Lezione 9 e 10

- Segmenti di memoria e puntatori
- Aritmetica dei puntatori
- Formattazione dei dati
- Significato ed impiego del cast
- Tipi di dato strutturato

Fabio Scotti (2004-2009)

Laboratorio di programmazione per la sicurezza

Valentina Ciriani (2005-2009)

Laboratorio di programmazione

Fabio Scotti (2004-2009)
Laboratorio di programmazione per la sicurezza

Valentina Ciriani (2005-2009)
Laboratorio di programmazione

Lezione 9 e 10

Segmenti di memoria e puntatori

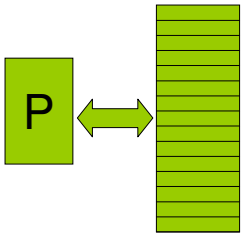
Obiettivi :

- Come è composta la memoria che usa un programma in esecuzione
- Comprendere cosa è un puntatore e come possa essere utilizzato per accedere a vettori e matrici

Rappresentazione della memoria

Si rappresenti la memoria dell'elaboratore come un insieme di celle contigue.

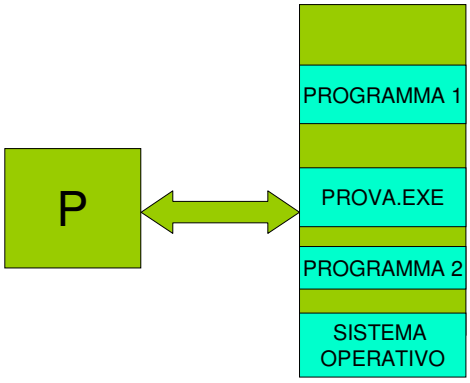
Il processore legge e scrive le celle della memoria per gestire i dati o per eseguire programmi.



3

Caricamento di un programma

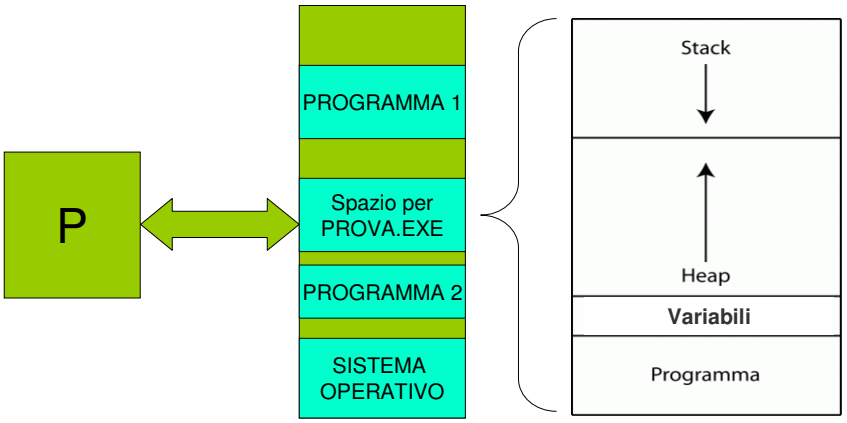
Quando si manda in esecuzione un programma eseguibile esso viene caricato ed il processore inizia ad eseguirlo.



4

Struttura di un programma

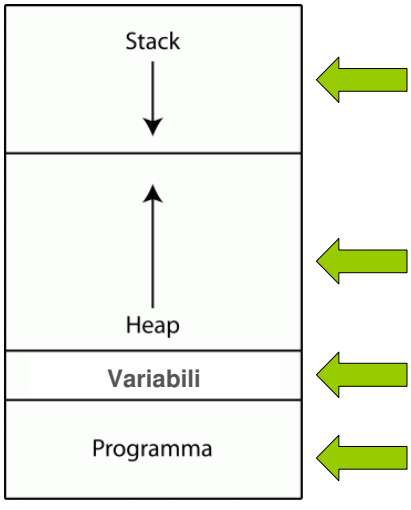
Lo spazio che il processore dedica ad un programma in esecuzione (processo) è divisibile in segmenti.



