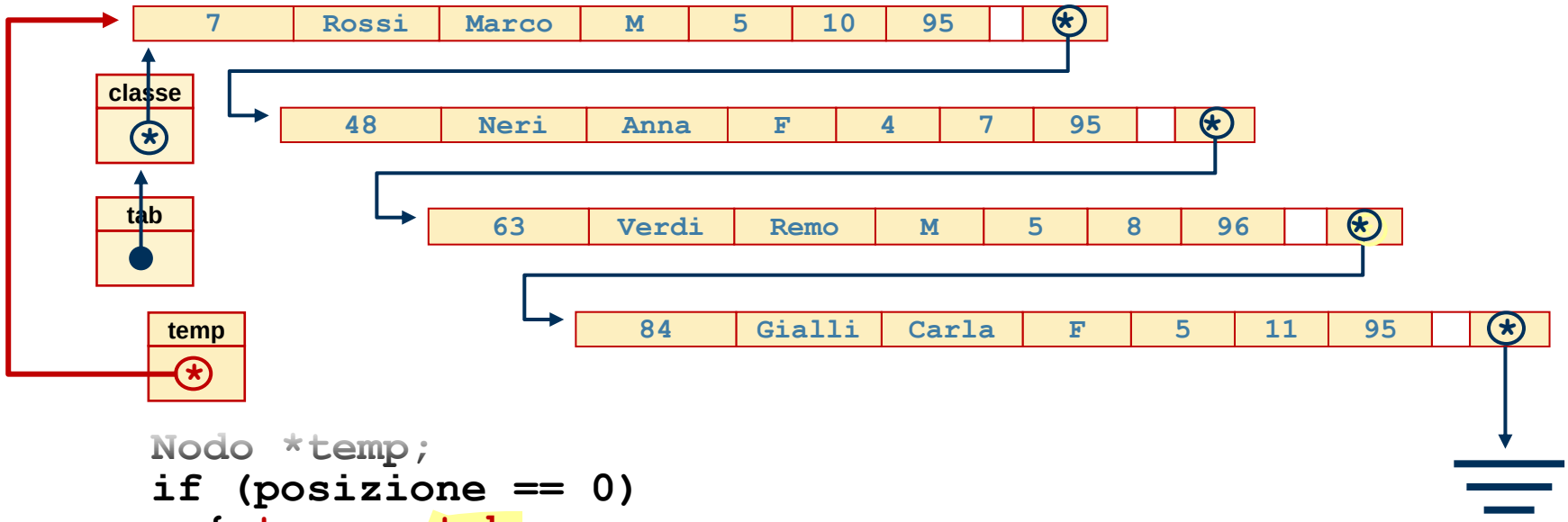
```
void eliminaSeEsiste(int matricola, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(matricola, tab, esiste, posizione);
  if (esiste == VERO)
    { cout << "eliminazione di studente con matricola"
           << setw(5) << matricola << endl;
      Nodo *temp;
      if (posizione == 0)
        { temp = tab;
          tab = tab->nextPtr;
        }
      else
        { temp = posizione->nextPtr;
          posizione->nextPtr = temp->nextPtr;
        }
      delete temp;
    }
}
```

```
void eliminaSeEsiste(int matricola, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(matricola, tab, esiste, posizione);
  if (esiste == VERO)
    { cout << "eliminazione di studente con matricola"
           << setw(5) << matricola << endl;
      Nodo *temp;
      if (posizione == 0)
        { temp = tab;
          tab = tab->nextPtr;
        }
      else
        { temp = posizione->nextPtr;
          posizione->nextPtr = temp->nextPtr;
        }
      delete temp;
    }
}
```

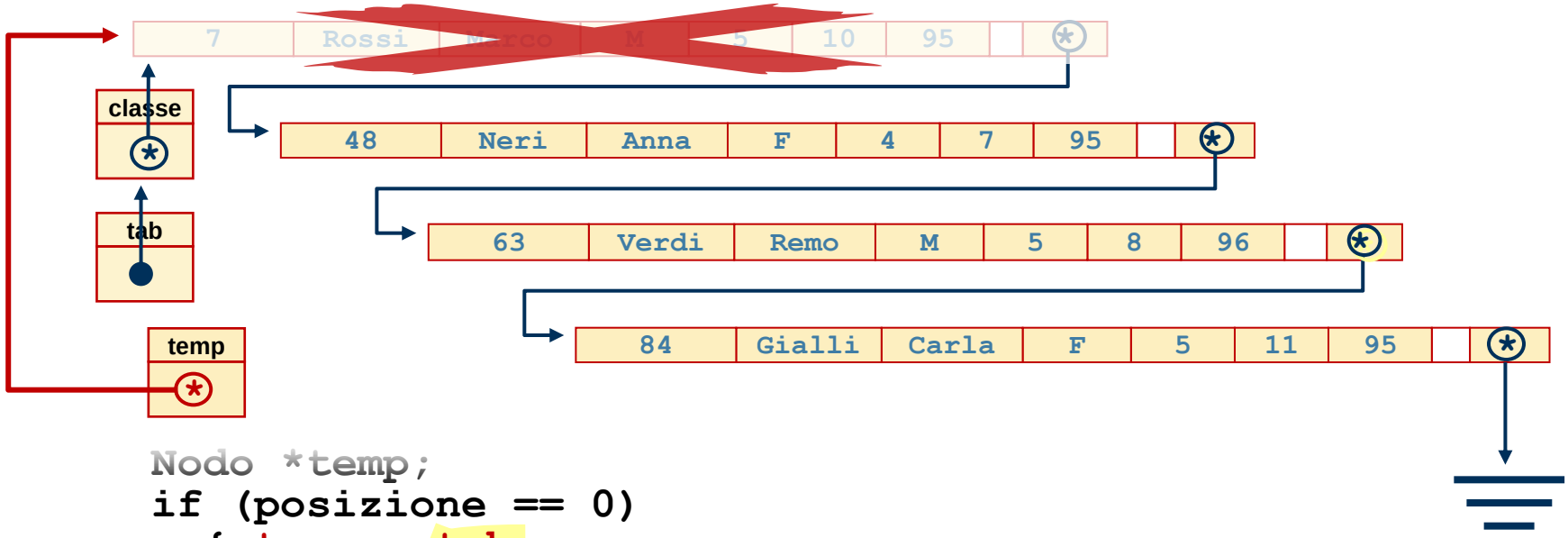
```
void eliminaSeEsiste(int matricola, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(matricola, tab, esiste, posizione);
  if (esiste == VERO)
    { cout << "eliminazione di studente con matricola"
           << setw(5) << matricola << endl;
      Nodo *temp;
      if (posizione == 0)
        { temp = tab;
          tab = tab->nextPtr;
        }
      else
        { temp = posizione->nextPtr;
          posizione->nextPtr = temp->nextPtr;
        }
      delete temp;
    }
}
```



