PUNTO DELLA SITUAZIONE

martedì 26 settembre 2017 11:07

→ STRUTTURA DI UN PROGRAMMA C

```
#include ....        ← DIRETTIVE PER IL PREPROCESSORE
Dichiarazioni        ← VARIABILI GLOBALI
main () {
   ⋮
}
int foo (int x, int y) {
   ⋮
}
```

FUNZIONI E PROCEDURE

SEQUENZE DI COMANDI

- ASSEGNAMENTI (MODIFICA DELLO STATO)
- CONTROLLO DI FLUSSO (CONDIZIONALE / ITERATIVO)
- BLOCCHI (RAGGRUPPANO DICHIARAZIONI E SEQ. DI COMANDI IN UN SINGOLO COMANDO)

ALL'INTERNO DEI COMANDI ABBIAMO ESPRESSIONI (LEGGONO LO STATO)

VISIBILITÀ DELLE VARIABILI (STATO LOCALE)

→ LA CHIAMATA DI UNA PROCEDURA / FUNZIONE SOSPENDE LA PROCEDURA / FUNZIONE CORRENTEMENTE IN ESECUZIONE.

QUANDO LA PROCEDURA / FUNZIONE CHIAMATA TERMINA, LA PROCEDURA / FUNZIONE CHIAMANTE RIPRENDE LA SUA ESECUZIONE

⟶

PRINCIPIO DI MODULARITÀ

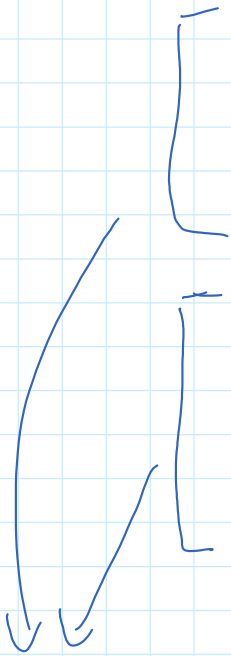
Un frammento di programma che ha senso di per sé è bene che venga implementato come una funzione o procedura

(UTILE PER RENDERE IL PROGRAMMA PIÙ LEGGIBILE E QUANDO IL FRAMMENTO È RIPETUTO PIÙ VOLTE)

↳ ESEMPIO "COMPLETO" : MCD di 3 NUMERI

```
#include <stdio.h>
int x;
int y;
int z;

main() {
    scanf("%d", &x);      // legge x
    scanf("%d", &y);      //  "  y
    scanf("%d", &z);      //  "  z
    while (x!=y) {
        if (x>y) x=x-y;
        else y=y-x;
    }
    while (x!=z) {
        if (x>z) x=x-z;
        else z=z-x;
    }
    printf("%d", x);      // stampo x sullo schermo
}
```



QUESTA PORZIONE DI PROGRAMMA HA SENSO DI PER SÉ ED È RIPETUTA

↳ FACCIAMONE UNA FUNZIONE

```
int x;
int y;
int z;

main () {
    int mcd1; int mcd2;
    scanf (....);
    scanf (....);
    scanf (....);
    mcd1 = mcd(x, y);
    mcd2 = mcd(mcd1, z);
    printf("%d", mcd2);
}

int mcd (int a, int b) {
    while (a != b) {
        if (a > b) a = a - b;
        else b = b - a;
    }
    return a;
}
```

5

VEDIAMO UN MODO DIVERSO DI IMPLEMENTARE LA FUNZIONE mcd

→ USIAMO UNA FUNZIONE AUSILIARIA CHE SCAMBIA I VALORI DI a E b

```
int mcd (int a, int b) {
    while (a!=b) {
        if (b > a) scambia (a, b);
        a = a - b;
    }
    return a;
}
```

VOGLIO CHE IL PIÙ GRANDE SIA a

NON C'È ELSE ...

```
void scambia (int p, int q) {
    p = q;
    q = p;
}
```

NO!!

```
void scambia (int p, int q) {
    int tmp;
    tmp = p;
    p = q;
    q = tmp;
}
```

VARIABILE DI APPOGGIO (TEMPORANEA)

