Tipo lista

Tipo astratto lista

Prof. G. Ascia

Una lista è una sequenza (ordinata) di elementi di un tipo atomo

Es. Una lista di interi è una sequenza ordinata di numeri interi

Una lista potremmo rappresentarla come:

L=(1 8 5 3) L= (1 3) L=(7 1 6) L=()

Per il tipo astratto lista possiamo definire le funzioni primitive:

- lista_vuota: verifica se una lista è vuota

  `lista_vuota: lista→ boolean`
- testa: restituisce l'elemento in testa alla lista

  `testa: lista → atomo`
- in_testa: inserisce un elemento in testa alla lista

  `in_testa: listaxatomo → lista`
- da_testa: elimina l'elemento in testa alla lista

  `da_testa: lista → lista`

Tipo astratto lista

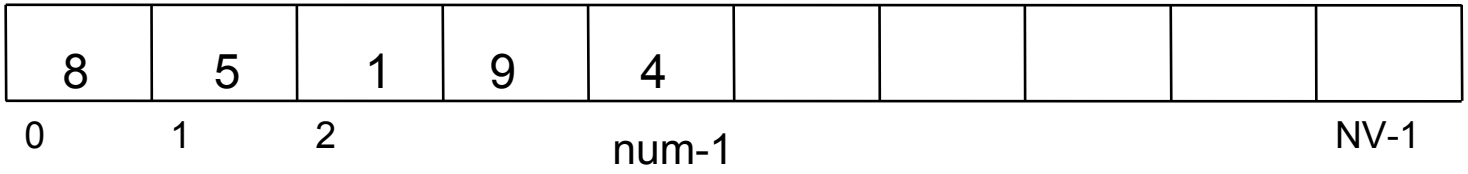
- Il tipo astratto lista può essere definito come la tripla Lista=<S,F,C>
  - S={lista, atomo, boolean} con lista dominio di interesse
  - F={lista_vuota, testa, in_testa, da_lista}
  - C={lista_vuota}

  - `lista_vuota: lista→ boolean`
  - `testa: lista → atomo`
  - `in_testa: lista × atomo → lista`
  - `da_testa: lista → lista`

Rappresentazione sequenziale mediante vettore di dimensione fissa di una lista

Prof. G. Ascia

- La lista può essere rappresentata mediante un vettore di NV elementi di tipo atomo e una variabile **num** indicante il numero di elementi in lista
- Ad esempio la lista L=(4, 9, 1, 5, 8) può essere rappresentata nel seguente modo:



- Nel caso in cui la lista è vuota, la variabile **num** assume valore 0. Nel caso di inserimento in testa (se il vettore non è pieno) l'elemento nuovo viene inserito in posizione num e il valore della variabile **num** viene incrementato .
- Nel caso di eliminazione dell'elemento in testa alla lista, (se la lista non è già vuota) viene decrementato il valore della variabile num.

Rappresentazione sequenziale mediante vettore di dimensione fissa di una lista

- La lista di interi possiamo rappresentarla all'interno di un programma C mediante la seguente struct:

```
struct lista
{ int L[NV];
  int num;
};
```

- La funzione lista_vuota restituisce 1 se list.testa è uguale a NV, 0 altrimenti.

```
int lista_vuota( struct lista list)
{
  if(list.num==0) return 1;
  else return 0;
}
```

Rappresentazione sequenziale mediante vettore di dimensione fissa di una lista

- La funzione testa restituisce il valore dell'elemento in testa alla lista.

```
int testa (struct lista list)
{
   return list.L[list.num-1];
}
```

- La funzione in_testa inserisce l'elemento **e** in testa alla lista se il vettore non è pieno

```
void in_testa (struct lista *pl, int e)
{
 if(pl->num< NV) /* Se il vettore non e' pieno */
  { pl->L[pl->num]=e;
    pl->num++;
  }
 else printf("Vettore pieno\n");
}
```

Rappresentazione sequenziale mediante vettore di dimensione fissa di una lista

- La funzione da_testa elimina l'elemento in testa alla lista, se la lista non è già vuota(!lista_vuota(*pl)).

```
void da_testa (struct lista *pl)
{
 if(!lista_vuota(*pl))
    pl->num--;
 else printf("La lista e' gia' vuota\n");
}
```

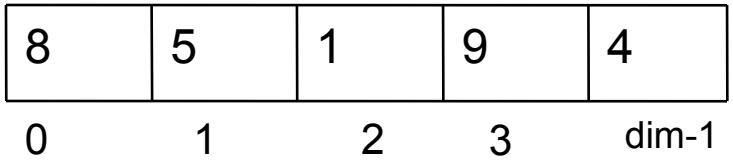
Problema con questo tipo di rappresentazione:

al massimo NV elementi possono essere inseriti nella lista.

Rappresentazione sequenziale mediante vettore di dimensione variabile di una lista

Prof. G. Ascia

- La lista può essere rappresentata mediante un vettore allocato dinamicamente di elementi di tipo atomo e una variabile **dimensione** indicante la dimensione del vettore.
- Ad esempio la lista L=(4, 9, 1, 5, 8) può essere rappresentata nel seguente modo:



- Nel caso in cui la lista è vuota, la dimensione assume valore 0
- Nel caso di inserimento in testa viene aumentata la dimensione del vettore, il valore della variabile **dimensione** viene incrementato e l'elemento nuovo viene inserito in posizione dimensione-1.
- Nel caso di eliminazione dell'elemento in testa alla lista, (se la lista non è già vuota) viene decrementato il valore della variabile dimensione e viene ridotta la dimensione del vettore.

Rappresentazione sequenziale mediante vettore di dimensione variabile di una lista

- La lista di interi possiamo rappresentarla all'interno di un programma C mediante la seguente struct:

```
struct lista
{ int *L;
  int dim;
};
struct lista list;
```

Rappresentazione sequenziale mediante vettore di dimensione variabile di una lista


```
int lista_vuota( struct lista list)
{
  if(list.dimensione==0) return 1;
  else return 0;
}

int testa (struct lista list)
{
   return list.L[list.dimensione-1];
}
```

Rappresentazione sequenziale mediante vettore di dimensione variabile di una lista

Prof. G. Ascia

```
void in_testa (struct lista *pl, int e)
{
 int *Vaux;

 Vaux=realloc(pl->L,sizeof(int)*(pl->dim+1));
   if(!Vaux) return; /* Se l'allocazione non ha successo
                        la funzione viene interrotta  */
   else { pl->L=Vaux;
          pl->dim+=1;
        }
 }
  /* Inserimento */
  pl->L[pl->dimensione-1]=e;
  pl->dimensione++;
}
```

Rappresentazione sequenziale mediante vettore di dimensione variabile di una lista


```
void da_testa (struct lista *pl)
{
 if(!lista_vuota(*pl))
 { pl->dimensione--;
   pl->L=realloc(pl->L,sizeof(int)*(pl->dimensione));
 }
 else printf("La lista e' gia' vuota\n");
}
```

Rappresentazione sequenziale mediante vettore di dimensione variabile di una lista

- **Problemi con questo tipo di rappresentazione**:
  - Nel caso di operazioni di inserimento o di cancellazione di elementi della lista in posizione diversa rispetto alla testa della lista, è richiesto un numero considerevole di operazioni di assegnamento, dovuto allo spostamento degli elementi del vettore per garantire che tutti gli elementi siano in posizione adiacente.