Differenze C – Java

E Mumolo

Maggiori differenze

JAVA	C
Orientato agli oggetti	Procedurale
Interprete	Compilatore
Gestione della memoria	Il programmatore deve gestire la memoria
Riferimenti	Puntatori
Eccezioni	Codici errore

Java è una collezione di oggetti

- Esempio: scambio di due valori in Java

costruttore

```
class myvar{
    private int x;
    public myvar(int a) {x = a;}
    public int get() { return x; }
    public void set(int a) {x = a;}
}
```

```
public class Swap {
   static void swap(myvar a, myvar b)
   {
      // scambio
      int t = a.get();
      a.set(b.get());
      b.set(t);
   }

   public static void main(String[] args)
   {
     int x = 23,y = 47;
      myvar a = new myvar(x);
      myvar b = new myvar(y);
      swap(a, b);
      System.out.println(" x:" + a.get + ", y: " + b.get);
   }
}
```

oggetti

C è una collezione di procedure

- Esempio: scambio di due valori in C

```
#include <stdio.h>
void swap(int *i, int *j) {
    int t = *i;
    *i = *j;
    *j = t;
}
void main() {
    int a = 23, b = 47;
    swap(&a, &b);
    printf("a: %d, b: %d\n", a, b);
}
```

```
int main() {
    int a = 23, b = 47;
    printf("Prima. a: %d, b: %d\n", a, b);
    {int t = a; a = b; b = t;}
    printf("Dopo.  a: %d, b: %d\n", a, b);
    return 0;
}
```

- Esempio: scambio di due valori in C usando il preprocessore

```
#define swap(type, i, j) {type t = i; i = j; j = t;}
int main() {
    int a = 23, b = 47;
    printf("Prima. a: %d, b: %d\n", a, b);
    swap(int, a, b)
    printf("Dopo.  a: %d, b: %d\n", a, b);
    return 0;
}
```

Interprete/Compilatore

- Programmi Java

multipiattaforma

File sorgente Java (*.java) → file bytecode (*.class) → linguaggio macchina (esecuzione)

javac *.java

java *.class

Interprete JVM: esegue uno per uno il bytecode trasformandolo in linguaggio macchina