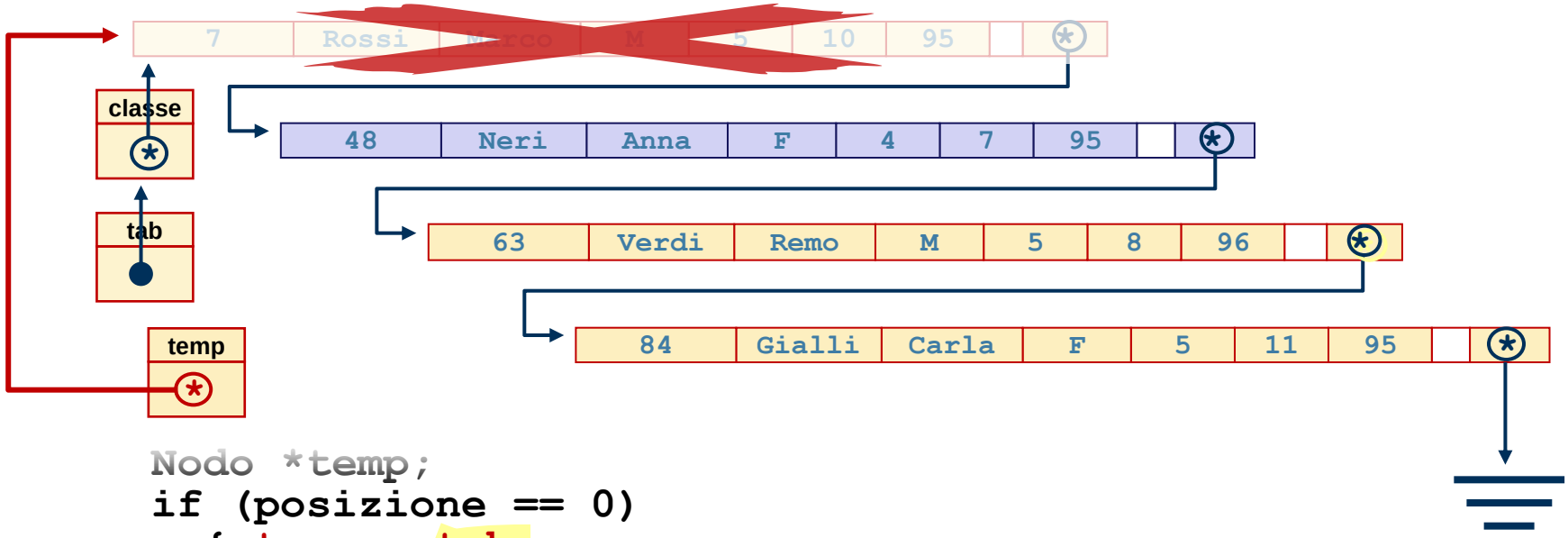

```
        Nodo *temp;
        if (posizione == 0)
          { temp = tab;
            tab = tab->nextPtr;
          }
        else
          { temp = posizione->nextPtr;
            posizione->nextPtr = temp->nextPtr;
          }
        delete temp;
    }
}
```



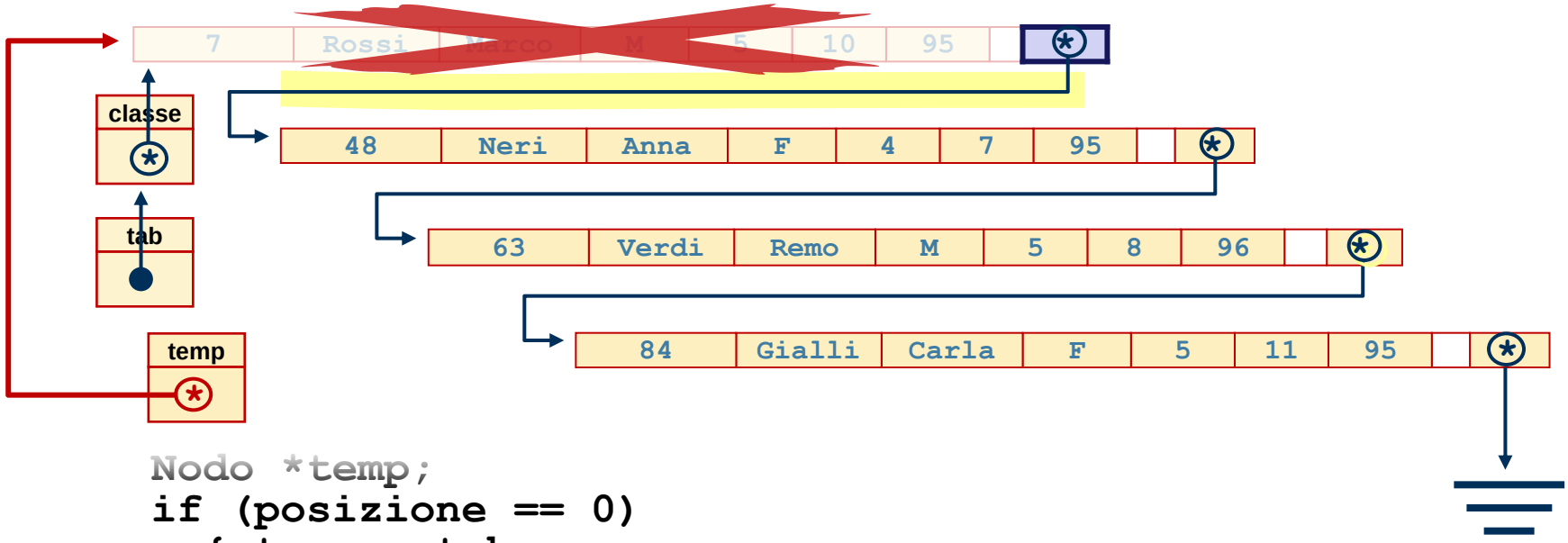

```
        Nodo *temp;
        if (posizione == 0)
          { temp = tab;
            tab = tab->nextPtr;
          }
        else
          { temp = posizione->nextPtr;
            posizione->nextPtr = temp->nextPtr;
          }
        delete temp;
    }
}
```