5

Quattro segmenti

Ogni segmento assolve una funzione precisa.



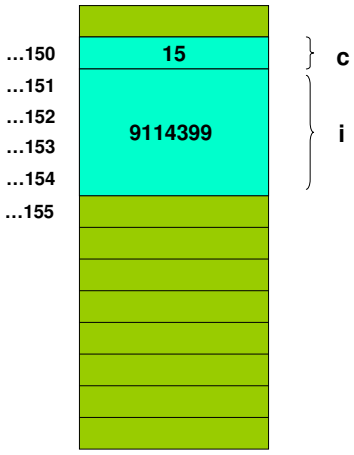
- Dati e parametri delle funzioni
- Dati dinamici
- Variabili definite:
  - esempio: int indice;
- Codice eseguibile del programma

6

Allocazione (1)

Segmento variabili: cosa succede quando vengono dichiarate le variabili?

```
…
char c;
int i;
```

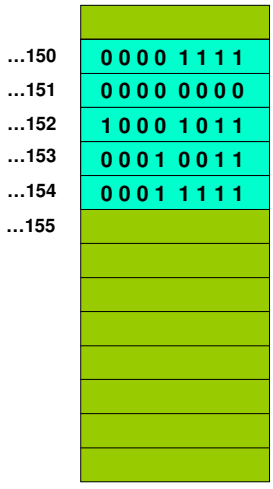


7

Allocazione (2)

Focalizzare l'attenzione sul vero contenuto celle.

```
…
char c;
int i;
```



c 0 0 0 0 1 1 1 1 (15)

i 0 0 0 0 0 0 0 .... 1 1 1 1 (9114399)

8

Puntatore (1)

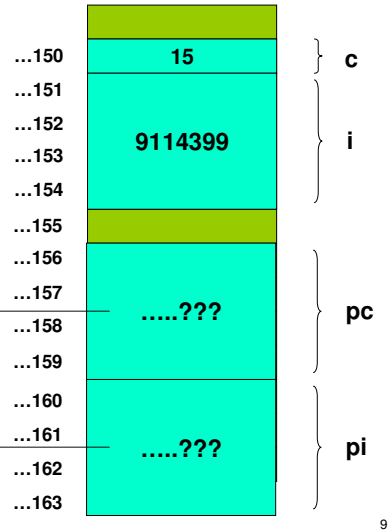
Il puntatore è una cella che contiene un indirizzo di una variabile.

```
…
char c;
int i;

char * pc;
int * pi;
```

??
Momentaneamente si sa che punta ad un carattere.

??
Momentaneamente si sa che punta ad un intero.



9

Puntatore (2)

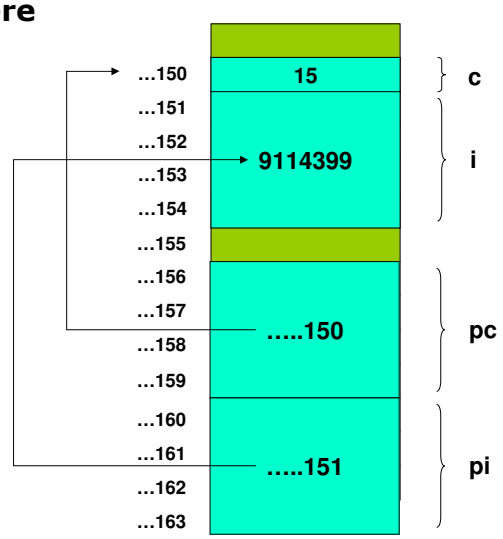
Il puntatore deve essere inizializzato.

