PROGRAMMI BASE IN C

lezione 7 - slide

```c
#include <stdio.h>

int main() {
    printf("hello, world!\n");
    return 0; // non obbligatorio
}
```

- COSTRUTTO IF

si aspetta espressione controllo tipo intero, 0 è falso

```c
int main() {
    int x = 0;

    if (x)
        printf("x è falso\n");
    else
        printf("x è vero\n");
    return 0;
}
```

- COSTRUTTO WHILE

```c
int main() {
    float fahr = 0;
    printf("Tabella Fahrenheit-Celsius:\n");
    while (fahr <= 300) {
        float celsius = (5.0 / 9.0) * (fahr - 32.0);
        printf("%3.0f °F -> %6.1f °C\n", fahr, celsius);
        fahr = fahr + 20;
    }
    return 0;
}
```

- USO DELLE COSTANTI

```c
#define LOWER 0
#define UPPER 300
#define STEP 20

int main() {
    for (float fahr = LOWER; fahr <= UPPER; fahr = fahr + STEP)
        printf("%3.0f °F -> %6.1f °C\n", fahr, (5.0 / 9.0) * (fahr - 32));
    return 0;
}
```

- I/O CARATTERI, VERSIONE SEMPLIFICATA DEL cat

con ciclo while

```c
int main() {
    int c = getchar();

    while (c != EOF) {
        putchar(c);
        c = getchar();
    }
    return 0;
}
```

con ciclo for

```c
int main() {
    for (int c = getchar(); c != EOF; c = getchar()) {
        putchar(c);
    }
    return 0;
}
```

- COMANDO UNIX wc -c (conta i byte)

con ciclo while

```c
int main() {
    long nc = 0;

    while (getchar() != EOF) {
        ++nc;
    }
    printf("%ld\n", nc);
    return 0;
}
```

con ciclo for

```c
int main() {
    long nc = 0;

    for (; getchar() != EOF; ++nc);

    printf("%ld\n", nc);
    return 0;
}
```

- COMANDO UNIX wc -l (conta le linee)

```c
int main() {
    long nc = 0;

    for (int c = getchar(); c != EOF; c = getchar())
        if (c == '\n')
            ++nc;

    printf("%ld\n", nc);
    return 0;
}
```

lezione 7 - esercizi

1. Scrivere un programma C che stampi il valore della costante simbolica EOF

```c
#include <stdio.h>
int main(){
    int e = EOF;
    printf("EOF = %d\n", e);
    return 0;
}
```

2. scrivere un programma C che conti il numero di spazi, tab e newline (whitespace characters) rappresenti nei caratteri immessi sullo standard input

```c
#include <stdio.h>
int main() {
  int tab = 0, spazi = 0, newline = 0;
  for(int c = getchar(); c != EOF; c = getchar()) {
    switch(c) {
      case ' ':
        spazi++;
        break;
      case '\t':
        tab++;
        break;
      case '\n':
        newline++;
        break;
    }
  }
  printf("Numero di spazi: %d, numero di tab: %d, numero di newline: %d",spazi,tab,newline);
  return 0;
}
```

3. scrivere un programma che stampi un istogramma orizzontale (con '-') raffigurante le lunghezze delle parole (delimitate da whitespace characters) immesse sullo standard input (parola per parola)

```c
#include <stdio.h>

int main() {
  int i = 0, n = 0; // i è un indice, n è un contatore. n conta i caratteri, resettandosi a ogni spa

  for(int c = getchar(); c != EOF; c = getchar()) //per ogni carattere fino EOF
  {
    if(c != ' ' && c != '\t' && c != '\n') { // se il carattere non è uno spazio, un tab o un a capo
      n++; // incremento il contatore
    } else { // altrimenti

      for(i = 0; i < n; i++) // stampo tanti trattini fino a quanto è arrivato n
        printf("-");

      if(n > 0) // se n è maggiore di 0, ovvero ho contato uno spazio, un tab o un a capo
        printf("\n"); // vado a capo

      n = 0; // resetto il contatore
    } //dopo aver resettato conterò ancora una volta, se è EOF il ciclo terminerà
  }
  return 0;
}
```