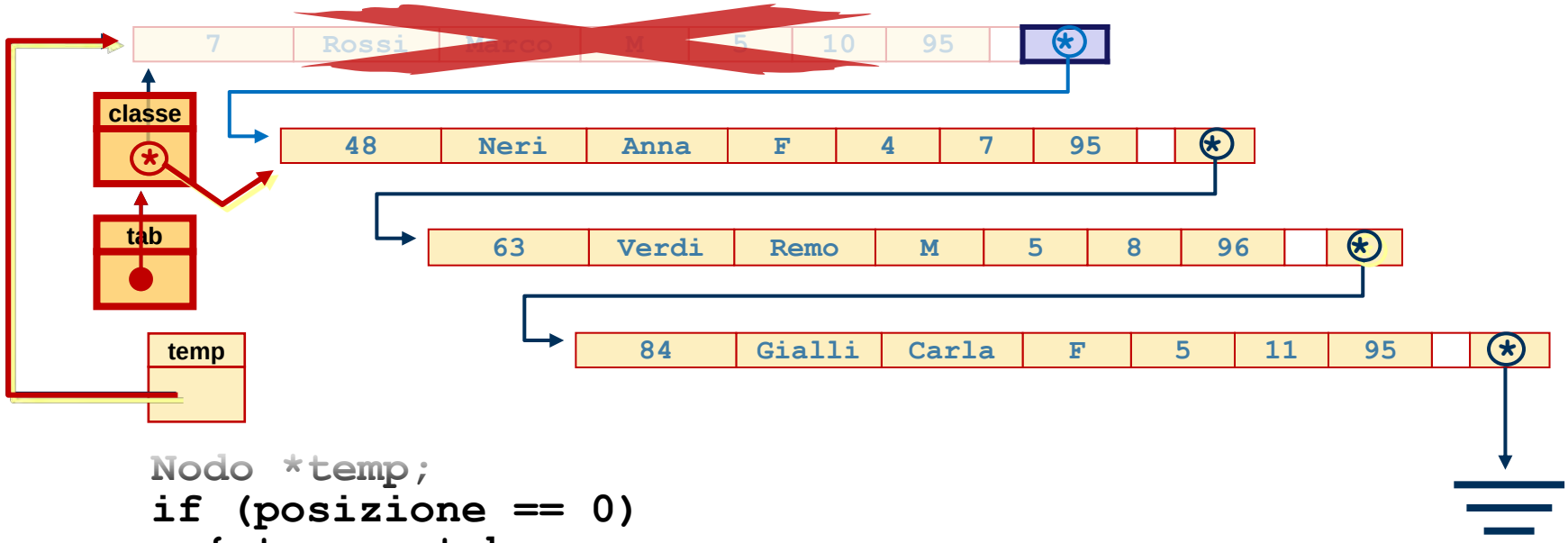

```
        Nodo *temp;
        if (posizione == 0)
          { temp = tab;
            tab = tab->nextPtr;
          }
        else
          { temp = posizione->nextPtr;
            posizione->nextPtr = temp->nextPtr;
          }
        delete temp;
    }
}
```



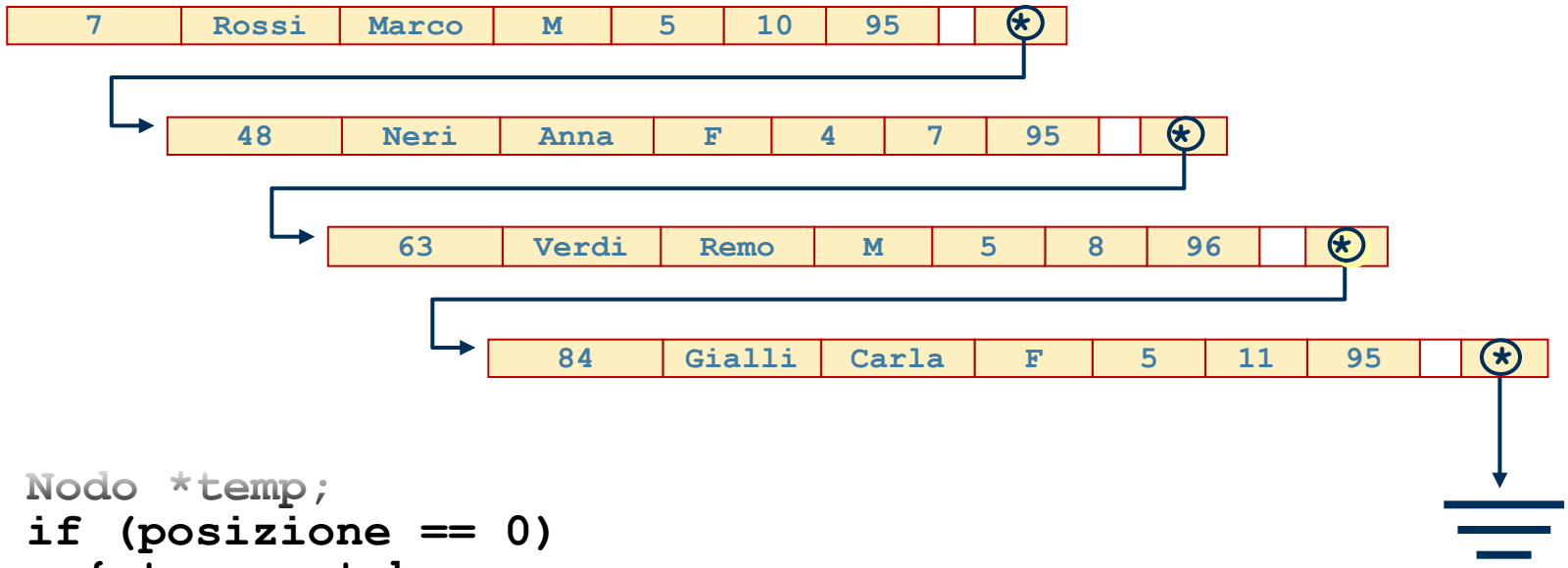

```
        Nodo *temp;
        if (posizione == 0)
          { temp = tab;
            tab = tab->nextPtr;
          }
        else
          { temp = posizione->nextPtr;
            posizione->nextPtr = temp->nextPtr;
          }
        delete temp;
    }
}
```