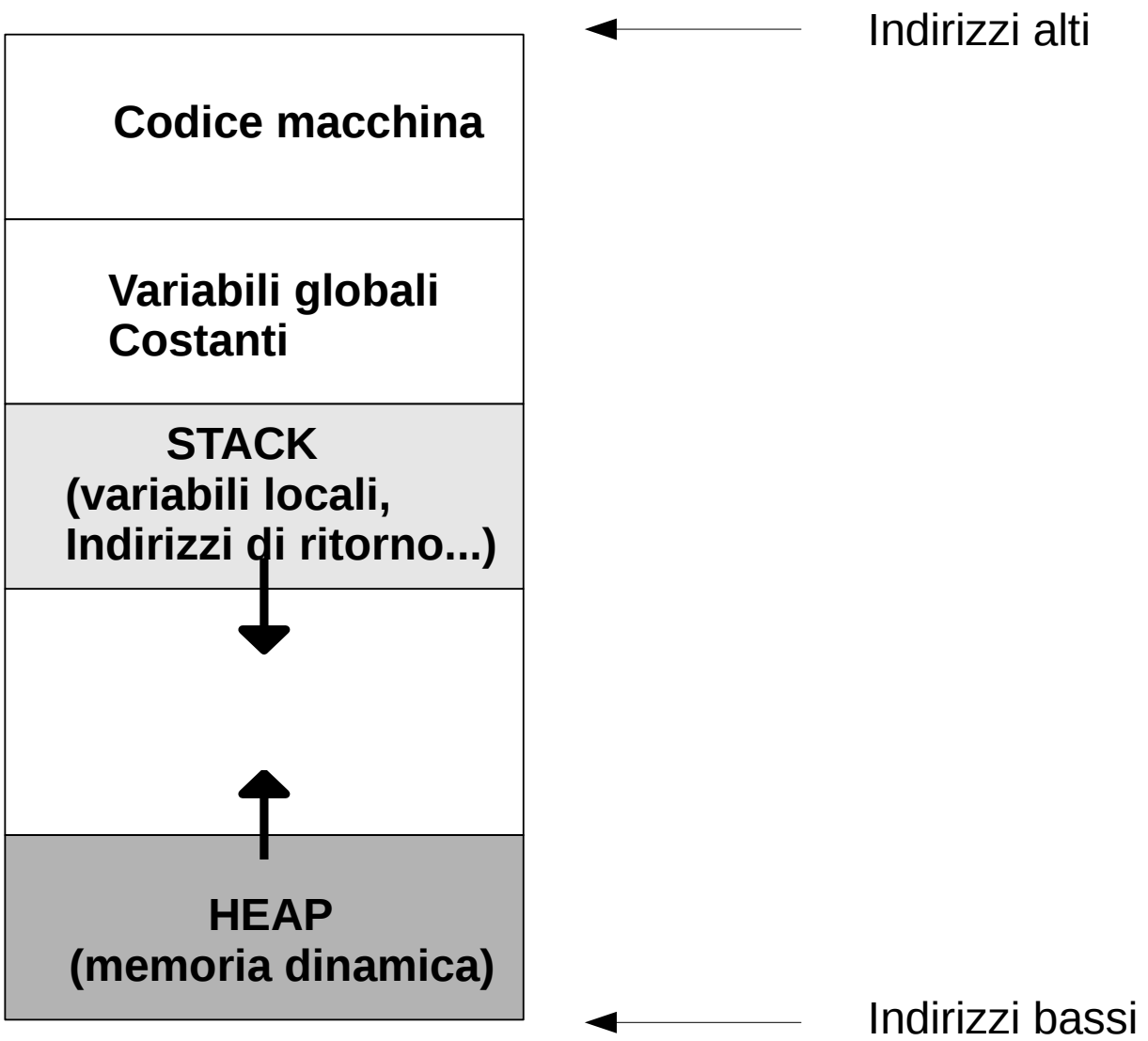
- Programmi C

File sorgente C (*.c) → linguaggio macchina (esecuzione)

gcc *.c

- Conclusione: Java interpretato/C compilato

Come il Sistema Operativo vede la memoria



Gestione memoria

- In Java, nuovi oggetti sono creati con 'new'
  - Gli oggetti allocano memoria in Heap
  - Quando l'oggetto non è più usato, viene automaticamente rimosso dalla memoria
- In C, blocchi di memoria vengono allocati/cancellati in Heap usando le procedure:
  - sizeof, malloc, free
- Il programmatore deve ricordarsi di cancellare la memoria allocata
- Allocazione statica in C (int a; float b;...)
  - O in memoria globale o in stack

C vs Java

- Riferimenti/puntatori
  - Un riferimento in Java è l'indirizzo di un oggetto
  - Un puntatore in C è l'indirizzo di una variabile o struttura dati
- Eccezioni/codici d'errore
  - Java può catturare gli errori run time (eccezioni)
  - C non ha eccezioni. Un programma o va in crash o ritorna un codice d'errore

Esempio uso codici errore

```
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>

main()
{

    int fd; //file descriptor
    fd = open("/home/em/test",O_CREAT|O_EXCL);
    if ( fd < 0 ) {
         printf ("Errore in open. Errore n.: %d\n", errno);
    }
    else
         printf("Open OK\n");
    }
```

```
extern int errno;
extern char *program_invocation_name,
            *program_invocation_short_name;
typedef int error_t;
```

Open system call

Variabile globale errno

Tabella dei primi 10 codici d'errore

Error number	Error Code	Error Description
1	EPERM	Operation not permitted
2	ENOENT	No such file or directory
3	ESRCH	No such process
4	EINTR	Interrupted system call
5	EIO	I/O error
6	ENXIO	No such device or address
7	E2BIG	Argument list too long
8	ENOEXEC	Exec format error
9	EBADF	Bad file number
10	ECHILD	No child processes

Vari esempi uso codici errore

Funzione che ritorna un puntatore

```
...
...
char *ptr = NULL;
ptr = funzione();
if(ptr == NULL)
{
    /**
    Gestisci l'errore
     **/
}
...
...
...
```

Errore di puntatore

```
int funzione(int ErrorCode,char **ErrorText,
             int *Size)
{
// controllo dei parametri di ingresso
  if (ErrorText == NULL || Size == NULL)
  {
    return PARAMETRO_INVALIDO;
  }
...
...
...
}
```

Errore nei parametri di ingresso

Altri esempio uso codici errore inC

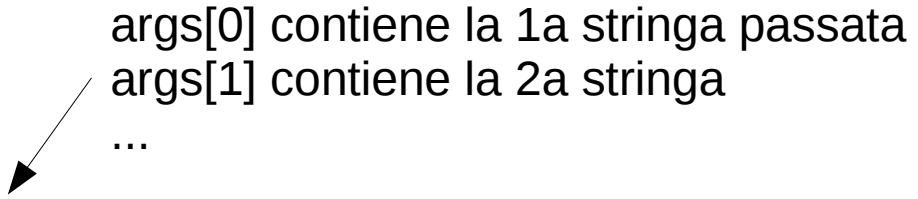
```
int FunzA(int x)
{
 if (x > Soglia)
    return ERROR;
 else
    return OK;
}
```

Ritorna o ERRORE o OK

```
int FunzB(int x)
{
  int err = FunzA(x);
  if (err != OK)
     return GestisciErrore(err);
  else
     return(0) ;
}
```