L'indirizzo di una cella viene letto mediante l'operatore '&'.

```
…
char c;
int i;

char * pc;
int * pi;

pc = &c;
pi  = &i;
```



10

Uso del puntatore (1)

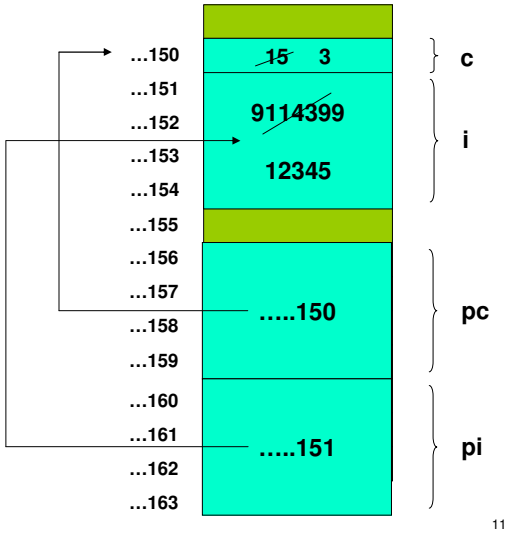
Usare l'operatore '*' e il puntatore per accedere alla cella puntata.

```
…
char c;
int i;

char * pc;
int * pi;

pc = &c;
pi  = &i;

*pc = 3;
*pi =  12345;
```



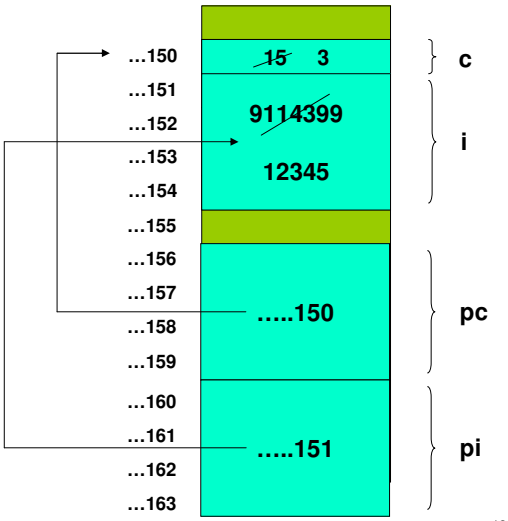
11

Uso del puntatore (2)

Il puntatore e l'operatore '*' permettono di leggere o scrivere.

```
…
printf("%c", c );
printf("%c", *pc );
…

*pc = 3;
*pi =  12345;
```



12

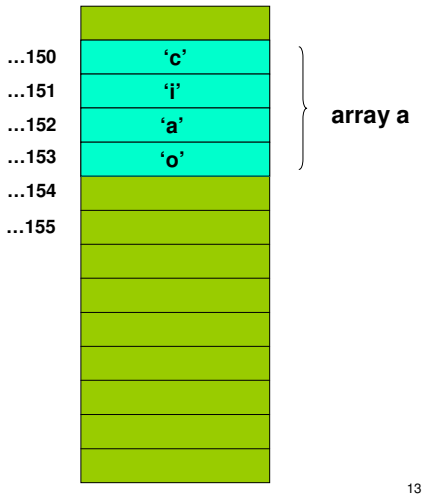
Puntatore ed array

Il nome di un array è il puntatore alla sua cella di inizio.

```
char a[4] ;
a[0]='c'; a[1]='i';
a[2]='a'; a[3]='o';

printf("%d", a);   //→ …150


printf("%c", a[0]);          //→ c
printf("%d", &a[0]);   //→ …150
```



13

In sintesi …

- **I puntatori sono celle contenenti solo degli indirizzi e sono dichiarati esprimendo il tipo di dato puntato :**
  - esempio: **char * pc;**
- **Leggere l'indirizzo di una cella usando '&' :**
  - esempio: **pc = &c;**
- **Accedere in lettura o scrittura ad una cella puntata da un puntatore mediante il segno '*' :**
  - esempio: ***pc = 'z';**

14

Fabio Scotti (2004-2009)
Laboratorio di programmazione per la sicurezza

Valentina Ciriani (2005-2009)
Laboratorio di programmazione

Lezione 9 e 10

Aritmetica dei puntatori

Obiettivi :

- Essere in grado di utilizzare correttamente un puntatore sfruttando l'aritmetica dei puntatori
- Riconoscere gli errori nei programmi derivanti dalla non corretta applicazione dell'aritmetica dei puntatori

Aritmetica dei puntatori in C

L'aritmetica dei puntatori è un ausilio alla programmazione per accedere alle celle degli array di dati.