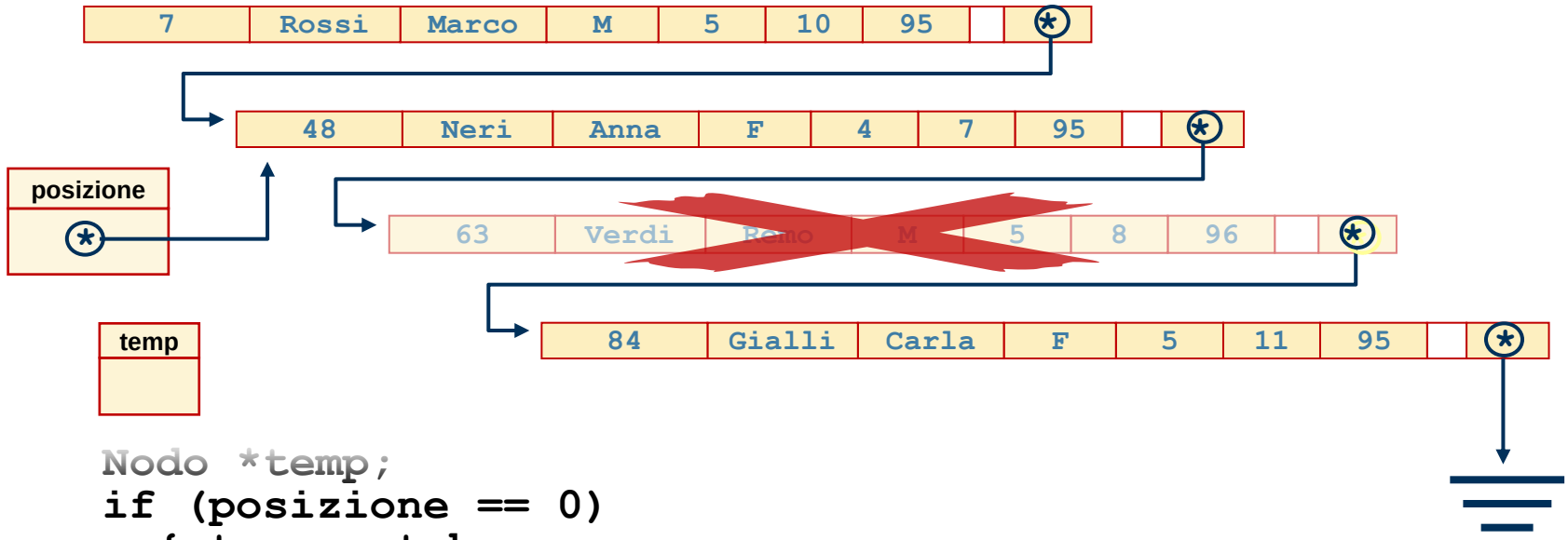

```
        Nodo *temp;
        if (posizione == 0)
          { temp = tab;
            tab = tab->nextPtr;
          }
        else
          { temp = posizione->nextPtr;
            posizione->nextPtr = temp->nextPtr;
          }
        delete temp;
    }
}
```



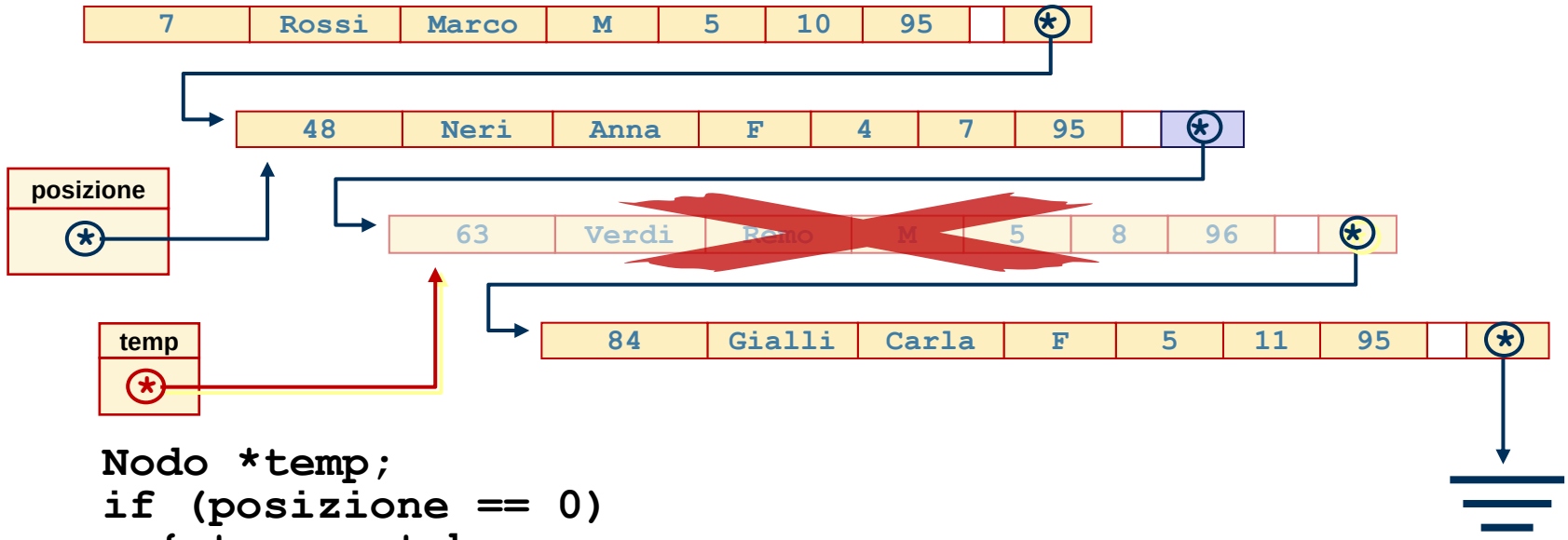

```
        Nodo *temp;
        if (posizione == 0)
          { temp = tab;
            tab = tab->nextPtr;
          }
        else
          { temp = posizione->nextPtr;
            posizione->nextPtr = temp->nextPtr;
          }
        delete temp;
    }
}
```



