UNIVERSITÀ DEGLI STUDI DI UDINE
Francesco Davini

Sistemi operativi - Laboratorio

Introduzione a C

Esercizio: Hello World

Scrivere un programma in C che stampi "Hello, World!" sulla console.

```c
#include<stdio.h>
/* Questo programma
stampa un saluto */
int main() {
    printf("Hello, World!\n");
    return(0); // il valore restituito: 0
}
```

- **main()** è una funzione che restituisce un valore intero (di tipo int) tramite lo statement **return(0)**;
- Il simbolo **//** inizia un commento (fino a fine riga), mentre commenti su più righe sono delimitati da /* */
- **printf("Hello, World!\n");** scrive sullo standard output stream (chiamato stdout) la stringa "Hello, World!\n" (il carattere **\n** indica un ritorno a capo)
- le definizioni di stdout e della funzione **printf()** sono rese disponibili tramite la direttiva **#include<stdio.h>** che richiede l'inclusione della libreria standard di I/O
- Il programma deve essere scritto in un file con estensione .c Supponiamo di scriverlo in un file di testo hello.c
- a differenza di Java, non c'è relazione tra il nome del file e il suo contenuto
- Per poter esser eseguire, il programma hello.c deve essere:
    - PRE-PROCESSATO
    - COMPILATO
    - COLLEGATO (LINKATO)
    - CARICATO (LOADED)

Esercizio: Struct

- Si definisca un tipo struct, per rappresentare i numeri complessi della forma a + b·i [SUGG: utilizzare due campi float uno per la componente reale e l'altro per la parte immaginaria] (**es.1**)
- Scrivere un programma che dichiara delle variabili del nuovo tipo, ne inizializza i valori e li stampa in output usando printf (**es.2**)
- Definire una funzione che valuti se due numeri complessi siano o meno uguali (**es.3**)
- Definire delle funzioni per le operazioni tra numeri complessi, quali somma, prodotto, coniugato, ecc. (**es.4**)

Es1:

```c
#include <stdio.h>
typedef struct {
    float real;   // parte reale
    float imag;   // parte immaginaria
} Complex;

Complex sommaRealeImmaginaria(float a, float b) {
    Complex z;
    z.real = a;      // la parte reale è a
    z.imag = b;      // la parte immaginaria è b
    return z;
}

int main() {
    Complex z = sommaRealeImmaginaria(3.5f, -2.0f);

    printf("z = %f + %fi\n", z.real, z.imag);
    return 0;
}
```

Output atteso:

```
z = 3.500000 + -2.000000i
```

Es2:

```c
#include <stdio.h>
// Definizione della struttura per i numeri complessi
typedef struct {
    float real;   // parte reale
    float imag;   // parte immaginaria
} Complex;

int main() {
    // Dichiarazione e inizializzazione di variabili Complex
    Complex z1 = {3.0f, 4.5f};
    Complex z2 = {-2.0f, 1.0f};
    Complex z3;

    // Inizializzazione manuale di z3
    z3.real = 0.0f;
    z3.imag = -7.2f;

    // Stampa dei valori
    printf("z1 = %f + %fi\n", z1.real, z1.imag);
    printf("z2 = %f + %fi\n", z2.real, z2.imag);
    printf("z3 = %f + %fi\n", z3.real, z3.imag);

    return 0;
}
```

Output atteso:

```
z2 = -2.000000 + 1.000000i
z3 = 0.000000 + -7.200000i
```

Es3:

```c
#include <stdio.h>
// Definizione della struttura per i numeri complessi
typedef struct {
    float real;
    float imag;
} Complex;

int complexEqual(Complex a, Complex b) {
    return (a.real == b.real) && (a.imag == b.imag);
}

int main() {
    Complex z1 = {3.0f, 4.0f};
    Complex z2 = {3.0f, 4.0f};
```

```
    Complex z3 = {3.0f, 4.1f};

    if (complexEqual(z1, z2))
        printf("z1 e z2 sono uguali\n");
    else
        printf("z1 e z2 NON sono uguali\n");

    if (complexEqual(z1, z3))
        printf("z1 e z3 sono uguali\n");
    else
        printf("z1 e z3 NON sono uguali\n");

    return 0;
}
```

Output atteso:

```
z1 e z2 sono uguali
z1 e z3 NON sono uguali
```

Es4:

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

// Definizione della struttura per i numeri complessi
typedef struct {
    float real;
    float imag;
} Complex;

// Somma: (a + bi) + (c + di)
Complex complexAdd(Complex x, Complex y) {
    Complex r;
    r.real = x.real + y.real;
    r.imag = x.imag + y.imag;
    return r;
}

// Sottrazione: (a + bi) - (c + di)
Complex complexSub(Complex x, Complex y) {
    Complex r;
    r.real = x.real - y.real;
    r.imag = x.imag - y.imag;
    return r;
}

// Prodotto: (a + bi)(c + di)
```

```c
Complex complexMul(Complex x, Complex y) {
    Complex r;
    r.real = x.real * y.real - x.imag * y.imag;
    r.imag = x.real * y.imag + x.imag * y.real;
    return r;
}

// Divisione: (a + bi) / (c + di)
Complex complexDiv(Complex x, Complex y) {
    Complex r;
    float denom = y.real * y.real + y.imag * y.imag;

    r.real = (x.real * y.real + x.imag * y.imag) / denom;
    r.imag = (x.imag * y.real - x.real * y.imag) / denom;

    return r;
}

// Coniugato: a + bi -> a - bi
Complex complexConjugate(Complex z) {
    Complex r;
    r.real = z.real;
    r.imag = -z.imag;
    return r;
}

// Modulo: sqrt(a^2 + b^2)
float complexMagnitude(Complex z) {
    return sqrtf(z.real * z.real + z.imag * z.imag);
}

// Stampa
void printComplex(Complex z) {
    printf("%f + %fi\n", z.real, z.imag);
}

// MAIN di test
int main() {
    Complex a = {3.0f, 4.0f};
    Complex b = {1.0f, -2.0f};

    printf("a = ");
    printComplex(a);

    printf("b = ");
    printComplex(b);

    printf("\nSomma: ");
    printComplex(complexAdd(a, b));

    printf("Sottrazione: ");
    printComplex(complexSub(a, b));

    printf("Prodotto: ");
```

```
    printComplex(complexMul(a, b));

    printf("Divisione: ");
    printComplex(complexDiv(a, b));

    printf("Coniugato di a: ");
    printComplex(complexConjugate(a));

    printf("Modulo di a: %f\n", complexMagnitude(a));

    return 0;
}
```

Output atteso:

```
a = 3.000000 + 4.000000i
b = 1.000000 + -2.000000i

Somma: 4.000000 + 2.000000i
Sottrazione: 2.000000 + 6.000000i
Prodotto: 11.000000 + -2.000000i
Divisione: -1.000000 + 2.000000i
Coniugato di a: 3.000000 + -4.000000i
Modulo di a: 5.000000
```

Esercizio: Scheduling CPU

Es.55:

	T.arrivo in ready	durata in burst
P1	0	11
P2	1	8
P3	3	5
P4	3	5

- Round Robin con tempo=4 ms
- context switch dura 1ms

La CPU al tempo zero c'è solo P1