Laboratorio II

Corso A

Lezione 7

- Classi di memorizzazione e scoping
- Gestione della memoria e puntatori
- Puntatori e funzioni: passaggio per riferimento di parametri

Classi di memorizzazione per variabili, scoping

- **Variabili locali** (interne) ad un blocco/funzione - visibili fino alla fine del blocco/funzione
  - Allocate all'entrata di un blocco, deallocate al suo termine sullo stack
  - Classe di memorizzazione `auto` (implicita)
- **Variabili globali** (esterne) - definite fuori dalle funzioni - visibili fino alla fine del file corrente e in tutti i file che lo includono
  - Tutte le variabili globali sono dichiarate `extern` (implicita)
  - Funzioni sono sempre globali - non possiamo definire funzioni dentro ad altre funzioni
- **Variabili `static`**
  - Possiamo impedire ad altri file di utilizzare variabili (anche funzioni) globali, utilizzando `static`
  - Possono esistere variabili `static` locali a funzioni - persistenti tra una chiamata della funzione ad un'altra
  - Allocate sul segmento di memora dei **dati globali**

File header `.h`

Per modularità e riutilizzo del codice può essere utile dividere il programma in vari file (librerie)

La dichiarazione delle funzioni e costanti in un file `.h` da includere (`#include`) negli altri file che utilizzano le funzioni

La definizione delle funzioni in un file `.c` - può essere compilato separatamente

max.c

```c
int max(int a, int b){
  if (a>b)
    return a;
  return b;
}
```

max.h

```c
int max(int, int);

```

main.c

```c
#include <stdio.h>
#include "max.h"

int main(void) {
  int a=5, b=10;
  printf("%d\n", max(a,b));
}

```

Esempio

max.h

```c
int max(int, int);
```

main.c

```c
#include <stdio.h>
#include "max.h"

int main(void) {

  int a=0,b=0;

  do{
    int a=0;
    scanf("%d %d",&a,&b);
    printf("Max(%d,%d)=%d\n",a,b,max(a,b));
  } while(a!=b);

  printf("Hai inserito a>b %d volte\n",countA);

}
```

max.c

```c
int countA=0;
int countB=0;

int max(int a, int b){
  if (a>b){
    countA++;
    return a;
  }
  countB++;
  return b;
}
```

Esempio

max.h

```
1   int max(int, int);
```

main.c

```
1   #include <std
2   #include "max
3
4   int main(void
5
6     int a=0,b=0
7
8     do{
9       int a=0;
10      scanf("%d
11      printf("M
12    } while(a!=
13
14    printf("Hai
15
16  }
```

```
main.c:14:40: error: use of undeclared
identifier
        'countA'
  printf("Hai inserito a>b %d volte\n",
countA);

^
1 error generated.
exit status 1
```

max.c

```
                                    int b){
11  }
```

Esempio

max.h

```c
int max(int, int);
```

main.c

```c
#include <stdio.h>
#include "max.h"

extern int countA;

int main(void) {

  int a=0,b=0;

  do{
    int a=0;
    scanf("%d %d",&a,&b);
    printf("Max(%d,%d)=%d\n",a,b,max(a,b));
  } while(a!=b);

  printf("Hai inserito a>b %d volte\n",countA);

}
```

Quando ci fermiamo?

max.c

```c
int countA=0;
int countB=0;

int max(int a, int b){
  if (a>b){
    countA++;
    return a;
  }
  countB++;
  return b;
}
```

Esempio

main.c

```c
#include <stdio.h>
#include "max.h"

extern int countA;

int main(void) {

  int a=0,b=0;

  do{
    int a=0;
    scanf("%d %d",&a,&b);
    printf("Max(%d,%d)=%d\n",a,
  } while(a!=b);

  printf("Hai inserito a>b %d v

}
```

```
> ./main
4 7
Max(4,7)=7
7 2
Max(7,2)=7
8 1
Max(8,1)=8
6 8
Max(6,8)=8
3
3
Max(3,3)=3
0
0
Max(0,0)=0
Hai inserito a>b 2 volte
```

max.h

```c
int max(int, int);
```

```c
int countA=0;
int countB=0;

int max(int a, int b){
  if (a>b){
    countA++;
    return a;
  }
  countB++;
  return b;
}
```