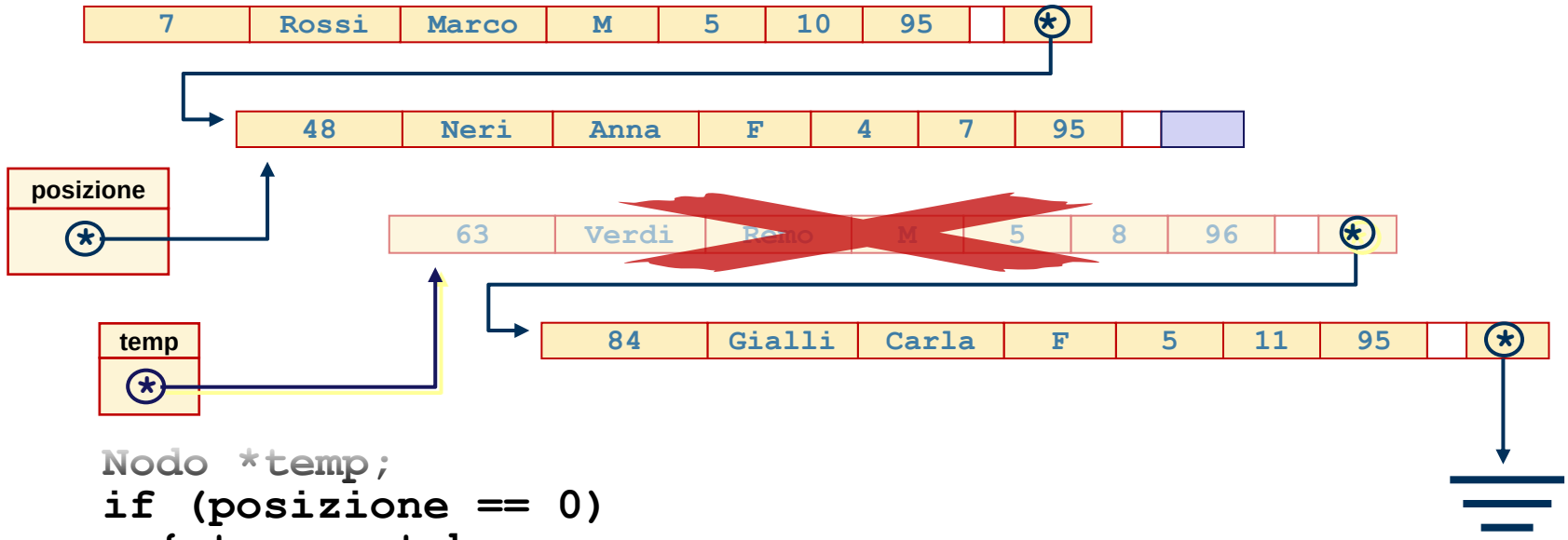

```
      Nodo *temp;
      if (posizione == 0)
        { temp = tab;
          tab = tab->nextPtr;
        }
      else
        { temp = posizione->nextPtr;
          posizione->nextPtr = temp->nextPtr;
        }
      delete temp;
    }
}
```



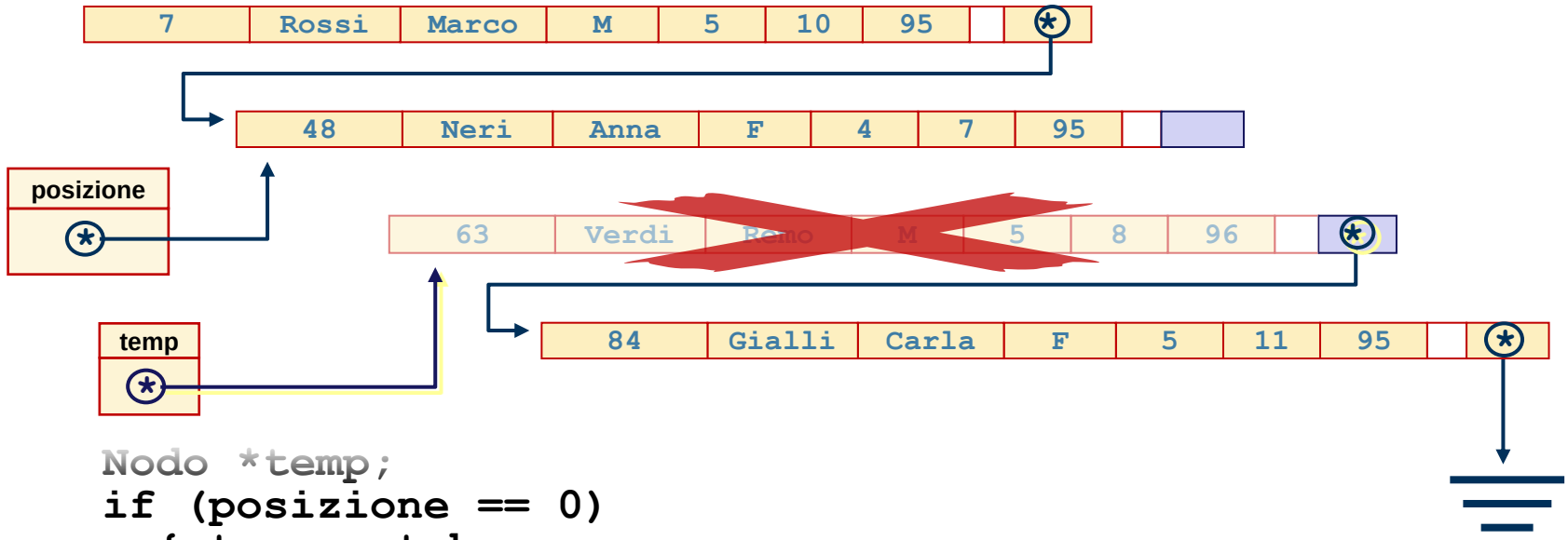

```
        Nodo *temp;
        if (posizione == 0)
          { temp = tab;
            tab = tab->nextPtr;
          }
        else
          { temp = posizione->nextPtr;
            posizione->nextPtr = temp->nextPtr;
          }
        delete temp;
    }
}
```



