PASCAL

Vs

ANSI C

Un esempio per cominciare…

La somma di 3 numeri

PASCAL

```
PROGRAM SOMMA;
VAR A,B,C,d:REAL;
BEGIN
(*  Inizializzazione di A ,B,C *)
   READLN(C);
   A:=5.;
   B:=7.;
(* Calcolo della somma tra A,B e C *)
   d:=A+B+C;
(* Stampa del risultato *)
   WRITELN ('SOMMA=',d);
END.
```

C

```
#include <stdio.h>
main()
{ /* inzio */
   float a,b,C,d;
/*  Inizializzazione di a, b, c */
   scanf("%f" ,& C );
   a=5.;  b=7.;
/* Calcolo della somma tra a,b e C*/
    d=a+b+C;
   printf("SOMMA= %f \ n",d);   /*
  Stampa del risultato*/
} /* fine */
```


Inizio e fine di un programma

PASCAL

- **L'inizio** di un programma è determinato da **BEGIN**
- **La fine** di un programma è determinata da **END.**

C

- **L'inizio** di un programma è determinato dalla **prima parentesi {**
- **La fine** di un programma è determinata dall'**ultima parentesi }**

```
main( )
  {      /* inizio */
  ......
  }      /* fine */
```


Regole sintattiche

PASCAL

- Possibilità di scrivere più istruzioni sulla stessa linea;
- **Non** c'e' **differenza** tra lettere **minuscole** o **maiuscole**.

C

- Possibilità di scrivere più istruzioni sulla stessa linea;
- **C'è differenza** tra lettere minuscole e maiuscole.

I Commenti

PASCAL

```
(*   Inizializzazione di A, B, C, d *)
```

La linea di commento è racchiusa tra

(* … *) oppure { … }

C

```
/*    Inizializzazione di a, b, C  */
```

La stringa di caratteri di un commento è racchiusa tra

/*…… */

Dichiarazioni delle variabili

PASCAL

```
VAR A,B,C : REAL;
VAR I,J : INTEGER;
```

C

```
float a,b,C;
int i,j;
```


Operazioni di I/O

PASCAL

```
WRITELN('SOMMA=',C);
```

Le operazioni di I/O sono eseguite da **istruzioni**

`READ();, READLN();`

`WRITE();,WRITELN( );`

C

```
printf("SOMMA= %f \n",c);
```

Le operazioni di I/O sono eseguite da **funzioni di libreria**

`scanf( );`

`printf( );`

Assegnazioni

PASCAL

```
A:=5;
```

L'operazione di assegnazione si effettua attraverso l'operatore

:=

C

```
a=5;
```

L'operazione di assegnazione si effettua attraverso l'operatore

=

C : Utilizzo degli header file

Prima dell'istruzione `main()` e' presente una direttiva del tipo:

`#include <"file.h">`

file.h contiene:
- specifiche di funzioni;
- definizioni di strutture di dati;
- definizioni di parametri;
- ...

Per eseguire Operazioni di I/O ⇒ `#include<stdio.h>`

Per utilizzare Funzioni Matematiche ⇒ `#include<math.h>`

Gestione Stringhe ⇒ `#include<string.h>`

...

stdio.h

- *int scanf(const char *format , ….)* — *Lettura da tastiera;*
- *int fscanf(FILE *strean, const char *format, …)* — *lettura da file ;*
- *int printf(const char *format , ….)* — *Stampa a video;*
- *int fprintf(FILE *strean, const char *format, …)* — *Stampa su file;*
- *FILE *fopen (cont char *filename, cost char *mode)* — *Apertura file;*
- *FILE *fclose (FILE * stream)* — *Chiusura file;*
- …

math.h

Gli argomenti *x* ed *y* sono di tipo *double*, n è di tipo *int* e tutte le funzioni restituiscono un *double*.

