

Tipi astratti di dati

Tipo astratto di dato

Prof. G. Ascia

- Facendo riferimento a un tipo di dato si è detto che un tipo indica l'insieme di valori che possono essere assunti.
- Es.
 - il tipo **int** è il sottoinsieme dei numeri interi che una variabile di tipo int può assumere;
 - Il tipo **double** è il sottoinsieme dei numeri reali rappresentati in doppia precisione che una variabile di tipo double può assumere.
- Ciascun tipo di dato, oltre ad essere caratterizzato dai valori che può assumere, può essere caratterizzato da un insieme di operazioni primitive sui valori del tipo e da alcune costanti.
- Ad esempio, il tipo **double** può essere caratterizzato dalle quattro operazioni elementari: +, -, ×, /, dai predicati >, < e dagli elementi neutri della somma e del prodotto, 0.0 e 1.0.
- Le quattro operazioni hanno come dominio il prodotto cartesiano **double** × **double** e codominio **double**.

Tipo astratto di dato

Prof. G. Ascia

- I due predicati hanno come codominio il prodotto cartesiano $\text{double} \times \text{double}$ e codominio l'insieme booleano $\{\text{false}, \text{true}\}$
- Il tipo di dato potremmo dunque pensarlo come :
 - Un insieme di domini: sottoinsieme dei reali, boolean;
 - Un insieme di operazioni e predicati: $+$, $-$, $*$, $/$, $<$, $>$;
 - Un insieme di costanti: 0.0 e 1.0

Tipo insieme su V

Prof. G. Ascia

- Il tipo insieme su V è un tipo i cui elementi sono costituiti da elementi di tipo V :
- Es. Il tipo insieme su $V=\{0,1,2,3,4,5,6,7,8,9\}$ è il tipo i cui elementi sono un sottoinsieme di V .

Per il tipo insieme su V possiamo definire le seguenti funzioni primitive:

- **test_insieme_vuoto**: verifica se un insieme è vuoto
`test_insieme_vuoto: insieme → boolean`
- **test_appartiene**: verifica se un elemento e appartiene a un insieme
`test_appartiene: insieme × V → boolean`
- **in_insieme**: inserisce un elemento in un insieme
`in_insieme: insieme × V → insieme`
- **da_insieme**: elimina un elemento da un insieme
`da_insieme: insieme × V → insieme`

Tipo insieme su V

Prof. G. Ascia

- Il tipo insieme su V possiamo caratterizzarlo con il seguente valore costante: l'insieme vuoto.
- Dunque, il tipo insieme su V, può essere definito mediante la terna $T = \langle S, F, C \rangle$
 - $S = \{ \text{insieme}, V, \text{boolean} \}$
 - $F = \{ \text{test_insieme_vuoto}, \text{test_appartiene}, \text{in_insieme}, \text{da_insieme} \}$
 - $C = \{ \text{insieme_vuoto} \}$

dove ***insieme*** è il dominio di interesse del tipo insieme su V.

- Nel definire il tipo insieme su V non è stato fatto alcun riferimento al modo in cui esso deve essere rappresentato all'interno di un programma. Pertanto quanto definito sopra viene definito ***tipo astratto insieme su V***.

Tipo astratto di dato

Prof. G. Ascia

- Un tipo astratto di dato è una tripla $T = \langle S, F, C \rangle$
dove:
 - $S = \{Dom_1, Dom_2, \dots, Dom_N\}$
 - $F = \{f_1: Dom_{11} \times Dom_{1k} \rightarrow Dom_{1m}, \dots, f_r: Dom_{r1} \times Dom_{rk} \rightarrow Dom_{rm}\}$
 - $C = \{C_1, \dots, C_i\}$
- con $Dom_{ij} \in S$ e con uno dei domini $Dom_i \in S$ che costituisce il dominio di interesse.

Rappresentazione di un tipo astratto di dato

Prof. G. Ascia

- Questo problema si presenta quando un tipo astratto deve essere usato in un programma.
- E' necessario rappresentare il tipo astratto in termini di costrutti (tipi, variabili, ecc.) del linguaggio di programmazione in cui il programma deve essere scritto.
- Una rappresentazione di un tipo astratto deve fornire le regole per rappresentare:
 1. i domini in S ;
 2. le operazioni in F ;
 3. le costanti in C .
- Il primo punto riguarda la rappresentazioni dei domini in termini di tipi e costrutti del linguaggio.
- Il secondo riguarda la definizione delle operazioni primitive in termini dei tipi usati per rappresentare i domini.
- Il terzo punto riguarda la rappresentazione delle costanti nel linguaggio di programmazione

Rappresentazione 1 del tipo astratto insieme su $V=\{0,1,\dots,9\}$

Prof. G. Ascia

- Una possibile rappresentazione dei valori del dominio di interesse può essere un vettore di int avente tanti elementi quanti sono gli elementi presenti in V .
- Un elemento e appartiene a un insieme su V se l'elemento del vettore di indice e ha valore 1, altrimenti non appartiene ovvero:

$$V[e]= \begin{cases} 1 & \text{appartiene all'insieme} \\ 0 & \text{non appartiene all'insieme} \end{cases}$$

- L'insieme $I=\{2,6,8\}$ verrebbe rappresentato dal vettore

0	0	1	0	0	0	1	0	1	0
0	1	2	3	4	5	6	7	8	9

- Il tipo booleano rappresentiamo mediante il tipo int
- La costante insieme vuoto può essere rappresentata mediante un vettore i cui elementi hanno tutti valore nullo.