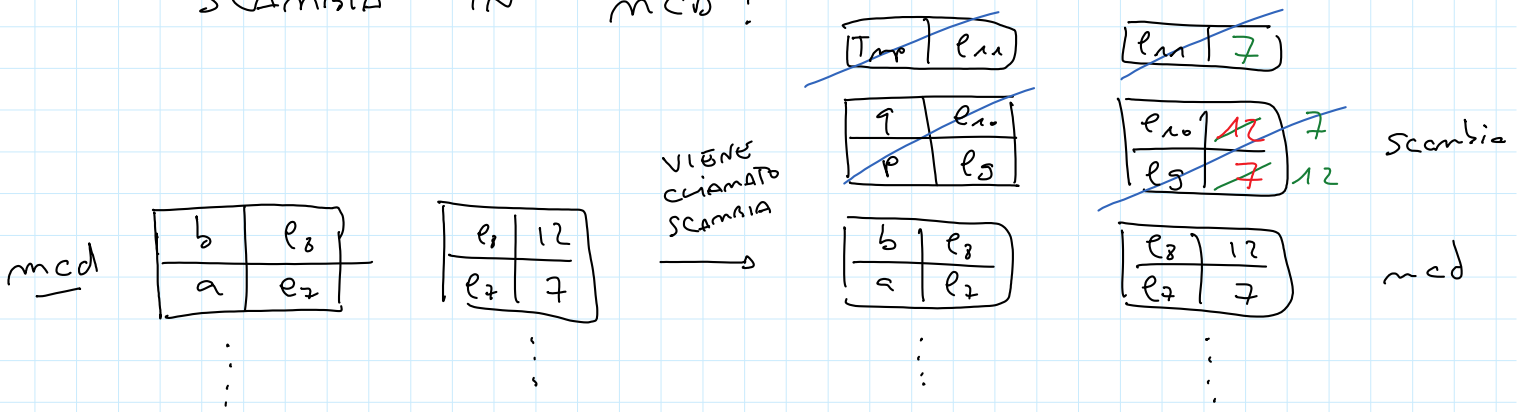
OK

q	e_2
p	e_1

e_2	~~7~~	7
e_1	~~4~~	7

HO PERSO IL VALORE DI p DOPO IL PRIMO ASSEGNAMENTO

COSA SUCCEDE QUANDO CHIAMO SCAMBIA IN mcd?



AL TERMINE DELLA FUNZIONE SCAMBIA I FRAME ATTIVATI SPARISCONO SENZA AVER MODIFICATO a E b (DOVUTO AL PASSAGGIO DEI PARAMETRI PER VALORE)

martedì 26 settembre 2017 11:56

PUNTATORI

UN PUNTATORE È UNA VARIABILE IL CUI VALORE È UN INDIRIZZO DI MEMORIA (UNA LOCAZIONE)

→ POSSIAMO USARE UN PUNTATORE PER MEMORIZZARE LA LOCAZIONE DI UN'ALTRA VARIABILE

DICHIARAZIONE DI UN PUNTATORE

```
int *p;
```

DICHIARA UNA VARIABILE P CHE' PUNTATORE A VARIABILI DI TIPO INT

```
{ int x = 10;
  int *p;
  p = &x;
  x = x + 1;
  p = x + 1
```

p	ℓ_1
x	ℓ_0

amb

ℓ_1	ℓ_0
ℓ_0	~~10~~

mem

&x RAPPRESENTA LA LOCAZIONE ASSOCIATA ALLA VARIABILE x NELL'AMBIENTE

$\&x \equiv \ell_0$

NO! L'ESPRESSIONE x+1 HA TIPO int, NON int *

COME POSSO USARE UN PUNTATORE?

→ SE P VOGLIO MODIFICARE IL VALORE DELLA VARIABILE "PUNTATA" DA P (OSSIA X)

DEREFERENZIAZIONE DEL PUNTATORE (ACCEDE ALLA VARIABILE PUNTATA)

← STA PER LA VARIABILE (DI TIPO INT) IL CUI INDIRIZZO È QUELLO MEMORIZZATO DA P

```
{ int x=10;
  int y=20;
  int *p = &x;
  int *q = &y;
```

q	e_3
p	e_2
y	e_1
x	e_0

AMB

e_3	e_1
e_2	~~e_0~~
e_1	20
e_0	~~10~~ 21

e_1

MEM

```
  x = y+1;
  *p = *q+1;
```

← QUESTE DUE OPERAZIONI SONO EQUIVALENTI

```
  p = &y;
  *p = *q+1;
  y = y+1;
}
```

← QUESTA VOLTA p RIFERISCE A y

QUESTE DUE OPERAZIONI SONO EQUIVALENTI NEL NUOVO STATO (DOPO CHE HO RIASSEGNATO p)