Se funzA ritorna ERRORE → gestisce
Se funzA ritorna OK → ritorna 0

Passaggio argomenti al main da linea di comandi in Java




```
import java.io.*;
public class argomenti {
    public static void main (String[] args) {
        int primo; String secondo;

        if (args.length > 0) {
              try {
               primo = Integer.parseInt(args[0]);
               System.out.println("1o argomento " + primo );
              } catch (NumberFormatException e) {
                  System.err.println("il 2o argomento "+args[0]+" deve
                                                          essere int");
                  System.exit(1);
               }
               Secondo=args[1];
               System.out.println("2o argomento " + secondo );
        }
    }
}
```

---

javac argomenti.java
java argomenti 5 aa

Passaggio argomenti al main da linea di comandi in C

```
#include <stdio.h>

int main(int argc, char* argv[])
{
     int i;
     int primo;
     char s[10];

     if(argc != 2)
     {
          printf(“inserire due argomenti\n”);
          exit(0);
     }
     for(i = 1; i < argc; i++)
     {
          printf(“Argomento %d è %s”, i, argv[i]);
     }
     primo=atoi(argv[1]);
     strcpy(s,argv[2]);
     printf(“primo argomento=%d, secondo=%s\n”, primo,s);

     return (0);
}
```

Numero di parametri passati

Array di stringhe passate dalla linea di comando

Conversione ascii → intero

Copia stringhe

---

gcc argomenti.c
./a.out 5 aa

Lettura/scrittura su terminale

```
int i;
scanf("%d", &i);
printf("%d",i);
```

Definisce un intero complemento a 2 a 32 bit

Range=- 2^{31} ... +2^{31}

Legge un intero a 32 bit

Scrive un intero a 32 bit

```
unsigned int u;
scanf("%u", &u);
printf("%u",u);
```

Definisce un intero senza segno a 32 bit

Legge un intero senza segno a 32 bit

Scrive un intero senza segno a 32 bit

```
scanf("%o", &u);
printf(%o",u);
```

Legge 'u' in base 8

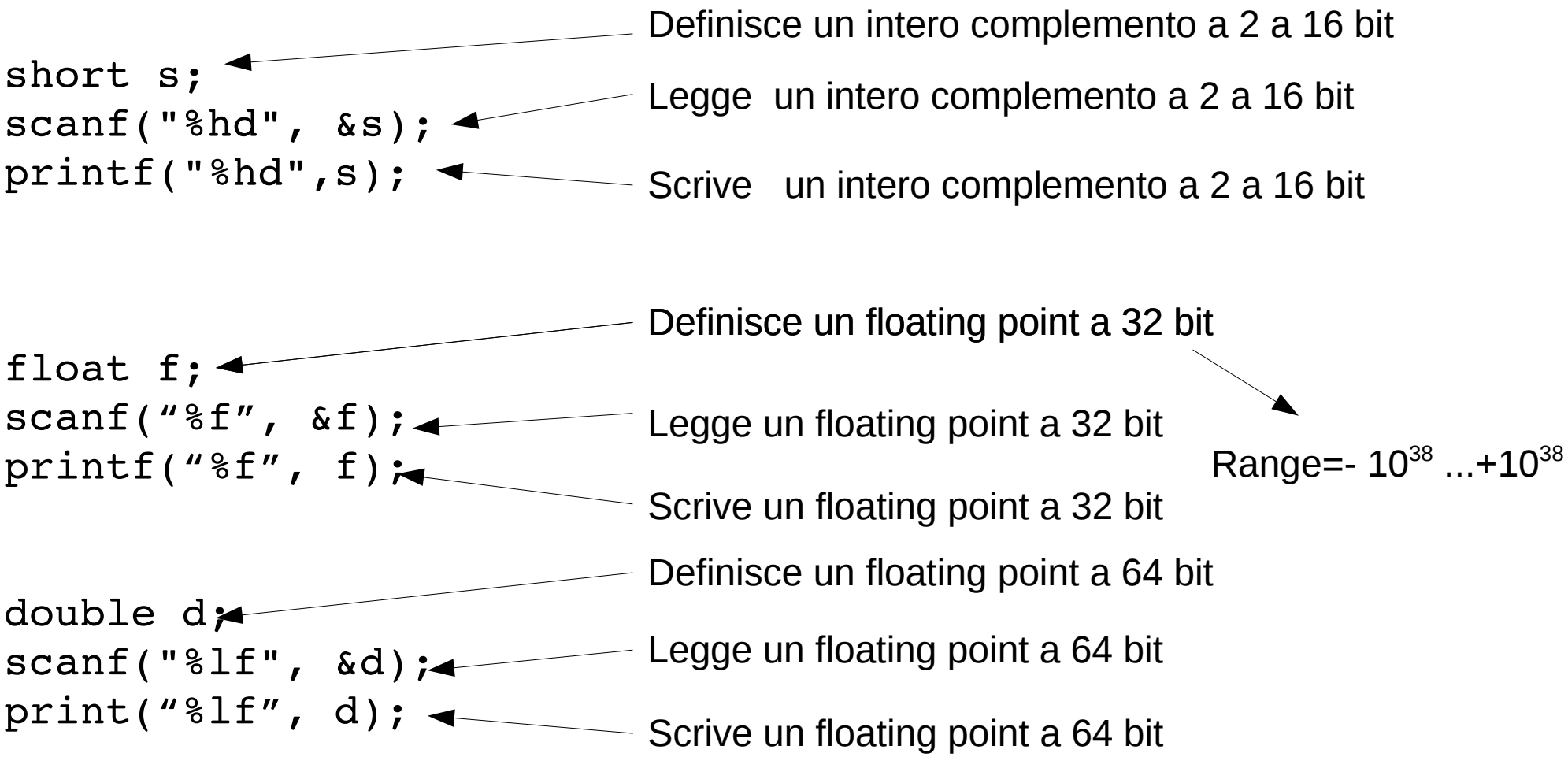
Scrive 'u' in base 8

```
scanf("%x", &u);
printf(%x",u);
```