!! Attenzione !!

Le spiegazioni che seguono funzionano solo su array di dati.

16

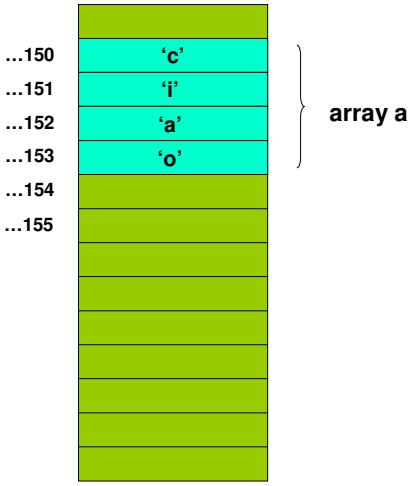
Esempio (1)

Array di caratteri :

```
char a[4] ;
char * pc;

pc = a; // oppure pc = &a[0];

printf("%d", pc);          //→ …150
printf("%d", pc+1);        //→ …151
// fino qui tutto ok…
```



17

Esempio (2)

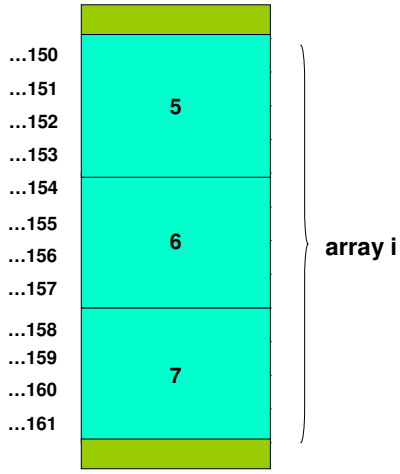
Array di interi (occupano 4 celle):

```
int i[3] ;
int * pc;

pc = i; // oppure pc = &i[0];
i[0]=5;   i[1]=6;   i[2]=7;

printf("%d", pc);          //→ …150
printf("%d", pc+1);        //→ …154

// vediamo 154 e non 151!!
```



18

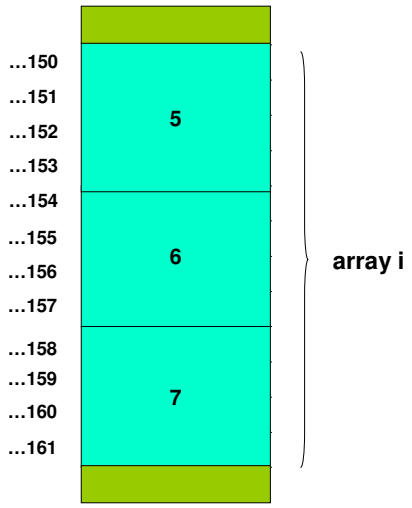
Uso dell'aritmetica sugli array

Si può accedere alle celle dell'array ricorrendo all'aritmetica dei puntatori:

```
int i[3] ;
int * pc;

pc = i; // oppure pc = &i[0];
* pc      = 5;
*(pc + 1) = 6 ;
*(pc + 2) = 7 ;

// in realtà *(pc + 2) funziona
// come *(pc + 2* sizeof(int) )
```



19

In sintesi ...

