Lezione 3 - 22/3/21 (C)

C

N.B.

I return dei main.c hanno un solo byte e quindi una rappresentazione di 255 massimo

Librerie

stdio.h —> FILE, EOF, stderr, stdin, stdout, fclose(), etc…

stdlib.h —> atof(), atoi(), malloc(), free(), exit(), system(), rand(), etc…

string.h —> memset(), memcpy(), strncat(), strcmp(), strlen(), etc…

math.h —> sin(), cos(), sqrt(), floor(), etc…

unistd.h —> STDOUT_FILENO, read(), write(), fork(), pipe(), etc…

fcntl.h —> creat(), open(), etc…

Union

Si dichiara come una struct ma al posto di avere i singoli valori elencati può averne uno o l'altro e se vado a modificare quello che non ho ancora assegnato vado a perdere quello che avevo nella variabile tipo precedentemente assegnata. Praticamente è una variabile che accetta più tipi e scrive sulla stessa cella di memoria.

```
union Result{
    int intero;
    float decimale;
} result1, result2;

union Result result3;
result3.intero = 22;
result3.decimale = 11.5;
```

enum

E' un tipo di variabile che può prendere dei valori diversi pre-decisi e dichiarati come costanti nel modo seguente:

```
#include <stdio.h>

enum State {Undef = 9, Working = 1, Failed = 0};
void main() {
    enum State state=Undef;
    printf("%d\n", state);
}
```

Se non scrivevo "Undef=9, ..." venivano associati automaticamente i valori "0,1,2" alle variabili messe dentro ("Undef, Working, Failed").

Interazioni con i file

```c
#include <stdio.h>

FILE *ptr;
ptr = fopen("filename.txt","r+");

int id;
char str1[10], str2[10];
while (!feof(ptr)){
    fscanf(ptr,"%i %s %s", &id, str1, str2);
    printf("%i %s %s\n",id,str1,str2);
}

printf("End of file");

fclose(ptr);
```

Funzioni utili:

- **fopen("filename.txt","mode")** → Apre un file e salva il suo inizio in un puntatore.
- **rewind(ptr)** → Resetta il puntatore all'inizio del file.
- **fgets(DestString, NOfChar, ptr)** → Legge un numero "NOfChar" predefinito di caratteri e li salva su una stringa.
- **fgetc(ptr)** → Legge un solo carattere dal file.
- **printf("ciao")** → Stampa "ciao" su terminale.
- **fprintf(stdout, "ciao")** → Stampa "ciao" su stdout (cioè il termninale). (Di solito utilizzato per scrivere su file).
- **creat("nomeFile.txt", S_IRUSR|S_IWUSR);** → Crea un file e da all'utente che lo crea i diritti di lettura (S_IRUSR) e di scrittura (S_IWUSR).

- **open("nomeFile.txt", 0_CREAT|0_RDWR, S_IRUSR|S_IWUSR);** → Apre o crea (se non esiste) un file se però li viene dato come parametro "0_CREAT" (Attenzione è uno zero!!) e poi accetta come parametri anche i diritti di scrittura e lettura come il comando precedente.
- **lseek(fileptr, n, SEEK_SET)** → Permette di spostare la "testina" di letture del file di "n" posti. (SEEK_SET, SEEK_CUR, SEEK_END) (SEEK_SET → dall'inizio)
- **sizeof(arg)** → Ci dice quanto spazio occupa una variabile o una costante o un tipo.

Placeholder

- **%d** → int
- **%c** → char
- **%s** →string

Vettori C

`char str[5]={'c', 'i', 'a', 'o', 0}` →5*1 bytes

str è un puntatore a char e quindi:

`str[n] = *(str+n)`

e dunque:

`str[0] = *(str+0) = *(str) = *str`

Funzioni stringhe

- **char *strcat(char *dest, const char *src)** → Attacca "src" in coda a "dst".
- **char *strchr(const char *str, in c)** → Cerca la prima occorrenza di "c" in "str".
- **int strcmp(const char *str, const char *str2)** → Confronta "str1" con "str2" (0 se uguali).
- **size_t strlen(const char *str)** → calcola la lunghezza di "str".