Rappresentazione 1 del tipo astratto insieme su $V=\{0,1,\dots,9\}$

Prof. G. Ascia

- La primitiva `test_insieme_vuoto` viene definita usando la seguente funzione che restituisce il valore 0 se nel ciclo for sul vettore `V` viene trovato un elemento con valore 1.
- Se tutti gli elementi hanno valore nullo, dopo il ciclo for viene restituito 1.

```
int test_insieme_vuoto (int *V)
{
    int i;

    for(i=0;i<NV;i++)
        if(V[i]) /* L'elemento i e' presente */
            return 0;
    return 1;
}
```

- La primitiva `test_appartiene` viene definita usando la seguente funzione che restituisce il valore dell'elemento `V[e]`.

```
int test_appartiene (int *V, int e)
{
    return V[e];
}
```

Rappresentazione 1 del tipo astratto insieme su $V=\{0,1,\dots,9\}$

Prof. G. Ascia

- La primitiva `in_insieme` viene definita dalla seguente funzione che inserisce l'elemento `e` ponendo $V[e]=1$

```
void in_insieme (int *V, int e)
{
    V[e]=1;
}
```

- La primitiva `da_insieme` viene definita dalla seguente funzione che elimina l'elemento `e` ponendo $V[e]=0$

```
void da_insieme (int *V, int e)
{
    V[e]=0;
}
```

Rappresentazione 2 del tipo astratto insieme su $V=\{0,1,\dots,9\}$

Prof. G. Ascia

- Una rappresentazione alternativa dei valori del dominio di interesse può essere la seguente:
 - un vettore di int di dimensione pari a quella di V
 - una variabile di tipo int indicante il numero di elementi presenti.
- L'insieme $I=\{2,6,8\}$ verrebbe rappresentato dal vettore

2	6	8							
0	1	2	3	4	5	6	7	8	9

`numelementi=3;`

L'insieme vuoto è rappresentato dal valore 0 della variabile **numelementi**.

Rappresentazione 2 del tipo astratto insieme su $V=\{0,1,\dots,9\}$

Prof. G. Ascia

- La primitiva `test_insieme_vuoto` viene definita usando la seguente funzione che restituisce il valore 0 se la il numero di elementi (indicato da parametro `num`) è diverso da zero, 1 altrimenti.

```
int test_insieme_vuoto (int num)
{
    return !num;
}
```

- La primitiva `test_appartiene` viene definita usando la seguente funzione che restituisce un valore ≥ 0 se l'elemento appartiene all'insieme, -1 altrimenti.
- La ricerca viene effettuata mediante un ciclo `for` che viene interrotto se l'elemento appartiene all'insieme, restituendone la posizione in cui si trova. Se l'elemento non appartiene il ciclo `for` viene completato e viene restituito -1.

```
int test_appartiene (int *V, int num, int e)
{
    int i;

    for (i=0; i<num; i++)
        if (V[i]==e) return i;
    return -1;
}
```

Rappresentazione 2 del tipo astratto insieme su $V=\{0,1,\dots,9\}$

Prof. G. Ascia

- La primitiva `in_insieme` viene definita dalla seguente funzione che inserisce l'elemento `e` in posizione `*pnum` e incrementa `*pnum`.

```
void in_insieme (int *V, int *pnum, int e)
{
    if (appartiene (V, *pnum, e) == -1)
        /* L'elemento non appartiene gia' all'insieme */
        {
            V[*pnum]=e;
            (*pnum)++;
        }
    else printf("\nElemento gia' presente");
}
```

Rappresentazione 2 del tipo astratto insieme su $V=\{0,1,\dots,9\}$

Prof. G. Ascia

- La primitiva `da_insieme` viene definita dalla seguente funzione che elimina l'elemento `e` dall'insieme se esso appartiene all'insieme.
- Tale funzione sfrutta la funzione `appartiene` per ottenere, se l'elemento appartiene all'insieme, la sua posizione.

```
void da_insieme (int *V, int *pnum, int e)
{  int i, pos;

   pos=appartiene(V, *pnum, e);
   /* pos è la posizione dell'elemento */

   /* Eliminazione da vettore dell'elemento di indice pos*/
   if( pos>=0)
       { for(i=pos; i<*pnum-1; i++)
           V[i]=V[i+1];
         (*pnum) --;
       }
}
```