- **Usare l'aritmetica dei puntatori solo quando si accede ad array.**
- **Usare gli indici** (esempio **a[i]**) **quando è possibile.**
- **Porre molta attenzione a quando si fanno somme e differenze di puntatori: interviene l'aritmetica dei puntatori e si possono avere risultati inaspettati.**

20

Fabio Scotti (2004-2009)
Laboratorio di programmazione per la sicurezza

Valentina Ciriani (2005-2009)
Laboratorio di programmazione

Lezione 9 e 10

Formattazione dei dati

Obiettivo :

- Essere in grado di gestire correttamente l'output di numeri, vettori e stringhe

Formattazione dei dati

- **Caratteri e numeri possono essere mostrati a video o salvati su file di testo in modo formattato.**

```
Hai passato al programma 6 argomenti
In ordine:
[0] : ElencaArgomenti
[1] : 12
[2] : Esplora risorse
[3] : laboratorio
[4] : di
[5] : informatica


I dati raccolti duranti una sessione di esperimenti sono :
[A]     [B]     v
2.06    1.32    0.05
0.445   1.32    0.047
0.178   1.32    0.043
0.089   1.32    0.036
2.06    2.65    0.084
0.445   2.65    0.078
0.178   2.65    0.068
0.089   2.65    0.056
2.06    5.37    0.137
```

```
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
C C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
D D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
E E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
F F G H I J K L M N O P Q R S T U V W X Y Z A B C D E
G G H I J K L M N O P Q R S T U V W X Y Z A B C D E F
H H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
I I J K L M N O P Q R S T U V W X Y Z A B C D E F G H
J J K L M N O P Q R S T U V W X Y Z A B C D E F G H I
K K L M N O P Q R S T U V W X Y Z A B C D E F G H I J
L L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
M M N O P Q R S T U V W X Y Z A B C D E F G H I J K L
N N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
O O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
P P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
Q Q R S T U V W X Y Z A B C D E F G H I J K L M N O P
R R S T U V W X Y Z A B C D E F G H I J K L M N O P Q
S S T U V W X Y Z A B C D E F G H I J K L M N O P Q R
T T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
U U V W X Y Z A B C D E F G H I J K L M N O P Q R S T
V V W X Y Z A B C D E F G H I J K L M N O P Q R S T U
W W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
X X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
Y Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
Z Z A B C D E F G H I J K L M N O P Q R S T U V W X Y
```

- **Questo si ottiene usando opportunamente delle opzioni della funzione printf.**

Funzione printf() (1)

- **Espressione generale:**

```
printf(stringa, espressione, espressione,
espressione...)
```

- **La funzione printf ha sempre come primo argomento una stringa di controllo.**
- **Dopo la stringa è possibile mettere un numero qualsiasi di espressioni, in base alla necessità.**
- **La stringa deve includere un segnaposto (esempio %d) per ciascuna espressione successivamente elencata.**

Funzione printf() (2)

- **Ogni valore delle espressioni viene stampato nel segnaposto corrispondente.**

Esempi:

```
a = 3;
printf("Il valore della variabile a e': %d", a);
```

Stamperà:

```
Il valore della variabile a e': 3
```

```
printf("Il valore della variabile a e': %d",a+1);
```

Stamperà:

```
Il valore della variabile a e': 4
```

Funzione printf() (3)

- **Usando più segnaposti ed espressioni:**

```
a = 3; b = 4;
printf("a vale %d, mentre b vale %d",a ,b );
```

Stamperà:

```
a vale 3, mentre b vale 4
```

25

Attenzione ai formati

- **Il tipo di segnaposto (%d, %c, %f, ecc..) deve essere scelto in modo coerente rispetto al tipo di dato presente nelle espressioni:**

```
printf("a vale %d", a ); // dove a e' un intero
```

Esempi:

- variabile **intera**
  (es. dichiarata `int i;`) → nella stringa usare %d
- variabile **carattere**
  (es. dichiarata `char x;`) → nella stringa usare %c
- variabile **floating point**
  (es. dichiarata `float z;`) → nella stringa usare %f