- *double sin(x)* seno di *x*;
- *double cos(x)* coseno di *x*;
- *double tan(x)* tangente di *x*;
- *double exp(x)* funzione esponenziale e^x;
- *double log(x)* logaritmo naturale *ln(x)*, con *x>0*;
- *double log10(x)* logaritmo in base 10 *log(x)*, con *x>0*;
- *double pow(x,y)* potenza x^y ;
- *double sqrt(x)* radice quadrata di *x*, con *x>0*;
- …

C: operazioni a basso livello

1. Operazioni dirette sui bit
2. Accesso agli indirizzi delle variabili.

Problema

Verificare se un numero naturale e' PARI o DISPARI

PASCAL

```
PROGRAM  PARDIS;
VAR N,RESTO : INTEGER;
BEGIN
   WRITELN( 'inserisci numero ');
   READLN(N);
   RESTO:=N-(N DIV 2)*2;
   IF RESTO = 0 THEN
    WRITELN('NUMERO PARI')
   ELSE
    WRITELN('NUMERO DISPARI');
END.
```

1 addizione + 2 molt.

C

```
#include <stdio.h>
main()
{
 int  n,resto;
 printf("inserisci numero\ n");
 scanf(" %d",&n);
 resto= n & 1;
 if(resto==0) {
   printf("numero pari\n");}
 else {
  printf("numero dispari\n");}
}
```

1 confonto bit a bit.

Operatore &

Regole di composizione dell'operatore &

&	0	1
0	0	0
1	0	1

ESEMPIO 1 (a & 1)

$a=65_{10}=1000001_2 \Rightarrow$ a & 1 = 1000001_2 &

$1_{10}=0000001_2$ 0000001_2 =

0000001_2

a & 1 = 1

ESEMPIO 2 (a & b)

$a=40_{10}=101000_2 \Rightarrow$ a & b = 101000_2&

$b=34_{10}=011000_2$ 011000_2=

001000_2

a & b = $001000_2=8_{10}$

Qual è il vantaggio di eseguire un'operazione bit a bit ?

La somma di due numeri con mantissa costituita da n bit richiede:

$\approx 2n$ confronti

Se $T_{confronto}=\mu$ e' il tempo necessario per un confronto, allora:

$T_{addizione} \approx 2n\,\mu$

Il tempo per eseguire un prodotto e' maggiore di quello per un'addizione

$T_{prodotto} \geq T_{addizione}$

$T_{addizione} + 2\ T_{prodotto} > nT_{confronto} = n\,\mu$

In particolare …

Nell'esempio, n & 1 equivale al resto della divisione di N per 2

PASCAL

```
R:=N-(N/2)*2;
```

C

Il resto della divisione per 2 e' il bit meno significativo 0 se il numero e' pari 1 se e' dispari

Es. $n=12_{10}=1100_2 \Rightarrow r = n \,\&\, 1 = 1100_2 \,\&$

$0001_2=$

0000_2

Es. $n=13_{10}=1101_2 \Rightarrow r = n \,\&\, 1 = 1101_2 \,\&$

$0001_2=$

0001_2

r = n & 1 ; & ⇒ AND bit a bit

ALTRE OPERAZIONI bit a bit

Divisione di un numero intero N per 2

PASCAL

```
N := N/2;
```

C

dividere per due significa spostare di un posto verso destra la rappresentazione binaria

Es. $n=13_{10}=1101_2 \Rightarrow n/2=6_{10}=0110_2$

n>>1 ; >> ⇒ SHIFT verso destra

Moltiplicazione di un numero intero N per 2

PASCAL

```
R := N * 2;
```

C

Moltiplicare per 2 significa spostare di 1 posto verso sinistra la rappresentazione binaria

Es. $n=13_{10}=1101_2 \Rightarrow n * 2 = 1101_2 * 0010_2 = 11010_2 = 26_{10}$

inv << 1; << ⇒ SHIFT verso sinistra

C: Operatori sui bit

&	and
>>	shift a destra
<<	shift a sinistra
|	or inclusivo
^	or esclusivo
~	complemento a 1

si utilizzano solo con operandi di tipo intero e carattere

OSSERVAZIONE

PASCAL

```
RESTO:= N- (N DIV 2) * 2
```

- Bisogna utilizzare l'operatore DIV per effettuare la divisione tra interi
- l'operatore / si utilizza per effettuare la divisione reali