Esempio

max.h

```c
int max(int, int);
```

main.c

```c
#include <stdio.h>
#include "max.h"

extern int countA;

int main(void) {

  int a=0,b=0;

  do{
    int a=0;
    scanf("%d %d",&a,&b);
    printf("Max(%d,%d)=%d\n",a,b,max(a,b));
  } while(a!=b);

  printf("Hai inserito a>b %d volte\n",countA);

}
```

max.c

```c
static int countA=0;
static int countB=0;

int max(int a, int b){
  if (a>b){
    countA++;
    return a;
  }
  countB++;
  return b;
}
```

Esempio

max.h

```
1   int max(int, int);
```

main.c

```
1   #includ
2   #includ
3
4   extern
5
6   int mai
7
8     int a
9
10    do{
11      int
12      sca
13      pri
14    } whi
15
16    printf("Hai inserito a>b %d volte\n",countA);
17
18  }
19
```

max.c

```
10    return b;
11  }
```

```
/tmp/main-df0f92.o: In function `main':
main.c:(.text+0x84): undefined referenc
e to `countA'
clang-7: error: linker command failed w
ith exit code 1 (use -v to see invocati
on)
exit status 1
```

Esempio

max.h

```c
int max(int, int);
```

main.c

```c
#include <stdio.h>
#include "max.h"


int main(void) {

  int a=0,b=0;

  do{
    int a=0;
    scanf("%d %d",&a,&b);
    printf("Max(%d,%d)=%d\n",a,b,max(a,b));
  } while(a!=b);

}
```

max.c

```c
#include <stdio.h>

int max(int a, int b){
  static int countA=0;
  static int countB=0;

  if (a>b){
    countA++;
    printf("countA=%d, countB=%d\n",countA,countB);
    return a;
  }
  countB++;
  printf("countA=%d, countB=%d\n",countA,countB);
  return b;
}
```

Esempio

max.h

```
int max(int, int);
```

main.c

```
#include <stdio.h>
#include "max.h"


int main(void) {

  int a=0,b=0;

  do{
    int a=0;
    scanf("%d %d",&a,&b);
    printf("Max(%d,%d)=%d\n",a,b,max
  } while(a!=b);

}
```

```
3 5
countA=0, countB=1
Max(3,5)=5
6 2
countA=1, countB=1
Max(6,2)=6
3 8
countA=1, countB=2
Max(3,8)=8
2 2
countA=1, countB=3
Max(2,2)=2
0 0
countA=1, countB=4
Max(0,0)=0
```

```
de <stdio.h>

x(int a, int b){
ic int countA=0;
ic int countB=0;

a>b){
untA++;
intf("countA=%d, countB=%d\n",countA,countB);
turn a;

tB++;
tf("countA=%d, countB=%d\n",countA,countB);
rn b;
```

Esempio

max.h

```c
int max(int, int);
```

main.c

```c
#include <stdio.h>
#include "max.h"

int main(void) {

  int a=0,b=0;

  do{
    int a=0;
    scanf("%d %d",&a,&b);
    printf("Max(%d,%d)=%d\n",a,b,max(a,b));
  } while(a!=b);

  printf("Hai inserito a>b %d volte\n",countA);

}
```

max.c

```c
int countA=0;
int countB=0;

int max(int a, int b){
  if (a>b){
    countA++;
    return a;
  }
  countB++;
  return b;
}
```