26

Segnaposto dell'ANSI C

Tipo	Espressione	A video
%c	char	singolo carattere
%d (%i)	int	intero con segno
%e (%E)	float or double	formato esponenziale
%f	float or double	reale con segno
%g (%G)	float or double	utilizza %f o %e in base alle esigenze
%o	int	valore base 8 senza segno
%p	pointer	valore di una variabile puntatore
%s	array of char	stringa (sequenza) di caratteri
%u	int	intero senza segno
%x (%X)	int	valore base 16 senza segno

27

Caratteri di controllo della formattazione

- **Non stampano caratteri visibili ma contribuiscono a formattare ciò che viene stampato:**

```
\b    cancella un carattere (backspace)
\n    nuova linea
\r    ritorna all'inizio linea senza andare a capo
\t    tabulatore
\'    apice
\0    null  (terminatore di stringa)
```

28

Esempio di formattazione: '\n'

- **In generale il carattere più usato è '\n'.**
- **E' usato per andare a capo inserendo una nuova riga.**

Esempio:

```
...
printf("prima riga ...\n");
printf("seconda riga ...\n");
```

Produce a monitor:

```
prima riga ...
seconda riga ...
```

29

Esempio di formattazione: '\t'

- **Molto usato è anche il tabulatore '\t'.**
- **E' usato per incolonnare i dati.**

Esempio:

```
...
printf("%f \t %f  \t %f  \n", a1, b1, c1);
printf("%f \t %f  \t %f  \n", a2, b2, c2);
```

Produce a monitor:

```
3.14     6.1233      4.123
2.345    234.233     23.23
```

30

Formattazione dei float (1)

- **Il segnaposto %f permette di formattare il numero di cifre totali, davanti e dietro la virgola:**

```
printf("%12f",  3.234343 );
// 12 cifre totali

printf("%12.4f",  3.234343 );
// 12 totali e 4 dietro la virgola

printf("%.4f",  3.234343 );
// fisso solo 4 cifre dietro la virgola
```

31

Formattazione dei float (2)

- **Consideriamo il seguente esempio di stringa di controllo alla quale passiamo il float x che vale di volta in volta 0, 0.5, 1, -1, 100, 1000, 10000, 12345, 10000, 123456:**

```
Printf("|%13.4f|%13.4e|%13.4g|\n", x, x, x);

Ecco le uscite output:                    13 caratteri
                                          ( )
|       0.0000|   0.0000e+00|            0|
|       0.5000|   5.0000e-01|          0.5|
|       1.0000|   1.0000e+00|            1|
|      -1.0000|  -1.0000e+00|           -1|
|     100.0000|   1.0000e+02|          100|
|    1000.0000|   1.0000e+03|         1000|
|   10000.0000|   1.0000e+04|        1e+04|
|   12345.0000|   1.2345e+04|    1.234e+04|
|  100000.0000|   1.0000e+05|        1e+05|
|  123456.0000|   1.2346e+05|    1.235e+05|
```

32

Fabio Scotti (2004-2009)
Laboratorio di programmazione per la sicurezza

Valentina Ciriani (2005-2009)
Laboratorio di programmazione

Lezione 9 e 10

Significato ed impiego del CAST

Obiettivo :

- Essere in grado di gestire correttamente la conversione tra tipi di dati diversi attraverso il cast

Significato del cast

- **E' un operatore che serve a convertire i dati da un tipo all'altro.**
- **Si consideri il cast fra tipi di dato built-in :**
  - char
  - int
  - double
  - float

Sintassi

- **Sintassi:**
  (tipo) espressione
- **Esempio:**

```
int x;
double d;
d = (double) x;
```

- **Nell'esempio considerato il cast provvede a copiare l'intero `x` nella variabile `d` tenendo conto che un double deve avere anche una parte dopo la virgola.**

35

Esempio

Memorizzare il risultato di una divisione fra interi:

```
// senza cast
int x;
double d;
d=( x )/2;  // se x = 5 d = 2 non 2.5!

// con cast
int x;
double d;
d=( (double) x )/2;   // d = 2.5 corretto!
```

Grazie al cast viene visto come un double

36

Arrotondamento

- **Arrotondare un valore ad un intero:**

```
int x;
double d;      // se d vale 3.14
x = (int) d;   // x varrà  3
```

- **Si applichi il concetto di arrotondamento in una chiamata a funzione:**

```
double y = ...; // un cateto di tipo float
disegnaUnRettangolo(x, (int) y, 0, 0);
```

NB: il cast da reale intero copia la parte intera senza veri arrotondamenti (v. `round()` e `ceil()`).