3. scrivere un programma che conti il numero di parole sullo standard input

```c
#include <stdio.h>
//Scrivere un programma C che conti il numero di parole immesse sullo standard input
int main() {
  int n = 0, inizio_parola = 0; //

  for(int c = getchar(); c != EOF; c = getchar()) { //ciclo che si interrmpe con EOF

    if(c ==' ' || c == '\t' || c == '\n') //se c'è uno spazio
    {
      if(inizio_parola) { //se inizio_parola è 1
        n++; //incrementa n
        inizio_parola=0; //e metti inizio_parola a 0
      }
    } else
      inizio_parola=1; //altrimenti metti inizio_parola a 1
  } //fine for

  if(inizio_parola) //se inizio_parola è 1
    n++; //incrementa n

  printf("Numero di parole: %d\n",n);
  return 0;
}
```

lezione 8 - slide

- DICHIARAZIONE E DEFINIZIONE FUNZIONE

tipo_ritornato nome_funzione(lista_parametri);

```c
int factorial(int n);

int factorial(int n) {
    if (n == 0)
        return 1;
    else
        return n * factorial(n - 1);
}
```

- ESEMPIO FUNZIONE POWER

```c
int power(int, int); // dichiarazione della funzione power

int main() {
    int n;
    for (int i = 0; i < 10; i++)
        printf("%d %3d %6d\n", i, power(2, i), power(-3, i));
    return 0;
}

int power(int m, int n) {
    int p = 1;
    for (int i = 1; i <= n; i++)
        p = p * m;
    return p;
}
```

-FAKE SWAP (PASSAGGIO PER VALORE)

```c
void fake_swap(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}

int main() {
    int x = 42, y = 0;
    fake_swap(x, y);
    printf("x = %d, y = %d\n", x, y);
    return 0;
}
```

- MEDIA PESATA ARRAY

```c
#define N_ESAMI 12

int main() {
    int voti[N_ESAMI] = {18, 30, 27, 20, 22, 24, 25, 26, 28, 29, 30, 30};
    int crediti[N_ESAMI] = {12, 12, 12, 12, 6, 6, 6, 6, 6, 6, 6, 6};

    int somma = 0, totale = 0;

    for (int i = 0; i < N_ESAMI; i++) {
        somma += voti[i] * crediti[i];
        totale += crediti[i];
    }

    float media = (float)somma / totale;
    printf("La mia media pesata è: %2.2f\n", media);
    return 0;
}
```

- TABELLA VALORI SIN

```c
#include <math.h>

#define N_CAMPIONI 10

int main() {
    float valori[N_CAMPIONI] = {0.0};
    float step = 2 * M_PI / N_CAMPIONI; // M_PI è una costante definita in math.h, linker necessita

    for (int i = 0; i < N_CAMPIONI; i++) {
        valori[i] = sin(i * step);
    }

    for (int i = 0; i < N_CAMPIONI; i++) {
        printf("sin(%1.3f) = %1.3f\n", i * step, valori[i]);
    }

    return 0;
}
```

- CASO DI ERRORE INDICE

```c
int main() {
    int risposta = 42;
    int valori[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

    // Azzeriamo l'array
    for (int i = 0; i <= 10; i++) { // errore nel <=, stampa 0
        valori[i] = 0;
    }

    printf("La risposta è: %d\n", risposta);
    return 0;
}
```

lezione 8 - esercizi

1. ripetere esercizio 2 o 3 della lezione 7, definendo una funzione is_whitespace() per controllare se un carattere è uno spazio bianco oppure no (nota: non esiste il tipo bool)

```c

```

2. definire una funzione int lg(int n) che trovi il massimo numero m tale che 10^m sia minore o uguale a n (parte intera di logaritmo in base 10 di n)

```c

```

lezione 9 - slide

- PUNTATORI
- SWAP

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

int main() {
    int x = 42, y = 0;
    swap(&x, &y);

    printf("x: %d, y: %d\n", x, y);

    return 0;
}
```

- SCANF

```c
int main() {
    int x = 0;

    printf("Inserisci un numero intero: ");
    scanf("%d", &x);

    printf("Il doppio di %d è %d\n", x, x * 2);
    return 0;
}
```

- ESEMPIO

```c
#define SIZE 10

int main() {
    int array[SIZE] = {};
    int val = 0;

    printf("Inserisci un valore: ");
    scanf("%d", &val);

    for (int *p = array; p < array + SIZE; ++p) {
        *p = val;
    }

    for (int i = 0; i < SIZE; ++i) {
        printf("array[%d] = %d\n", i, array[i]);
    }
    return 0;
}
```

- ESEMPIO 2