Legge 'u' in base 16

Scrive 'u' in base 16

Lettura/scrittura su terminale



Tipo carattere

Hex	Char
...	…
41	A
42	B
...	...
61	a
62	b

- Il tipo carattere è **`char`**
- Una variabile char occupa 8 bit
- Un carattere può essere usato come numero da -127 a +128
- Un carattere da 0 a 255 è definito unsigned char
- Normalmente un char contiene un carattere in codice Ascii
- Relazione d'ordine tra caratteri: `'A' < 'a';`

```
char c='a'; printf("%c %c %d\n",c, 'a','a');
```

- Lettura/scrittura:

```
char ch;
scanf("%c",&ch);
printf("%c", ch);
```

Oppure(più semplice)

```
char ch;
putchar(ch);
ch = getchar();
```

Tipo carattere

- minuscolo → maiuscolo

```
#include <stdio.h>
int main(void) {
    char c;
    for ( ; ; ) {
        c = getchar();
        if (c == EOF) break;
        if ((c >= 97) && (c < 123))
            c -= 32;
        putchar(c);
    }
    return 0;
}
```

- Oppure (`'a'-'A'`=20_{hex}=32_{dec})

```
#include <stdio.h>
int main(void) {
    char c;
    for ( ; ; ) {
        c = getchar();
        if (c == EOF) break;
        if ((c >= 97) && (c < 123))
            c = c -'a' + 'A';
        putchar(c);
    }
    return 0;
}
```

Stringhe

- Stringa = array di caratteri

```
char nome[6];
nome[6] = "Giulio";
printf("nome=%s\n",nome);

scanf("%s",nome);
```

- Nel passaggio parametri al main

```
int main(int argc, char **argv)
{
   int i;
   for (i=0;i<argc;i++) {
      printf("%s\n", argv[i]);
   }
   return 0;
}
```

Esempio: lunghezza stringa (I)

```
#include <stdio.h>
int main(void)
{
   char ch;
   int len = 0;
   printf("Scrivi una stringa: ");
   ch = getchar();
   while (ch != '\n') {
      len++;
      ch = getchar();
   }
   printf("la stringa e' lunga %d caratteri\n", len);
   return 0;
}
```

Esempio: lunghezza stringa (II)

```
#include <stdio.h>
int main(void)
{
   char ch;
   int len = 0;
   printf("Scrivi una stringa: ");
   while (ch=getchar() != '\n') {len++;}
   printf("La stringa e' lunga %d caratteri\n", len);
   return 0;
}
```

Copia di stringhe

```
#include <stdio.h>
char *my_strcpy(char *destination, char *source)
{
    char *p = destination;
    while (*source != '\0'){ *p++ = *source++;    }
    *p = '\0';
    return destination;
}

char *my_strcpy1(char dest[], char source[])
{
    int i = 0;
    while (source[i] != '\0'){
        dest[i] = source[i];
        i++;
    }
    dest[i] = '\0';
    return dest;
}

int main(void)
{   char strA[10]="abcd"; char strB[10];
    my_strcpy(strB, strA);//my_strcpy1(strB, strA);
    puts(strB);
}
```

Libreria stringhe (string.h)

- concatena src a dest

```
strcat(char *dest, const char *src);
```

- cerca in str la prima occorrenza di c

```
char *strchr(const char *str, int c);
```

- confronta s1 s2

```
int strcmp(const char *s1, const char *s2); → <0 =0 >0
```