37

Fabio Scotti (2004-2009)
Laboratorio di programmazione per la sicurezza

Valentina Ciriani (2005-2009)
Laboratorio di programmazione

Lezione 9 e 10

Dati strutturati

Obiettivi :

- Comprendere l'importanza della progettazione dei tipi di dato come regola di buona programmazione
- Essere in grado di progettare ed impiegare correttamente tipi di dato definiti dall'utente
- Conoscere le modalità di accesso dei tipi di dato definiti dall'utente

Tipi di dato user-defined

- **Il linguaggio C, oltre ai tipi di dato predefiniti (tipi built-in), dispone di meccanismi per definire nuovi tipi di dato (tipi user-defined).**
- **Analizzeremo le regole generali che governano la definizione di nuovi tipi.**
- **Tutti i tipi non predefiniti utilizzati in un programma, devono essere dichiarati come ogni altro elemento del programma (di solito nella parte dichiarativa del programma).**

Meccanismi di costruzione

- **I meccanismi che useremo sono:**
  1. ridefinizione (`typedef`);
  2. enumerazione (`typedef enum`);
  3. strutturati user-defined (`struct`).

Ridefinizione (`typedef`)

- **Un nuovo tipo di dato può essere definito dando semplicemente un nuovo nome a un tipo già esistente.**
- **Sintassi della ridefinizione:**

```
typedef TipoEsistente NuovoTipo;
```

Può essere un tipo di dato built-in o user-defined.

Esempi:

```
typedef int anno;
typedef int naturale;
typedef char carattere;
```

41

Enumerazione (`typedef enum`)

- **Consente di definire un nuovo tipo di dato enumerando i suoi valori.**
- **Sintassi della enumerazione:**

```
typedef enum {v1, v2, ... , vn}
NuovoTipo;
```

Esempi:

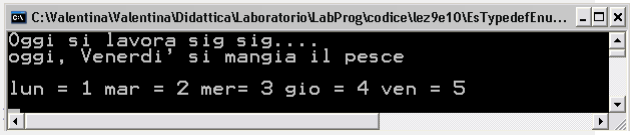
```
typedef enum {lun, mar, mer, gio, ven, sab, dom}
Giorno;
typedef enum {m, f} sesso;
```

42

Esempio Enumerazione

```
#include<stdio.h>
typedef enum {lun =1,mar,mer,gio,ven,sab,dom} settimana;
int main()
{
   settimana giorno;
   giorno = ven;
   if ((giorno == sab)||(giorno == dom) )
     printf("Oggi si non si fa nulla!!\n");
   else
      printf("Oggi si lavora sig sig....\n");
   switch (giorno)
   {
      case lun :
            printf("oggi,
      break;
      case mar :
            printf("oggi,
      break;
      case ven :
            printf("oggi, Venerdi' si mangia il pesce\n\n");
      break;
      default  :
         printf("Oggi si mangia pizza\n");
   }
   printf("lun = %d mar = %d mer= %d gio = %d ven = %d\n",lun,mar,mer,gio,ven);
   fflush(stdin);
   getchar();
}
```

Obblighiamo la numerazione a partire da 1.
Per default parte da 0.