NOTA `printf("%d", p)` ← STAMPEREBBE IL VALORE DEL PUNTATORE (L'INDIRIZZO DI MEMORIA)

`printf("%d", *p)` ← STAMPA IL VALORE (INTERO) DELLA VARIABILE PUNTATA DA p

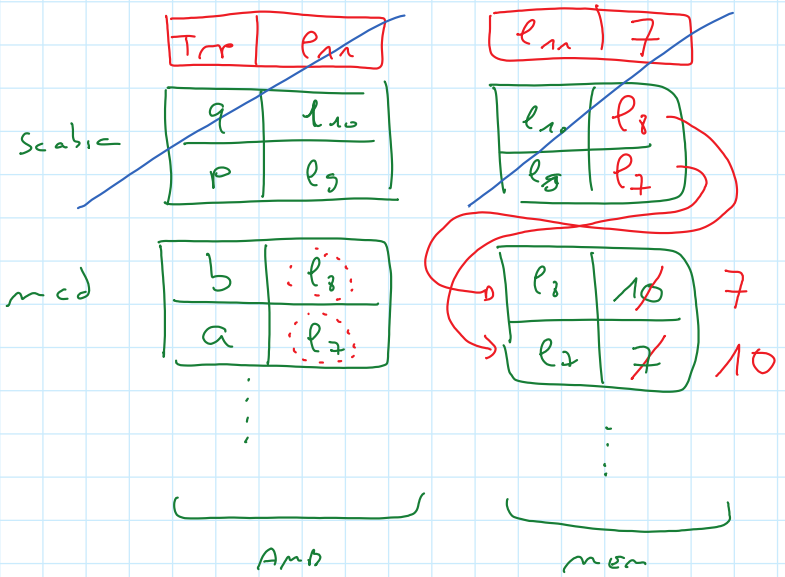
PASSAGGIO DI PARAMETRI PER INDIRIZZO (PER VARIABILE)

→ PASSO A UNA FUNZIONE / PROCEDURA NON IL VALORE DI UNA VARIABILE MA IL SUO INDIRIZZO IN MODO CHE POSSA MODIFICARLA

```
int mcd(int a, int b) {
    while (a != b) {
        if (b > a) scambia(&a, &b);
        a = a - b;
    }
}

void scambia(int *p, int *q) {
    int Tmp;
    Tmp = *p;
    *p = *q;
    *q = Tmp;
}
```

Scambia
"
Swap



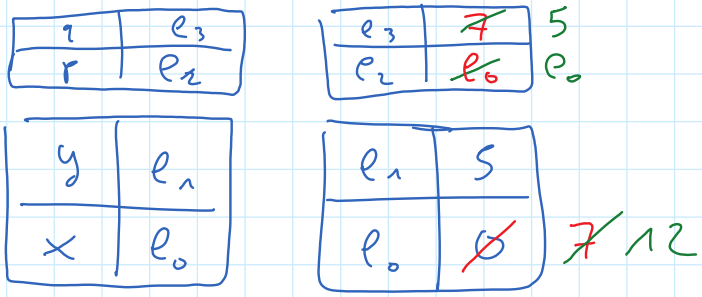
ALTRI ESEMPI

↳ PROCEDURA INCREMENTA UNA VARIABILE DI N

```
main () {
    int x = 0;
    int y = 5;
    incr (&x, 7);
    incr (&x, y);
}

void incr (int *p, int q) {
    *p = *p + q;
}
```

PASSO L'INDIRIZZO DI X PERCHÉ X DEVE ESSERE MODIFICATO.
Y PUÒ ESSERE PASSATO PER VALORE



⇝ FUNZIONE CHE DEVE RESTITUIRE PIU' DI UN RISULTATO

```
int sommaprodotto (int x, int y)
```

CALCOLA x+y E x*y RESTITUENDO ENTRAMBI I RISULTATI AL CHIAMANTE

```
int sommaprodotto (int x, int y, int *s) {
    *s = x+y;        ← SOMMA
    return x*y;      ← PRODOTTO
}
```

COME FARE? MI FACCIO PASSARE IL PUNTATORE A UNA VARIABILE IN CUI SCRIVERE UNO DEI DUE

CHIAMATA:

```
int somma;
int prodotto;
prodotto = sommaprodotto(10, 12, &somma);
```

PASSO ALLA FUNZIONE L'INDIRIZZO DELLA VARIABILE IN CUI SCRIVERE LA SOMMA