Esempio

max.h

```
1   int max(int, int);
```

main.c

```
1   #include <std
2   #include "max
3
4   int main(void
5
6     int a=0,b=0
7
8     do{
9       int a=0;
10      scanf("%d
11      printf("M
12    } while(a!=
13
14    printf("Hai
15
16  }
```

```
main.c:14:40: error: use of undeclared
identifier
        'countA'
  printf("Hai inserito a>b %d volte\n",
countA);

^
1 error generated.
exit status 1
```

max.c

```
                                    int b){
11  }
```

Esempio

max.h

```c
int max(int, int);
```

main.c

```c
#include <stdio.h>
#include "max.h"

extern int countA;

int main(void) {

  int a=0,b=0;

  do{
    int a=0;
    scanf("%d %d",&a,&b);
    printf("Max(%d,%d)=%d\n",a,b,max(a,b));
  } while(a!=b);

  printf("Hai inserito a>b %d volte\n",countA);

}
```

Quando ci fermiamo?

max.c

```c
int countA=0;
int countB=0;

int max(int a, int b){
  if (a>b){
    countA++;
    return a;
  }
  countB++;
  return b;
}
```

Esempio

main.c

```c
#include <stdio.h>
#include "max.h"

extern int countA;

int main(void) {

  int a=0,b=0;

  do{
    int a=0;
    scanf("%d %d",&a,&b);
    printf("Max(%d,%d)=%d\n",a,
  } while(a!=b);

  printf("Hai inserito a>b %d v

}
```

```
> ./main
4 7
Max(4,7)=7
7 2
Max(7,2)=7
8 1
Max(8,1)=8
6 8
Max(6,8)=8
3
3
Max(3,3)=3
0
0
Max(0,0)=0
Hai inserito a>b 2 volte
```

max.h

```c
int max(int, int);
```

```c
int countA=0;
int countB=0;

int max(int a, int b){
  if (a>b){
    countA++;
    return a;
  }
  countB++;
  return b;
}
```

Esempio

max.h

```c
int max(int, int);
```

main.c

```c
#include <stdio.h>
#include "max.h"

extern int countA;

int main(void) {

  int a=0,b=0;

  do{
    int a=0;
    scanf("%d %d",&a,&b);
    printf("Max(%d,%d)=%d\n",a,b,max(a,b));
  } while(a!=b);

  printf("Hai inserito a>b %d volte\n",countA);

}
```

max.c

```c
static int countA=0;
static int countB=0;

int max(int a, int b){
  if (a>b){
    countA++;
    return a;
  }
  countB++;
  return b;
}
```

Esempio

max.h

```
1   int max(int, int);
```

main.c

```
1   #includ
2   #includ
3
4   extern
5
6   int mai
7
8     int a
9
10    do{
11      int
12      sca
13      pri
14    } whi
15
16    printf("Hai inserito a>b %d volte\n",countA);
17
18  }
19
```

max.c

```
10    return b;
11  }
```

```
/tmp/main-df0f92.o: In function `main':
main.c:(.text+0x84): undefined referenc
e to `countA'
clang-7: error: linker command failed w
ith exit code 1 (use -v to see invocati
on)
exit status 1
```

Esempio

max.h

```c
int max(int, int);
```

main.c

```c
#include <stdio.h>
#include "max.h"


int main(void) {

  int a=0,b=0;

  do{
    int a=0;
    scanf("%d %d",&a,&b);
    printf("Max(%d,%d)=%d\n",a,b,max(a,b));
  } while(a!=b);

}
```

max.c

```c
#include <stdio.h>

int max(int a, int b){
  static int countA=0;
  static int countB=0;

  if (a>b){
    countA++;
    printf("countA=%d, countB=%d\n",countA,countB);
    return a;
  }
  countB++;
  printf("countA=%d, countB=%d\n",countA,countB);
  return b;
}
```