Costruttore struct

- **Una struttura è una aggregazione di elementi che possono essere eterogenei.**

Esempio:

```
struct data {
int giorno;
int mese;
int anno;
};
```

Etichetta della struttura

Campi della struttura

Regole sui campi della struttura

- **Devono avere nomi univoci all'interno di una struttura.**
- **Strutture diverse possono avere campi con lo stesso nome.**
- **Un campo di un tipo struttura non può essere del tipo struttura che si sta definendo ma può essere un suo puntatore.**

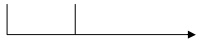
Esempio:

```
struct lista {int a;
              struct lista *p; };
```


Dichiarazione di variabili di tipo struttura

- **La definizione di una struttura non provoca allocazione di memoria, ma introduce un nuovo tipo di dato.**
- **Diverso è il caso in cui si dichiarano anche delle variabili (si può omettere l'etichetta):**

```
struct studente {
  char nome[20];
  long matricola;
  struct data ddn;
  } s1, s2;
```



VARIABILI DICHIARATE
di tipo struct studente

Accesso ai campi mediante operatore '.'

```
struct data {
int giorno;
int mese;
int anno;
};
typedef struct data Data;
…
Data oggi;
oggi.giorno = 10; oggi.mese = 10;
oggi.anno = 2004;
printf("%d %d %d", oggi.giorno, oggi.mese, oggi.anno);
```

47

Accesso ai campi mediante operatore '→'

```
//avendo definito il tipo Data
…
Data oggi,
Data * pd; // puntatore a data

pd = &oggi;
pd->giorno = 6;
pd->mese = 5;
pd->anno = 2004;

printf("%d %d %d",
   pd->giorno, pd-> mese, pd-> anno);
```

contenuto del campo giorno della struct puntata da pd

48

Accesso a struttura di struttura

```
struct dipendente
{ Persona   datiDip;
  Data      dataAssunzione;
};
typedef struct dipendente Dipendente;
Dipendente dip, *p;
...
dip.dataAssunzione.giorno = 3;
dip.dataAssunzione.mese = 4;
dip.dataAssunzione.anno = 1997;
...
(p->dataAssunzione).giorno = 5;
```


Esempio: MissItalia (1)

```
typedef struct {
      char nome[256] ;
      char cognome[256] ;
      int voto[3];
// voto[0] e' il voto della giuria (da 0 a 100),
// voto[1] quello della giuria speciale
// voto[2] il voto telefonico.
      int altezza;
      int peso;
      int cucina; // 0= aiuto!!!  10= ristorante
} MissItalia;     // Iniziale maiuscola
```

Esempio: MissItalia (2)

```
int main()
{
   int i;
   MissItalia ragazza1;
   MissItalia ragazza2;
   MissItalia gruppo[200];

   strcpy(ragazza1.nome, "Maria Rosaria" );
   ragazza1.altezza = 180;    ragazza1.voto[0] = 90;
   ragazza1.voto[1] = 90;     ragazza1.voto[2] = 90;

   printf("Nome=%s, altezza=%d," , ragazza1.nome,
   ragazza1.altezza );
```


Esempio: MissItalia (3)

La riga:

```
printf("Nome=%s, altezza=%d," , ragazza1.nome,
   ragazza1.altezza );
```

Stamperà:

```
Nome=Maria Rosaria, altezza=180,
```

Esempio: MissItalia (4)

Aggiungendo le seguenti righe di codice:

```
printf("voto=");
for (i=0; i<3; i++)
{
            printf(" %d " ,ragazza1.voto[i]);
}
```

Stamperà:

```
Nome=Maria Rosaria, altezza=180, voto= 90 90 90
```

53

Esempio: MissItalia (5)

Attenzione alla copia delle informazioni !

```
ragazza2 = ragazza1;  // sto copiando tutti i dati ???

printf(" %s \n" ,ragazza2.nome);
strcpy(ragazza1.nome, "Cambio il nome a Maria Rosaria");
printf(" %s \n" ,ragazza1.nome);  // Cambiato!!
printf(" %s \n" ,ragazza2.nome);  // Rimasto !!
```

Stamperà:

```
Maria Rosaria
Cambio il nome a Maria Rosaria
Maria Rosaria
```

54