

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
```

```
{
    printf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
```

```
{
    printf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



# Algorithms and Data Structures

## Entry Test

Stefano Quer

Department of Control and Computer Engineering

Politecnico di Torino

# Entry Test

## ❖ Theory entry test

### ➤ No correction

- Please, see solutions
- Please, see the “Programming Techniques” course

## ❖ Programing entry test

### ➤ Solution on source C file and in the following pages

- Please, see also unit 01 section 01 (common errors in C code)

## Problem-solving: Ex 1

- ❖ A file has the following content

```
100000 gandalf 30
123456 aragorn 28
...
```

- ❖ Is the following segment of code correct?

```
i = 0;
while (fgets (row, 100, fp) != NULL) {
    sscanf (row, "%d", &student[i].register);
    sscanf (row, "%s", student[i].name);
    sscanf (row, "%d", &student[i].mark);
    i++;
}
```

## Problem-solving: Ex 1

- ❖ A file has the following content

```
100000 gandalf 30
123456 aragorn 28
...
```

- ❖ Is the following segment of code correct?

```
i = 0;
while (fgets (row, 100, fp) != NULL) {
    sscanf (row, "%d", &student[i].register);
    sscanf (row, "%s", student[i].name);
    sscanf (row, "%d", &student[i].mark);
    i++;
}
```

No, it is not  
Because, sscanf (unlike fscanf) does  
not “move” along the string  
How to correct it?

## Problem-solving: Ex 2

- ❖ A file stores an indefinite sequence of integer values
- ❖ A programmer reads it as

```
i = 0;
while (!feof (fp)) {
    fscanf (fp, "%d", &v[i]);
    i++;
}
```

- ❖ Is the code correct?

## Problem-solving: Ex 2

- ❖ A file stores an indefinite sequence of integer values
- ❖ A programmer reads it as

```
i = 0;  
while (!feof (fp)) {  
    fscanf (fp, "%d", &v[i]);  
    i++;  
}
```

- ❖ Is the code correct?

No, it is not  
Because, function feof is true only  
after reading the end of file  
How to correct it?

## Problem-solving: Ex 3

❖ A programmer writes

```
switch (str) {  
    case "one": printf ("String one\n"); break;  
    case "two": printf ("String one\n"); break;  
    case "three": printf ("String one\n"); break;  
    default: printf ("String larger than 3); break;  
}
```

❖ Is the code correct?

## Problem-solving: Ex 3

❖ A programmer writes

```
switch (str) {  
    case "one": printf ("String one\n"); break;  
    case "two": printf ("String one\n"); break;  
    case "three": printf ("String one\n"); break;  
    default: printf ("String larger than 3); break;  
}
```

❖ Is the code correct?

No, it is not  
Because, In C string must be manipulated  
using the string library function  
How to correct it?



## Problem-solving: Ex 4

❖ A programmer writes

```
void swap_string (char str1[], char str2[]) {  
    char *tmp;  
    tmp = str1;  
    str1 = str2;  
    str2 = tmp;  
    return;  
}
```

❖ Is the code correct?

## Problem-solving: Ex 4

❖ A programmer writes

```
void swap_string (char str1[], char str2[]) {  
    char *tmp;  
    tmp = str1;  
    str1 = str2;  
    str2 = tmp;  
    return;  
}
```

❖ Is the code correct?

to swap strings, strcpy:  
strcpy(tmp, s1);  
strcpy(s1, s2);  
strcpy(s2, tmp);

Possibly, there are 2 problems

1. Swap changes the pointers not the string
2. str1 and str2 look like two array of characters. If they are constant pointers, they cannot be modified

## Problem-solving: Ex 5

- ❖ A programmer writes

```
struct student_s {  
    int register;  
    char name[L];  
    float mark;  
} student_t;  
student_t mys;
```

- ❖ Explain which are the differences between the following code instructions

```
mys.mark  
mys->mark  
(*mys).mark
```

## Problem-solving: Ex 5

- ❖ A programmer writes

```
struct student_s {  
    int register;  
    char name[L];  
    float mark;  
} student_t;  
student_t mys;
```

- ❖ Explain which are the differences between the following code instructions

```
mys.mark  
mys->mark  
(*mys).mark
```

In `mys.mark`, `mys` is supposed to be a variable  
In `mys->mark` and `(*mys).mark`, `mys` is supposed to be a pointer

## Problem-solving: Ex 6

- ❖ Given the following definitions

```
typedef struct item_s {  
    char c1;  
    int i1;  
    char c2;  
    int i2;  
} item_t;  
item_t myitem;
```

- ❖ If an integer occupies 4 bytes of memory, does **myitem** occupy always 10 bytes?