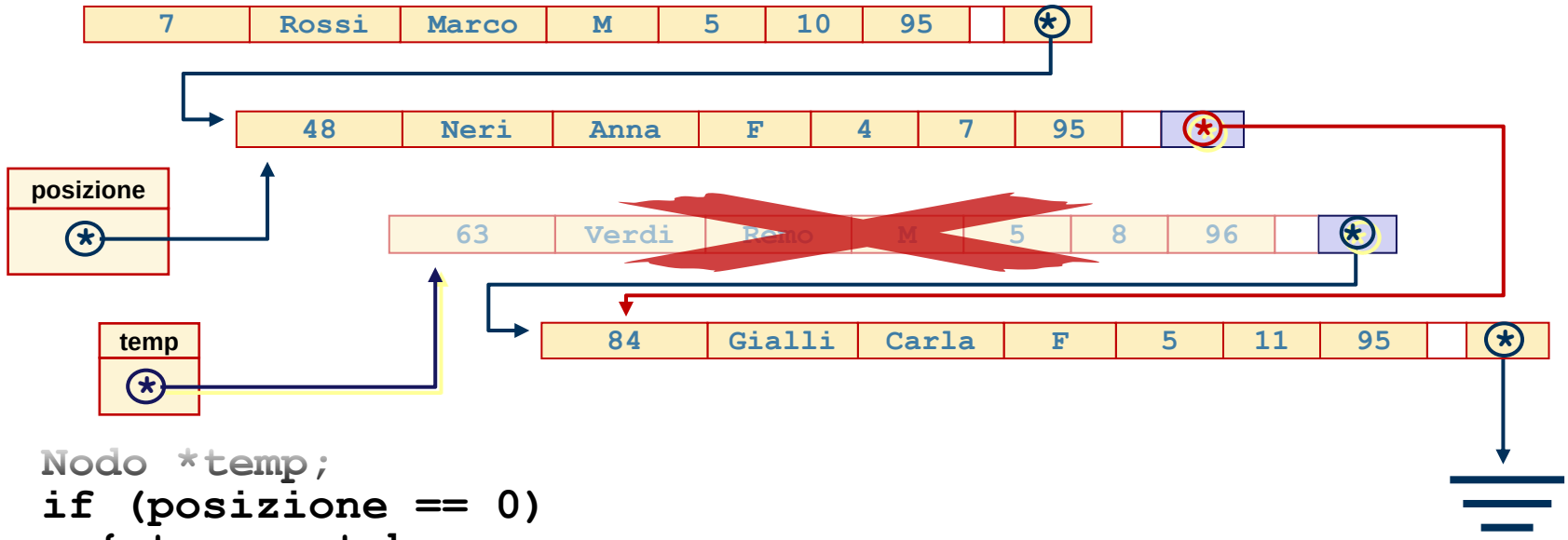

```
        Nodo *temp;
        if (posizione == 0)
          { temp = tab;
            tab = tab->nextPtr;
          }
        else
          { temp = posizione->nextPtr;
            posizione->nextPtr = temp->nextPtr;
          }
        delete temp;
    }
}
```




```
        Nodo *temp;
        if (posizione == 0)
          { temp = tab;
            tab = tab->nextPtr;
          }
        else
          { temp = posizione->nextPtr;
            posizione->nextPtr = temp->nextPtr;
          }
        delete temp;
    }
}
```

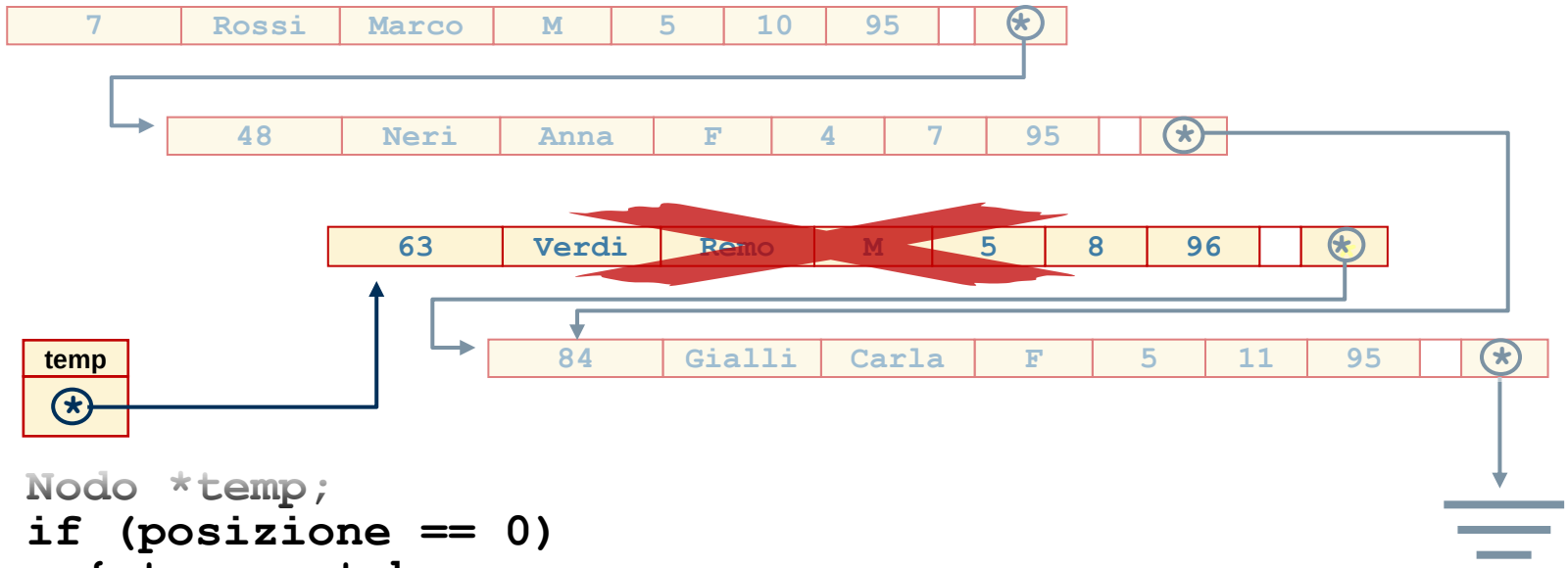



```
        Nodo *temp;
        if (posizione == 0)
          { temp = tab;
            tab = tab->nextPtr;
          }
        else
          { temp = posizione->nextPtr;
            posizione->nextPtr = temp->nextPtr;
          }
        delete temp;
    }
}
```