- copia src in dest

```
char *strcpy(char *dest, const char *src);
```

- calcola la lunghezza di una stringa

```
size_t strlen(const char *str);
```

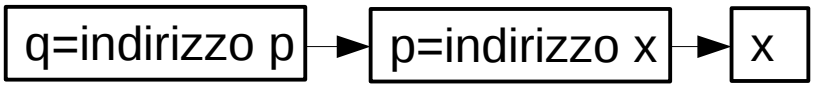
- cerca in str la prima occorrenza della stringa sub

```
char *strstr(const char *str, const char *sub)
```

Alcune definizioni riassuntive di Puntatori

int *C	C è un puntatore a interi
int *E[5]	E è un array di 5 puntatori a interi
int G (…)	G è una funzione che ritorna un intero
char * H (…)	H è una funzione che ritorna un puntatore a char
int **D	D è un puntatore a int*. Per es.:

```
{int x = 10; int* p=&x; int** q=&p;}
```



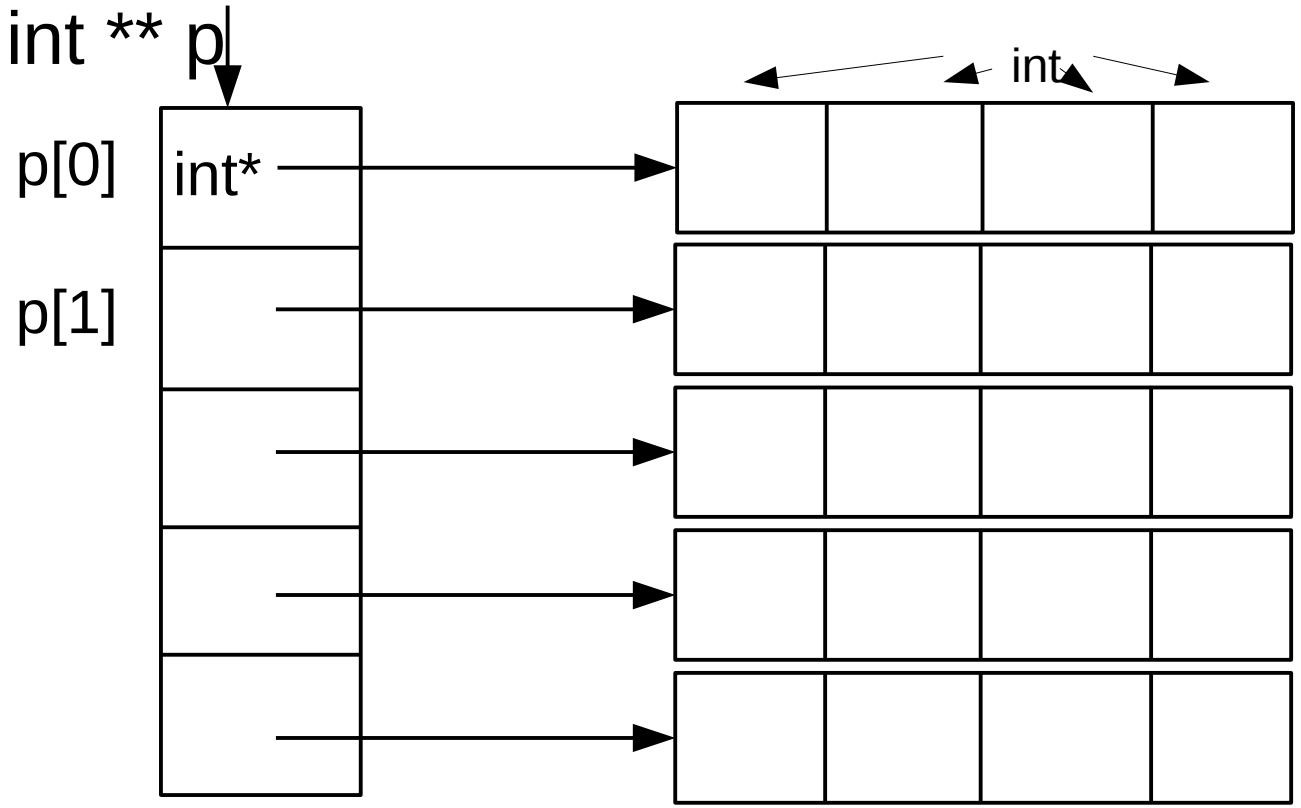
const int* ptr *ptr non può essere cambiato

int* const ptr ptr non può essere cambiato

const int* const ptr → nè *ptr nè ptr possono cambiare

Struttura delle matrici in memoria

- Dinamiche



- Statiche

int A[2][3] → A[0][0] A[0][1] A[0][2] A[1][0] A[1][1] A[1][2]
Per cui: A[0][2] = *(A[0] + 2), A[1][2] = *(A[1] + 2)