C

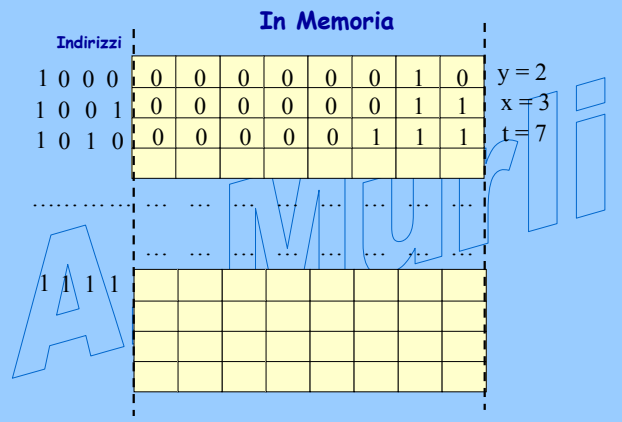
```
/*    Inizializzazione di a, b, C  */
```

- l'operatore / si utilizza per effettuare la divisione di variabili di QUALSIASI tipo

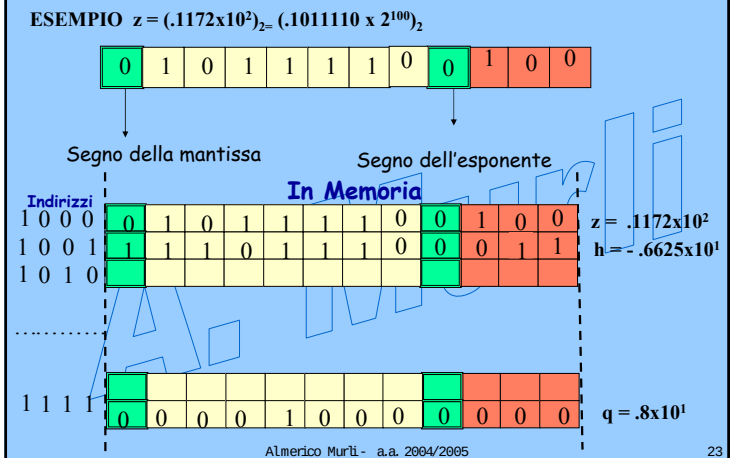
(⇒ Operatori sui bit)

(⇒ Puntatori)

Memorizzazione di un dato di tipo intero

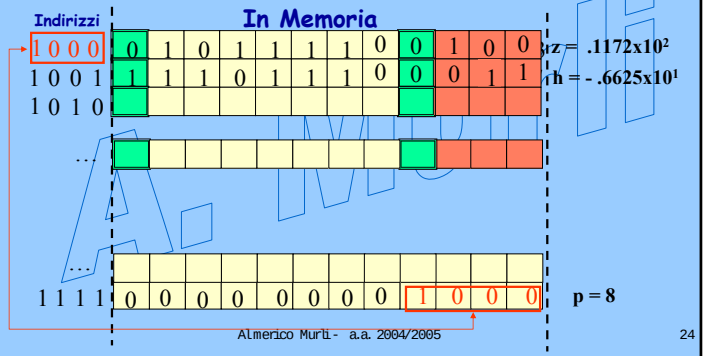


Memorizzazione di un dato di tipo reale



I PUNTATORI DEFINIZIONE

Il puntatore e' una variabile che contiene l'indirizzo di un'altra variabile



ESEMPI…

x = 3 variabile di tipo intero

x	0	0	0	0	0	0	1	1

Indirizzo: 1 0 1 0

Px contiene l'indirizzo di x

Px	0	0	0	0	1	0	1	0

z variabile di tipo floating-point

z	0	1	0	1	1	1	1	0	1	0	0

Indirizzo: 1 1 1 1

Pz contiene l'indirizzo di z

Pz	0	0	0	0	1	1	1	1

COME SI DICHIARA UN PUNTATORE?