```c
#define SIZE 10

void fill(int *, int, int);

int main(){
    int array[SIZE] = {}, val = 0;

    printf("inserisci un valore: ");
    scanf("%d",&val);
    for(int i = 0; i < SIZE; ++i)
        printf("array[%d] = %d\n", i, array[i]);

    return 0;
}

void fill(int *begin, int size, int value){
    for(int *p = begin; p < begin + size; ++p)
        *p = value;
}
```

lezione 9 - esercizi

1. scriviere un programma che legga dallo standard input dei numeri e ne stampi la somma totale

```c
#include <stdio.h>

int main()
{
  int n = 0, sum = 0;

  while(scanf("%d", &n) == 1) {
    sum += n;
  }

  printf("%d\n", sum);

  return 0;
}
```

2. scrivere una funzione reverse() che inverta l'ordine degli elementi in un array

```c

```

3.

lezione 10 - slide

- ARRAY DI CARATTERI (STRINGHE)

con scorrimento indicizzato

```c
#include<stdio.h>

int main(){
    char msg[] = "Ciao";
    printf("Stringa: \"%s\"\n", msg);
    printf("Caratteri: ");
    for(int i = 0; msg[i]; i++){
        printf("'%c' ", msg[i]);
    }
    putchar('\n');
    return 0;
}
```

con scorrimento a puntatori

```c
#include<stdio.h>

int main(){
    char msg[] = "Ciao";
    printf("Stringa: \"%s\"\n", msg);
    printf("Caratteri: ");
    for(char *p = msg; *p; p++){
        printf("'%c' ", *p);
    }
    putchar('\n');
    return 0;
}
```

- POSSIBILE IMPLEMENTAZIONE DI STRLEN

```c
int strlen(const char *s){
    int n = 0;
    for(char *p = s; *p; p++){
        n++;
    }
    return n;
}
```

- UTILIZZO DI STRNCMP()

```c
#include <stdio.h>
#include <string.h>
#define MAXINPUT 100

int main() {
    char input[MAXINPUT] = "";
    do {
        printf("Password: ");
        scanf("%99s", input);
    } while (strncmp(input, "passw0rd", MAXINPUT) != 0);
    printf("Il segreto è: 42\n");
    return 0;
}
```

- POSSIBILE IMPLEMENTAZIONE DI STRNCMP()

```c
char *strncpy(char *dest, char *source, unsigned len){
int i = 0;
while(i<len && source[i]){
    dest[i] = source[i];
    i++;
}
if (i < len){ // se la stringa source è più corta di len, aggiunge terminatore
    dest[i] = 0;
}
return dest;
}
```

- POSSIBILE IMPLEMENTAZIONE DI STRNCAT()

```c
char *strncat(char *dest, char *source, unsigned len){
    int destlen = strlen(dest);
    strncpy(dest + destlen, source, len);
    return dest;
}
```

- ESEMPIO DI USO DI STRNCPY() E STRNCAT()

```c
#include <stdio.h>
#include <string.h>

#define MAXLEN 20

int main(){
    char msg[MAXLEN] = "";
    char msg1[MAXLEN] = "Ciao,";
    char msg2[MAXLEN] = "mondo!";

    strncpy(msg, msg1, MAXLEN);
    strncat(msg, msg2, MAXLEN - strlen(msg)-1);
    printf("%s\n", msg);
    return 0;
}
```

- CODICE ERRATO

```c
#include <stdio.h>
#include <string.h>

int main(){
    char *ciao = "Ciao mondo !";
    strncpy(ciao, "Hello world!", 11);
    printf("%s\n", ciao);
    return 0;
}
```

- ESEMPIO DI ARGOMENTI DA RIGA DI COMANDO

```c
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv){
    int somma = 0;
    if(argc >=2 && strcmp(argv[1], "-s") == 0){
        somma = 1;
    }
    int x = 0, y = 0;
    printf("Inserisci due numeri: ");
    int n = scanf("%d %d", &x, &y);
    if(somma){
        printf("%d + %d = %d\n", x, y, x + y);
    } else {
        printf("%d * %d = %d\n", x, y, x * y);
    }
    return 0;
}
```

lezione 10 - esercizi

lezione 11 - slide

- ESEMPIO DI USO DI MALLOC