Puntatori e matrici

- L'incremento dei puntatori dipende dal tipo

```
#include <stdio.h>
#include <stdlib.h>
void scrivi(int *p, int m){
    short i,j;
    for(i=0;i<m;i++){
        printf("[] %d |",p[i]);
        printf("char* %d |",(int) *((char*)p + 4*i));
        printf("short* %d |",(int) *((short*)p + 2*i));
        printf("int* %d |",(int) *((int*)p + i)); //("int* %d |",*(p+i));
        printf("\n");
    }
}
int main() {
    int vet[5]={1,2,3,4,5};
    int *p = vet;

    printf("nr di byte dei char=%d ", sizeof(char));
    printf("nr di byte dei short=%d ", sizeof(short));
    printf("nr di byte dei int=%d ", sizeof(int));
    printf("\n\n");

    scrivi(p,5);  //oppure  scrivi(vet,5)
}
```

Se p diventa puntatore a char, le somme spostano p di 1 byte, per cui moltiplico per 4

Il vettore si passa tramite 'p' oppure tramite il suo nome 'vet'

Puntatori e matrici

- Funzione che legge un file di interi e li mette in A[]

```
#include <stdio.h>
#include <stdlib.h>

int leggi(char* filename, int A[], int* ptr){
   FILE* fp=NULL;      int num;    *ptr=0;
   if ((fp=fopen(filename,"r")) != NULL){
      while (fscanf(fp,"%d",&num)>0){
         A[*ptr]= num;   (*ptr)++;
      }
      return 0;
   }
   else return 1;
}
int main(void){
   int* p;   int A[10];   char nome[10]="file";     int i;

   leggi(nome,A,p);
   for(i=0;i<*p;i++)printf("%d ", A[i]);
   printf("\n");
}
```

Apro il file in lettura ('r')

Leggo il file un numero alla volta

Uso il puntatore come indice per avere il nr di elementi all'uscita della funzione

Esempio d'uso delle strutture

```
#include <stdio.h>
#include <string.h>
struct tag{          /* struttura dati */
   char lname[20];  /* cognome */
   char fname[20];  /* nome */
   int age;         /* eta' */
   float rate;      /* e.g. 12.75 all'ora */
};
struct tag my_struct; /* define the structure */

void show_name(struct tag *p)
{
   printf("%s ", p->fname); /* p punta alla struttura */
   printf("%s ", p->lname);
   printf("%d\n", p->age);
}
int main(void)
{
   struct tag *st_ptr;     /* puntatore alla struttura */
   st_ptr = &my_struct; /* inizializza il puntatore */
   strcpy(my_struct.lname,"Tamborra");
   strcpy(my_struct.fname,"Giulio");
   my_struct.age = 63;
   show_name(st_ptr);  /* passa il puntatore */
   return 0;
}
```

Operatore accedi al campo della struct: '→' se dinamico, '.' se statico

Array 2D globali

```
#include <stdio.h>
#define ROWS 5
#define COLS 10

int multi[ROWS][COLS];

int main(void)
{
    int row, col;
    for (row = 0; row < ROWS; row++){
        for (col = 0; col < COLS; col++){
            multi[row][col] = row*col;
        }
    }
    for (row = 0; row < ROWS; row++){
        for (col = 0; col < COLS; col++){
            printf("\n%d ",multi[row][col]);
            printf("%d ",*(*(multi + row) + col));
            printf("%d ",*(multi[row]+col));
        }
    }
    return 0;
}
```

Definizione matrice

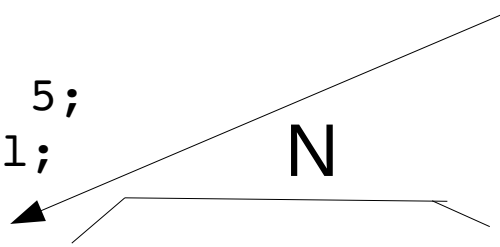
Riempimento elementi

3 modi per accedere agli elementi della matrica statica

Array 2D dinamici (I)

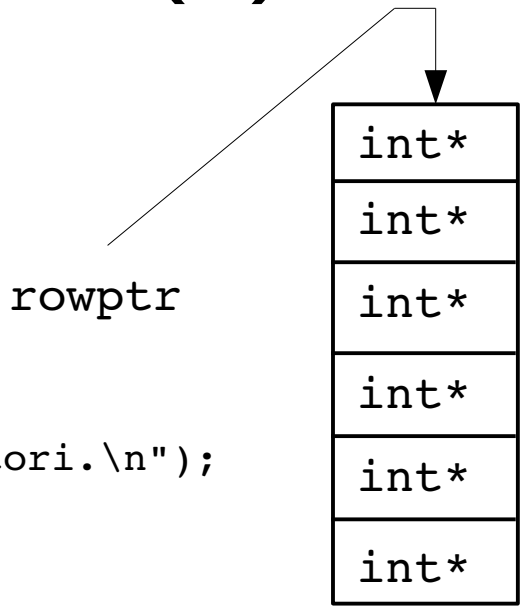
```
#include <stdio.h>
#include <stdlib.h>
#define COLS 5

int *rptr;
int main(void)
{
    int nrows = 5;
    int row, col;

    rptr = malloc(nrows * COLS*sizeof(int) );
    for (row = 0; row < nrows; row++){
        for (col = 0; col < COLS; col++){
            *(rptr+row*COLS+col) = row+col;
        }
    }
    for (row = 0; row < nrows; row++){
        for (col = 0; col < COLS; col++){
            printf("%d ",*(rptr+row*COLS+col) );
        }
        printf("\n");
    }
    return 0;
}
```