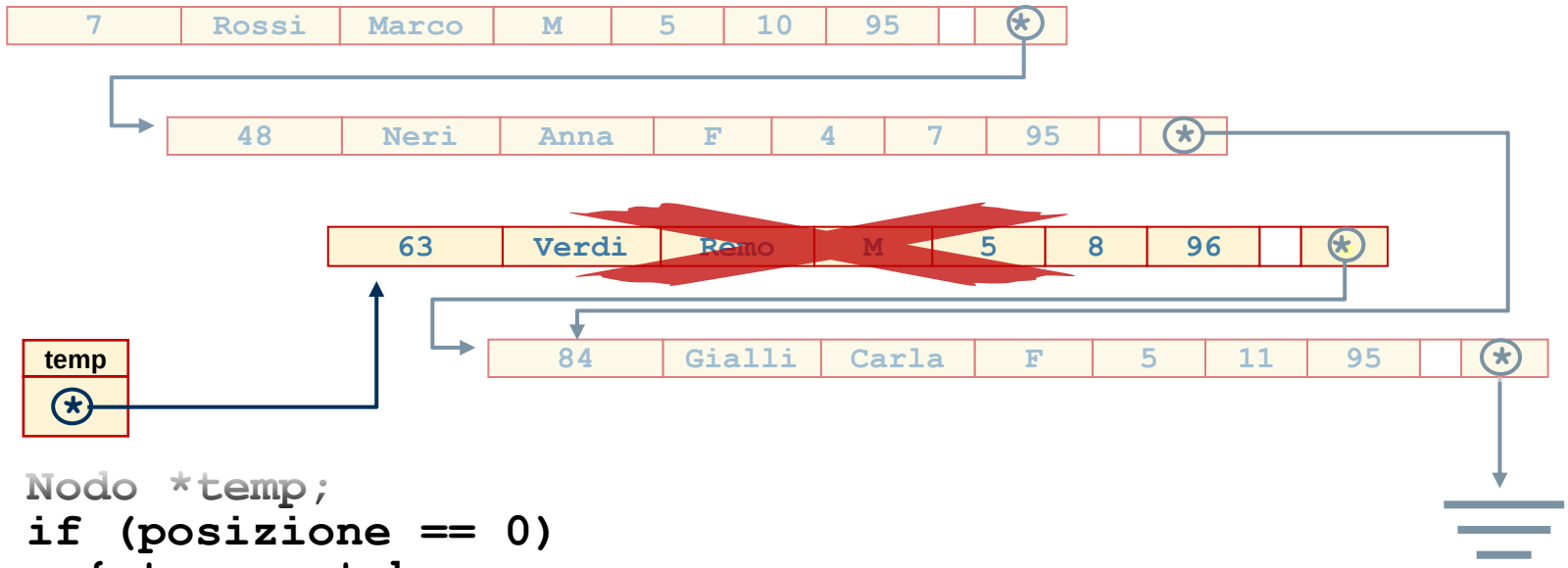



```
    Nodo *temp;
    if (posizione == 0)
       { temp = tab;
         tab = tab->nextPtr;
       }
    else
       { temp = posizione->nextPtr;
         posizione->nextPtr = temp->nextPtr;
       }
    delete temp;
  }
}
```