Esempio

max.h

```
int max(int, int);
```

main.c

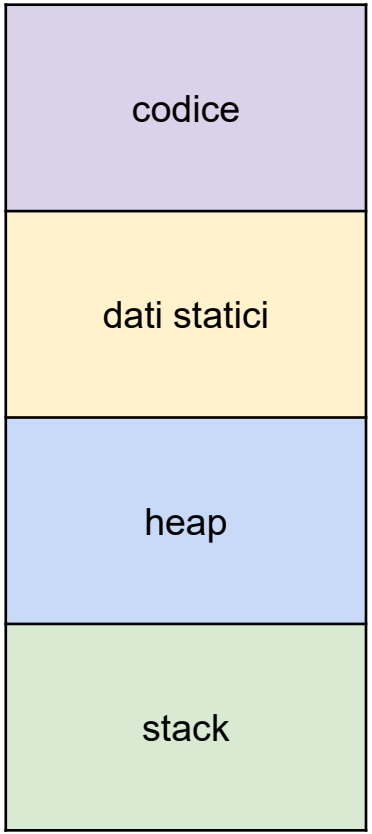
```
#include <stdio.h>
#include "max.h"


int main(void) {

  int a=0,b=0;

  do{
    int a=0;
    scanf("%d %d",&a,&b);
    printf("Max(%d,%d)=%d\n",a,b,max
  } while(a!=b);

}
```

```
3 5
countA=0, countB=1
Max(3,5)=5
6 2
countA=1, countB=1
Max(6,2)=6
3 8
countA=1, countB=2
Max(3,8)=8
2 2
countA=1, countB=3
Max(2,2)=2
0 0
countA=1, countB=4
Max(0,0)=0
```

```
de <stdio.h>

x(int a, int b){
ic int countA=0;
ic int countB=0;

a>b){
untA++;
intf("countA=%d, countB=%d\n",countA,countB);
turn a;

tB++;
tf("countA=%d, countB=%d\n",countA,countB);
rn b;
```

Gestione della memoria

- Programma e dati - memoria
- Zona dati
  - Dati statici - possono essere inferiti a tempo di compilazione
    - Variabili globali, variabili locali statiche
  - Stack - record di attivazione
    - 1 per ogni blocco di codice (incluso chiamate di funzioni)
    - spazio per variabili locali
  - Heap - dati dinamici (runtime), frammentati
    - Memoria allocata dinamicamente - tramite puntatori



Puntatori

- Qualsiasi dato/istruzione che sta in memoria ha **un valore** e un **indirizzo di memoria**
  - Un indirizzo in memoria è un numero intero dipendente dalla macchina
- Un puntatore è **una variabile** che contiene un indirizzo di memoria

  `int *a;`
  - Ha a sua volta un indirizzo di memoria
  - Il valore è un indirizzo di memoria - ci indica l'inizio di uno spazio di memoria che può essere nello stack, heap, anche istruzioni (codice)
  - Il tipo ***indica*** il tipo di dato che mi aspetto di vedere a quell'indirizzo di memoria.
  - Possono essere inizializzati all'indirizzo di un'altra variabile o con il valore `NULL`, che indica nessun indirizzo

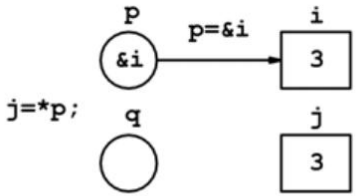
```
int main(void) {

  int a = 5;

  int *aPtr = &a;

}
```

valore: 5
indirizzo: 13

valore: 13
indirizzo: 36

0	1	2	3	4	5	6	7	8	9	10
11	12	13	14 **5**	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37 **13**	38	39	40	41	42	43
44	45	46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63	64	65

Puntatori



- **Nonostante siano numeri interi, il C non li considera normali interi!**
- Si può usare il puntatore direttamente in espressioni tra puntatori, ma non con altri tipi interi
  - **Aritmetica dei puntatori (futura lezione)**
- Possiamo leggere e usare l'indirizzo di memoria di una variabile attraverso l'**operatore &**
- **Dereferenziazione**
  - Possiamo leggere/scrivere il valore memorizzato ad un certo indirizzo di memoria atrraverso un puntatore usando l'**operatore ***

```
int b=1;
int c=22;
int *a = &c;
*a = b + *a; /* Viene cambiato il valore contenuto in c */
a = b + *a; /* NO! A è un puntatore, non un intero! */
*a = b + a; /* NO! A è un puntatore, non si può sommare ad un intero! */
int *a = c; /* NO! A è un puntatore, non si può assegnare un intero! */
```

  - Dereferenziare un puntatore con valore **`NULL`** o che punta a memoria non allocata porta ad un **errore a tempo di esecuzione**!