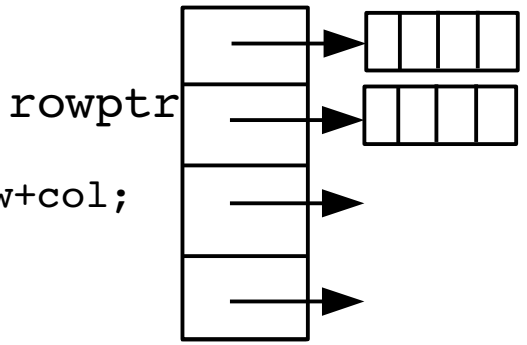
Alloca N=nrows*COLS bytes in heap



Array 2D dinamici (II)

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int nrows = 5;  int ncols = 4;
    int row,col;                int **rowptr;
    rowptr = malloc(nrows * sizeof(int *));
    if (rowptr == NULL){
        puts("\nErrore di allocazione vettore puntatori.\n");
        exit(0);
    }
// allocazione righe
    for (row = 0; row < nrows; row++){
        rowptr[row] = malloc(ncols * sizeof(int));
        if (rowptr[row] == NULL){
            printf("\nErrore allocazione riga [%d]\n",row);
            exit(0);
        }
        for(col = 0; col<ncols; col++) *(rowptr[row]+col)=row+col;
    }
// stampa matrice
    for (row = 0; row < nrows; row++){
        for(col = 0; col<ncols; col++)
            printf("%d ",*(rowptr[row]+col));
    }
    return 0;
}
```





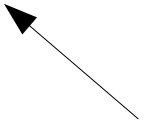
Passaggio di vettori (I)

```
#include <stdio.h>
int arr[10] = { 3,6,1,2,3,8,4,1,7,2};

void bubble(int a[], int N);

int main(void)
{
    int i;
    putchar('\n'); for (i = 0; i < 10; i++)   printf("%d ", arr[i]);
    bubble(arr,10);
    putchar('\n'); for (i = 0; i < 10; i++)   printf("%d ", arr[i]);
    return 0;
}
void bubble(int a[], int N)  /*oppure int *a */
{
    int i, j, t;
    for (i = N-1; i >= 0; i--){
        for (j = 1; j <= i; j++){
            if (a[j-1] > a[j]){
                t = a[j-1];
                a[j-1] = a[j];
                a[j] = t;
            }
        }
    }
}
```

Passaggio di vettori (II)

- Introduzione della funzione compare()

```
#include <stdio.h>
int arr[10] = { 3,6,1,2,3,8,4,1,7,2};
void bubble(int *p, int N);
int compare(int *m, int *n);
void main(void){
    int i;
    putchar('\n'); for (i = 0; i < 10; i++)  printf("%d ", arr[i]);
    bubble(arr,10);
    putchar('\n'); for (i = 0; i < 10; i++) printf("%d ", arr[i]);
}

int compare(int *m, int *n) { return (*m > *n); }

void bubble(int *p, int N)
{
    int i, j, t;
    for (i = N-1; i >= 0; i--)
        for (j = 1; j <= i; j++){
            if (compare(&p[j-1], &p[j])){
                t = p[j-1];
                p[j-1] = p[j];
                p[j] = t;
            }
        }
}
```



Ritorna 0 se ***m<***n
Ritorna 1 se *m>*n

Passaggio di vettori (III)

- Passa a compare() argomenti void* di cui fa il casting a int*

```
#include <stdio.h>
int arr[10] = { 3,6,1,2,3,8,4,1,7,2};
void bubble(int *p, int N);  int compare(void *m, void *n);
int compare(void *m, void *n){
    int *m1, *n1;
    m1 = (int *)m; n1 = (int *)n;
    return (*m1 > *n1);
}

void bubble(int *p, int N){
    int i, j, t;
    for (i = N-1; i >= 0; i--)
        for (j = 1; j <= i; j++){
            if (compare((void *)&p[j-1], (void *)&p[j])){
                t = p[j-1];  p[j-1] = p[j];   p[j] = t;
            }
        }
}
int main(void){
    int i;
    putchar('\n'); for (i = 0; i < 10; i++) printf("%d ", arr[i]);
    bubble(arr,10);
    putchar('\n'); for (i = 0; i < 10; i++) printf("%d ", arr[i]);
    return 0;
}
```

Passaggio di vettori (IV)

- Esempio di funzione memcpy