## Problem-solving: Ex 6

- ❖ Given the following definitions

```
typedef struct item_s {  
    char c1;  
    int i1;  
    char c2;  
    int i2;  
} item_t;  
item_t myitem;
```

- ❖ If an integer occupies 4 bytes of memory, does **myitem** occupy always 10 bytes?

because it is stored in words (4bytes collectio) so

```
c1 ---  
i1 i1 i1 i1  
c2 ---  
i2 i2 i2 i2  
--> 16bytes
```

it would be 12 if we declared it as:  
char c1,c2;  
int i1,i2;

Usually 16  
Possibly 12

## Problem-solving: Ex 7

❖ Given the following definitions

```
int i = 10;  
int j = 20;  
int *p1 = &i;  
int *p2 = &j;
```

❖ Which ones among the following conditions are true?

```
p1==p2  
*p1==*p2  
p1>=p2  
p1<=p2  
p1!=p2  
*p1!=*p2
```

## Problem-solving: Ex 7

❖ Given the following definitions

```
int i = 10;  
int j = 20;  
int *p1 = &i;  
int *p2 = &j;
```

❖ Which ones among the following conditions are true?

```
p1==p2  
*p1==*p2  
p1>=p2  
p1<=p2  
p1!=p2  
*p1!=*p2
```

```
F  
F  
T or F  
F or T  
T  
T
```



## Problem-solving: Ex 8

❖ Given the following definitions

```
char s1[]="string";  
char s2[6]={'s','t','r','i','n','g'};  
char *p1="string";
```

❖ Which ones among the following conditions are true?

```
sizeof(s1)==7  
sizeof(s2)==6  
sizeof(p1)==6  
sizeof(*p1)==6
```

## Problem-solving: Ex 8

❖ Given the following definitions

```
char s1[]="string";  
char s2[6]={'s','t','r','i','n','g'};  
char *p1="string";
```

❖ Which ones among the following conditions are true?

```
sizeof(s1)==7  
sizeof(s2)==6  
sizeof(p1)==6  
sizeof(*p1)==6
```

7  
6  
8  
1

because of the string terminator.  
`strlen(s1)=6`

pointer: 4 (x32) or 8 (x64  
machines)

datum pointed by pointer: first  
character

## Problem-solving: Ex 9

- ❖ The following function **myf** is called with the string **in** storing "This is a very loooong string", and with the string **out** and the value **n** undefined
- ❖ Specify the value of the string **out** and the integer **n** returned by the function

# Problem-solving: Ex 9

```
void myf (char *in, char *out, int *n) {  
    char *tmp1, *tmp2;  
    int l;  
    out[0] = '\\0';  
    tmp1 = in;  
    while (*tmp1!='\\0') {  
        while (*tmp1==' ') {  
            tmp1++;  
        }  
        tmp2 = tmp1;  
        while (*tmp2!=' ' && *tmp2!='\\0') {  
            tmp2++;  
        }  
        l = tmp2 - tmp1;  
        if (l > strlen(out)) {  
            *n=1;  
            strncpy (out, tmp1, l);  
            out[l] = '\\0';  
        }  
        tmp1=tmp2;  
    }  
    return;  
}
```

in = "This is a very loooong string"  
out undefined  
n undefined

out = ???  
n = ???

loooong

7

## Problem-solving: Ex 9

```
void myf (char *in, char *out, int *n) {  
    char *tmp1, *tmp2;  
    int l;  
    out[0] = '\\0';  
    tmp1 = in;  
    while (*tmp1!='\\0') {  
        while (*tmp1==' ') {  
            tmp1++;  
        }  
        tmp2 = tmp1;  
        while (*tmp2!=' ' && *tmp2!='\\0') {  
            tmp2++;  
        }  
        l = tmp2 - tmp1;  
        if (l > strlen(out)) {  
            *n=1;  
            strncpy (out, tmp1, l);  
            out[l] = '\\0';  
        }  
        tmp1=tmp2;  
    }  
    return;  
}
```

in = "This is a very loooong string"  
out undefined  
n undefined

out = loooong  
n = 7

## Problem-solving: Ex 10

- ❖ The following function **f** is called with the **s** storing "This 12345 is a string"
- ❖ Which is the value of **s** when the function returns?

s = "This 12345 is a string"

```
void f (char *s) {  
    int i, j;  
    i = 0;  
    while (i < strlen(s)) {  
        if (s[i]==' ' || (s[i]>='0' && s[i]<='9')) {  
            for (j=i+1; j<strlen(s)+1; j++)  
                s[j-1] = s[j];  
        } else {  
            i = i + 1;  
        }  
    }  
    return;  
}
```

s = ???