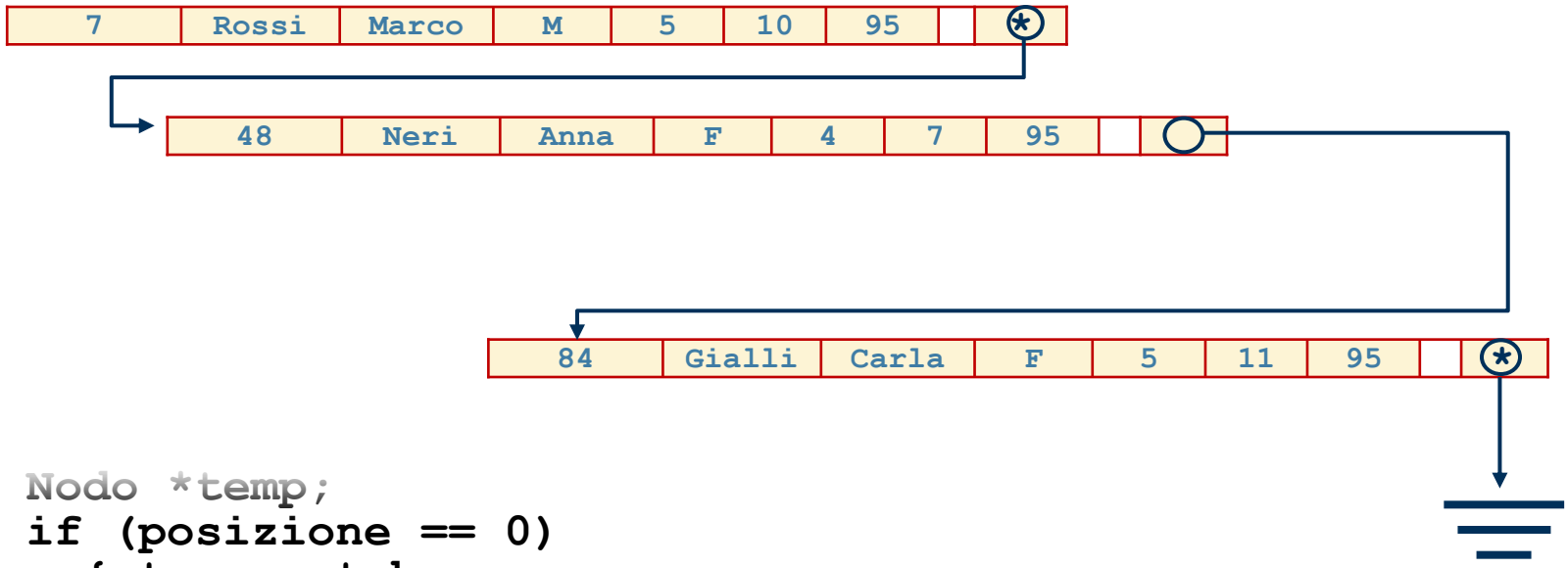
```
void eliminaSeEsiste(int matricola, Nodo *&tab)
{
  int esiste;
  Nodo *posizione;
  cercaSeEsisteEPosizione(matricola, tab, esiste, posizione);
  if (esiste == VERO)
    { cout << "eliminazione di studente con matricola"
           << setw(5) << matricola << endl;
      Nodo *temp;
      if (posizione == 0)
        { temp = tab;
          tab = tab->nextPtr;
        }
      else
        { temp = posizione->nextPtr;
          posizione->nextPtr = temp->nextPtr;
        }
      delete temp;
    }
}
```




```
      Nodo *temp;
      if (posizione == 0)
        { temp = tab;
          tab = tab->nextPtr;
        }
      else
        { temp = posizione->nextPtr;
          posizione->nextPtr = temp->nextPtr;
        }
      delete temp;
    }
}
```




```
      Nodo *temp;
      if (posizione == 0)
        { temp = tab;
          tab = tab->nextPtr;
        }
      else
        { temp = posizione->nextPtr;
          posizione->nextPtr = temp->nextPtr;
        }
      delete temp;
    }
}
```




```
      Nodo *temp;
      if (posizione == 0)
        { temp = tab;
          tab = tab->nextPtr;
        }
      else
        { temp = posizione->nextPtr;
          posizione->nextPtr = temp->nextPtr;
        }
      delete temp;
    }
}
```

```
void stampaTabella(Nodo *&tab)
{
  int i;
  cout << " tabella con il solo campo chiave " << endl;
  for(i = 0; i < tab.num; i++)
    cout << setw(5) << tab.elenco[i].matricola;
  cout << endl << endl;
}
```

```
void stampaTabella(Nodo *&tab)
{
  int i;
  cout << " tabella con il solo campo chiave " << endl;
  for(i = 0; i < tab.num; i++)
    cout << setw(5) << tab.elenco[i].matricola;
  cout << endl << endl;
}
```

```
void stampaTabella(Nodo *&tab)
{
  Nodo *studPtr;
  cout << " tabella con il solo campo chiave " << endl;
  for(i = 0; i < tab.num; i++)
    cout << setw(5) << tab.elenco[i].matricola;
  cout << endl << endl;
}
```

```
void stampaTabella(Nodo *&tab)
{
  Nodo *studPtr;
  cout << " tabella con il solo campo chiave " << endl;
  for(i = 0; i < tab.num; i++)
    cout << setw(5) << tab.elenco[i].matricola;
  cout << endl << endl;
}
```

```
void stampaTabella(Nodo *&tab)
{
  Nodo *studPtr;
  cout << " tabella con il solo campo chiave " << endl;
  for (studPtr = tab; studPtr != 0; studPtr = studPtr->nextPtr)
    cout << setw(5) << studPtr->datiStud.matricola;

  cout << endl << endl;
}
```

```
void stampaTabella(Nodo *&tab)
{
  Nodo *studPtr;
  cout << " tabella con il solo campo chiave " << endl;
  for (studPtr = tab; studPtr != 0; studPtr = studPtr->nextPtr)
    cout << setw(5) << studPtr->datiStud.matricola;

  cout << endl << endl;
}
```

```
void stampaTabella(Nodo *&tab)
{
  Nodo *studPtr;
  cout << " tabella con il solo campo chiave " << endl;
  for (studPtr = tab; studPtr != 0; studPtr = studPtr->nextPtr)
    cout << setw(5) << studPtr->datiStud.matricola;

  cout << endl << endl;
}
```

```
void stampaTabella(Nodo *&tab)
{
  Nodo *studPtr;
  cout << " tabella con il solo campo chiave " << endl;
  for (studPtr = tab; studPtr != 0; studPtr = studPtr->nextPtr)
    cout << setw(5) << studPtr->datiStud.matricola;

  cout << endl << endl;
}
```

```
void stampaTabella(Nodo *&tab)
{
  Nodo *studPtr;
  cout << " tabella con il solo campo chiave " << endl;
  for (studPtr = tab; studPtr != 0; studPtr = studPtr->nextPtr)
    cout << setw(5) << studPtr->datiStud.matricola;

  cout << endl << endl;
}
```

classe

num		elenco						
4		matricola	cognome	nome	sesso	dataNascita		
						giorno	mese	anno
	0	7	Rossi	Marco	M	5	10	95
	1	48	Neri	Anna	F	4	7	95
	2	63	Verdi	Remo	M	5	8	96
	3	84	Gialli	Carla	F	5	11	95
	4