```
#include <stdio.h>
#include <string.h>
long arr[10] = { 3,6,1,2,3,8,4,1,7,2};
void bubble(void *p, size_t width, int N); int compare(void *m, void *n);
int main(void)
{
    int i;
    putchar('\n'); for (i = 0; i < 10; i++) printf("%d ", arr[i]);
    bubble(arr, sizeof(long), 10);
    putchar('\n'); for (i = 0; i < 10; i++) printf("%ld ", arr[i]);
    return 0;
}
void bubble(void *p, size_t width, int N){
    int i, j;
    unsigned char buf[4];
    unsigned char *bp = p;
    for (i = N-1; i >= 0; i--)
        for (j = 1; j <= i; j++)
            if (compare((void *)(bp + width*(j-1)),(void *)(bp + j*width))) {
                memcpy(buf, bp + width*(j-1), width);          /* t = p[j-1]; */
                memcpy(bp + width*(j-1), bp + j*width , width);/* p[j-1] = p[j]; */
                memcpy(bp + j*width, buf, width);              /* p[j] = t; */
            }
}
int compare(void *m, void *n){
    long *m1, *n1;
    m1 = (long *)m; n1 = (long *)n;
    return (*m1 > *n1);
}
```

```
void *memcpy(void *dest, const void *src, size_t n)
```

Copia n bytes da src a dest

Passaggio di vettori (V)

- Gli elementi sono stringhe: confronto con strcmp

```
#include <stdio.h>
#include <string.h>
#define MAX_BUF 256
char arr2[5][20] = { "Topolino","Paperino","Minnie","Pippo","Pippa" };

void bubble(void *p, int width, int N){
     int i, j, k;  unsigned char buf[MAX_BUF];  unsigned char *bp = p;
     for (i = N-1; i >= 0; i--)
          for (j = 1; j <= i; j++){
               k = compare((void *)(bp + width*(j-1)),(void *)(bp + j*width));
               if (k > 0){
                    memcpy(buf, bp + width*(j-1), width);
                    memcpy(bp + width*(j-1), bp + j*width , width);
                    memcpy(bp + j*width, buf, width);
               }
          }
}
int compare(void *m, void *n){
     char *m1 = m;
     char *n1 = n;
     return (strcmp(m1,n1));
}
int main(void)
{
     int i;
     putchar('\n');for (i = 0; i < 5; i++)printf("%s\n", arr2[i]);
     bubble(arr2, 20, 5);
     putchar('\n\n');for (i = 0; i < 5; i++)printf("%s\n", arr2[i]);
     return 0;
}
```

Passaggio di matrici dinamiche con **

```
#include <stdio.h>
#include <stdlib.h>
void stampa(int **p, int nrows, int ncols){
     int row,col;
     for (row = 0; row < nrows; row++){// stampa matrice
          for(col = 0; col<ncols; col++)
               printf("%d ",p[row][col]);
               /*printf("%d ",*(*(p+row)+col));*/
               /*printf("%d ",*(p[row]+col));*/
     }
}
int main(void){
     int nrows = 5;    int ncols = 4;
     int row,col;      int **rowptr;
     rowptr = malloc(nrows * sizeof(int *));
     if (rowptr == NULL){
          puts("\nErrore di allocazione vettore puntatori.\n");
          exit(0);
     }
     for (row = 0; row < nrows; row++){// allocazione righe
          rowptr[row] = malloc(ncols * sizeof(int));
          if (rowptr[row] == NULL){
               printf("\nErrore allocazione riga [%d]\n",row);
               exit(0);
          }
          for(col = 0; col<ncols; col++) *(rowptr[row]+col)=row+col;
     }
     stampa(rowptr,nrows,ncols);
     return 0;
}
```

Passaggio puntatore doppio

Prima possibilità

Seconda possibilità

Terza possibilità

3 modi per Accedere agli elementi

Definizione puntatore doppio

Passaggio puntatore doppio

Passaggio per indirizzo di matrice globale

- Le matrici statiche sono definite come [int*][const int*]
- Non si possono definire come int**
- Si possono definire come vettori monodimensionali tramite casting

```
#include <stdio.h>
#define numRows 3
#define numCols 2

int arr[numRows][numCols]={ { 0, 0}, { 0, 0 }, { 0, 0 } };

void display(int *p){
    int i,j;
    for ( i = 0; i< numRows;i++){
        for ( j = 0;j< numCols;j++){
            printf("%d\t",*(p+i*numCols+j));
        }
        printf("\n");
    }
}
int main() {
      display((int*)&arr[0][0]); // oppure display((int*)arr)
}
```