Thisisastring

## Problem-solving: Ex 10

s = "This 12345 is a string"

```
void f (char *s) {
    int i, j;
    i = 0;
    while (i < strlen(s)) {
        if (s[i]==' ' || (s[i]>='0' && s[i]<='9')) {
            for (j=i+1; j<strlen(s)+1; j++)
                s[j-1] = s[j];
        } else {
            i = i + 1;
        }
    }
    return;
}
```

s = "Thisisastring"

## Problem-solving: Ex 11

### ❖ Write the function

```
void substring (char *str, int *letter, int *digit);
```

- Receiving a string **str** as a parameter
- Computing the length of the longest sub-sequence of small alphabetic characters (letter) and of decimal digits (digit)

### ❖ Example

- If `str = "This is 1 string including diGiTs: 12345 678 9"`
- The two sequences are "string" and "12345", thus the program must return `letter=6` and `digit=5`



## Problem-solving: Ex 11

```
void substring (char *str, int *letter, int *digit) {
    int i, l, d;
    *letter = l = 0;
    *digit = d = 0;
    for (i=0; i<strlen(str); i++) {
        if (str[i]>='a' && str[i]<='z') {
            l++;
        } else {
            if (l>*letter)
                *letter = l;
            l = 0;
        }
        if (str[i]>='0' && str[i]<='9') {
            d++;
        } else {
            if (d>*digit)
                *digit = d;
            d = 0;
        }
    }
    return;
}
```

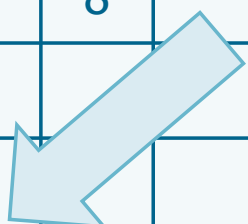
# Problem-solving: Ex 12

## ❖ Write the function

```
display (float **mat, int n);
```

- Receiving the matrix **mat** and its size **n**
- Displaying (on the standard output) all the values stored in the matrix (i.e., **nxn** float values) with the order reported by the following picture

	0	1	2	3	...	
0	1	2	4	7	11	k=2 02 11 20
1	3	5	8			k=3 03 14 23 32 41
2	6	9				
3	10					k=2 24 33 42
...						

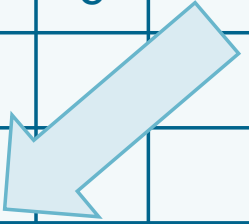


## Problem-solving: Ex 12

For each starting point  $k$   
 $i$  indicates the row  
 $j$  indicates the column

```
int display (float mat[][N], int n) {
    int i, j, k;
    for (k=0; k<n; k++) {
        for (i=0, j=k; i<n && j>=0; i++, j--) {
            fprintf (stdout, "%.2f ", mat[i][j]);
        }
    }
    for (k=1; k<n; k++) {
        for (i=k, j=n-1; i<n && j>=0; i++, j--) {
            fprintf (stdout, "%.2f ", mat[i][j]);
        }
    }
    fprintf (stdout, "\n");
    return 1;
}
```

	0	1	2	3	...
0	1	2	4	7	11
1	3	5	8		
2	6	9			
3	10				
...					



## Problem-solving: Ex 13

### ❖ Write the function

palindrome: same word if read normally or backwards

```
int palindrome (char *str);
```

- Receiving a string **str** of unknown length
- Returning the length of the longest palindrome substring of **str**

### ❖ Example

- If `str="1234554abccbaxxYY"`
- The longest palindrome substring of `str` is `"abccba"`, thus, the function must return 6

# Problem-solving: Ex 13

```
int palindromel (char *str) {
    int i, j, k, l, lm, s;

    s = lm = 0;
    for (i=0; i<strlen (str); i++) {
        for (j=strlen (str)-1; j>i; j--) {
            l = 0;
            for (k=0; k<=((j-i+1)/2); k++) {
                if (str[i+k]==str[j-k]) {
                    l+=2;
                } else {
                    break;
                }
            }
            if (l>lm) {
                s = i;
                lm = j-i+1;
            }
        }
    }
}
```

Start from left and right  
move toward center

```
#if 0
    if (lm!=0) {
        fprintf (stdout, "Longest substrig: ");
        for (i=s; i<s+lm; i++) {
            fprintf (stdout, "%c", str[i]);
        }
    }
#endif
return lm;
}
```

## Problem-solving: Ex 13

```
int palindrome2 (char *str) {
    int i, l, r, len, lenm;
    lenm = 0;
    for (i=0; i<strlen (str); i++) {
        l=r=i; len=-1;
        while (l>=0 && r<strlen(str)) {
            if (str[l]==str[r]) {
                len*=2; l--; r++; len+=2;
            } else {
                break;
            }
        }
        if (len>lenm) lenm = len;
    }
}
```

Start from center;  
move left and right

Check odd length

Check even length

```
l=i; r=i+1; len = 0;
while (l>=0 && r<strlen(str)) {
    if (str[l]==str[r]) {
        len+=2; l--; r++;
    } else {
        break;
    }
}
if (len>lenm) lenm = len;
}
return lenm;
}
```