PASCAL

```
type pointer =^integer;
var p : pointer;
```

p e' un puntatore ad una variabile intera.

```
type pointer =^real;
var p : pointer;
```

p e' un puntatore ad una variabile reale.

C

```
int *p
```

p e' un puntatore ad una variabile intera.

```
float *p
```

p e' un puntatore ad una variabile float.

In Pascal e' possibile accedere alle variabili attraverso il loro indirizzo.

In C e' possibile accedere alle variabili attraverso il loro indirizzo e conoscere l'indirizzo stesso.

COME SI DEFINISCE UN PUNTATORE in C ?

L'istruzione:

```
p = &x;
```

assegna l'indirizzo di x alla variabile puntatore P

(P punta a x)

In memoria:

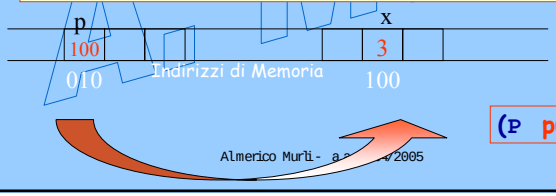
001	0	0	0	0	1	1	0	0	1	1	X
004								0	0	1	P

In particolare….

Differenza tra gli operatori & e *

Esempio

```
...
int x ,*p;
p=&x;     /*Assegnazione dell'indirizzo di x a p */
*p=3;     /* L'istruzione assegna il valore 3 alla
               variabile x puntata da p */
...
```



(p punta a x)

⇒ **Operatori sui bit**

⇒ **Puntatori**

⇒ **Allocazione dinamica**

Utilizzo degli array

PASCAL

```
PROGRAM  XXXX;
BEGIN
type vet = array[1..n] of integer;
     puntvet = ^vet;
var a : puntvet;
var n,x,i : integer;
          …
READLN(N); READLN(x);
new(a);
          for i:=1 to n do
          a(i)= x *i;
          ...
despose(a)
END.
```

C

```
#include <stdio.h>
main()
{ int *a,n,x,i;
  scanf("%d",&n);
  scanf("%d",&x);
  a=(int*)malloc(n*sizeof(int));
  …
  for (i=0; i<n; i++)
  { a[i]= x * i; }
 ...
free(a);
}
```

Se n = 4

Utilizzo solo della memoria necessaria

le dimensioni di un array sono stabilite *in runtime*

(ALLOCAZIONE DINAMICA)

In PASCAL ed in C

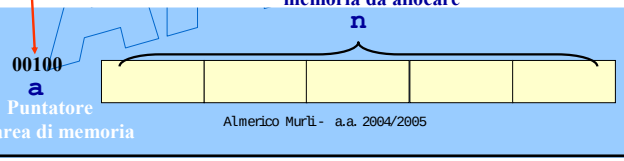
è possibile

allocare dinamicamente la memoria

Allocazione dinamica in PASCAL (cont.)

PASCAL

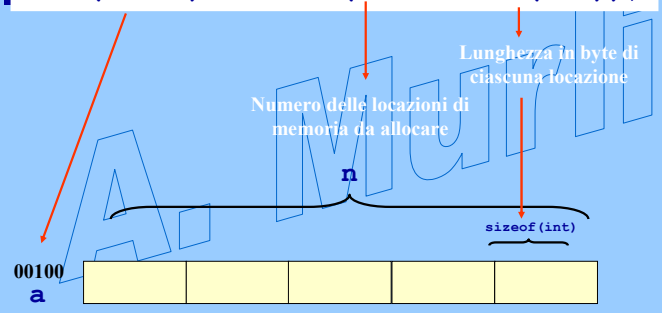
```
type vet = array[1..n] of integer;
     puntvet = ^vet;
var a : puntvet;
...
readln(n);
new(a);
...
```



Allocazione dinamica in C (cont.)

C

```
int *a;
a = (int *) malloc (n * sizeof(int));
```



Per deallocare un puntatore

PASCAL

```
dispose(p);
```

La memoria puntata da `p` e' libera.

C

```
free(p)
```

La memoria puntata da `p` e' libera.

FINE LEZIONE