Puntatori - esempio

```c
#include <stdio.h>

int main(void) {

  int a=0, b=0;

  int *aPtr=&a;

  printf("Il valore di a è %d, l'indirizzo di a è %p\n", a, &a);
  printf("Il valore di aPtr è %p, l'indirizzo di aPtr è %p\n", aPtr, &aPtr);
}
```

```
Il valore di a è 0, l'indirizzo di a è 0x7ffc743dbd1c
Il valore di aPtr è 0x7ffc743dbd1c, l'indirizzo di aPtr è 0x7ffc743dbd10
```

Puntatori

```c
#include <stdio.h>

int main(void) {

  double a=0;
  double *aPtr=&a, *aPtr1=&a;

  printf("Il valore di a è %f, l'indirizzo di a è %p\n", a, &a);

  (*aPtr)=10;
  printf("Il valore di a è %f, l'indirizzo di a è %p\n", a, &a);

  (*aPtr1)=20;
  printf("Il valore di a è %f, l'indirizzo di a è %p\n", a, &a);
}
```

```
Il valore di a è 0.000000, l'indirizzo di a è 0x7ffdb8ce4fe8
Il valore di a è 10.000000, l'indirizzo di a è 0x7ffdb8ce4fe8
Il valore di a è 20.000000, l'indirizzo di a è 0x7ffdb8ce4fe8
```

Puntatori

```c
#include <stdio.h>

int main(void) {

  int a=15;
  double *aPtr=&a;

  printf("Il valore di a è %d, l'indirizzo di a è %p\n", a, &a);

  printf("Il valore di aPtr è %p\n", aPtr);

  printf("Il valore memorizzato a %p è %f", aPtr, *aPtr);
}
```

```
Il valore di a è 15, l'indirizzo di a è 0x7ffcf87e0a3c
Il valore di aPtr è 0x7ffcf87e0a3c
Il valore memorizzato a 0x7ffcf87e0a3c è 0.000000
```

Puntatori

```c
#include <stdio.h>

int main(void) {

  int a=15;
  double *aPtr=&a;
```

```
main.c:6:11: warning: incompatible pointer types initializing 'double *' with an
      expression of type 'int *' [-Wincompatible-pointer-types]
  double *aPtr=&a;
          ^    ~~
1 warning generated.
```

```c
a, &a);

;

}
```

```
Il valore di a è 15, l'indirizzo di a è 0x7ffcf87e0a3c
Il valore di aPtr è 0x7ffcf87e0a3c
Il valore memorizzato a 0x7ffcf87e0a3c è 0.000000
```

Puntatori

```c
#include <stdio.h>

int main(void) {

  double a=15.6;
  int *aPtr=&a;

  printf("Il valore di a è %f, l'indirizzo di a è %p\n", a, &a);

  printf("Il valore di aPtr è %p\n", aPtr);

  printf("Il valore memorizzato a %p è %d", aPtr, *aPtr);
}
```

```
Il valore di a è 15.600000, l'indirizzo di a è 0x7ffee72ceed8
Il valore di aPtr è 0x7ffee72ceed8
Il valore memorizzato a 0x7ffee72ceed8 è 858993459
```

Passaggio parametri a funzioni: chiamata per indirizzo

- I parametri vengono allocati sullo stack della funzione chiamata
- Per poter cambiare il valore di una variabile nell'ambiente del chiamante, dall'interno di una funzione, si possono utilizzare i puntatori
  - Passando l'indirizzo di memoria di una variabile, quella variabile può essere modificata

```
a=5,b=10
a=10,b=5
```

```c
#include <stdio.h>

void swap(int*, int* );

int main(void) {
  int a=5, b=10;

  printf("a=%d,b=%d\n", a,b);

  swap(&a,&b);

  printf("a=%d,b=%d\n", a,b);
}

void swap(int* x, int* y){
  int var=*x;
  *x=*y;
  *y=var;
}
```

Domande?