```c
#include <stdlib.h>

int somma(int *array, int size){
    int s=0;
    for(int i=0; i<size; i++) s+=array[i];
    return s;
}

int main(){
    int n = 0;

    printf("Quanti numeri verranno inseriti?");
    scanf("%d",&n);
    if(n == 0)
        return 0;

    int *elementi = malloc(n * sizeof(int));

    printf("Inserire i numeri: ");
    for(int i = 0; i < n; ++i) scanf("%d", elementi + i);
    printf("La somma dei numeri inseriti è : %d\n", somma(elementi, n));
    return 0;
}
```

- CREAZIONE DINAMICA DI DUE STRINGHE

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

char *concat(char *str1, char *str2){
    int len = strlen(str1) + strlen(str2);
    char *result = malloc(len + 1);

    strcpy(result, str1);
    strcat(result, str2);

    return result;
}
```

- LETTURA DI UN NUMERO ARBITRARIO DI VALORI

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

void reverse(int *array, int size);

int main(){
    int size = 10;
    int *array = malloc(size * sizeof(int));

    int read = 0;
    while (scanf("%d", &array[read]) == 1) {
        if(++read == size){
            size *= 2;
            array = realloc(array, size * sizeof(int));
        }
    }
    reverse(array, read);
    for(int i = 0; i < read; ++i){
        printf("%d\n", array[i]);
    }
    free(array);
    return 0;
}
```

- IMPLEMENTAZIONE CALLOC

```c
#include <stdlib.h>

void *calloc(unsigned count, unsigned size){
    unsigned len = count * size;
    char *mem = malloc(len);
    for (int i = 0; i < len; ++i){
        mem[i] = 0;
    }
    return mem;
}
```

- DICHIARAZIONE STRUTTURA

```c
struct point{
    float x;
    float y;
};
```

- LISTA CONCATENATA
- STRUTTURA NODO

```c
struct node{
    int data;
    struct node *next;
};
```

- NUOVO NODO

```c
struct node *create(int data){
    struct node *ptr = malloc(sizeof(struct node));
    ptr->data = data;
    ptr->next = NULL;

    return ptr;
}
```

- LUNGHEZZA

```c
int length(struct node *head){
    int len = 0;
    for (struct node *n = head; n; n = n->next){
        ++len;
    }
    return len;
}
```

- RICERCA ELEMENTO

```c
struct node *find(struct node *head, int data){
    for(struct node *n = head; n; n = n->next){
        if(n->data == data)
            return n;
    }
    return NULL;
}
```

- CONCATENAZIONE DUE LISTE

```c
struct node *last(struct node *head){
    for(struct node *n = head; n; n = n->next){
        if(n->next == NULL)
            return n;
    }
    return NULL;
}

struct node *append(struct node *head1, struct node *head2){
    struct node *last1 = last(head1);
    last1->next = head2;

    return head1;
}
```

- ELIMINAZIONE LISTA

```c
void destroy(struct node *head){
    struct node *next = head;

    while(next){
        struct node *n = next;
        next = n->next;
        free(n);
    }
}
```

- UTILIZZO DEI FILE HEADER:
- add.h

```c
int add(int x, int y);
```

- add.c

```c
#include "add.h"

int add(int x, int y){
    return x + y;
}
```

- main.c

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

int main(){
    printf("3+4 = %d\n", add(3,4));
    return 0;
}
```

- ESEMPIO ACCESSO AI FILE:

```c
#include <stdio.h>

int main(int argc, char **argv){
    if(argc < 2){
        fprintf(stderr, "fornire nome file\n");
        return 1;
    }

    char *filename = argv[1];

    FILE *file = fopen(filename, "r");
    if(!file){
        fprintf(stderr, "errore nell'apertura del file!\n");
        return 2;
    }

    int n = 0, sum = 0;
    while(fscanf(file, "%d", &n) == 1){
        sum += n;
    }

    if(!feof(file)){
        fprintf(stderr, "Il file non conteneva solo numeri.\n");
        return 3;
    }

    printf("Somma dei numeri contenuti: %d\n", sum);
    return 0;
}
```

- GESTIONE DEGLI ERRORI

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

int main(int argc, char **argv){
    if(argc < 2){
        fprintf(stderr, "specificare il nome di un file\n");
        return 1;
    }

    FILE *file = fopen(argv[1], "r");
    if (file == NULL){
        fprintf(stderr, "%s: Impossibile aprire %s: %s\n",
            argv[0], argv[1], strerror(errno));
        return 2;
    }
    return 0;
}
```

- CONOSCERE A PRIORI LA LUNGHEZZA DI UN FILE:

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

int main(int argc, char **argv){
    char *filename = argv[1];
    FILE *file = fopen(filename, "r");
    if(!file){
        fprintf(stderr, "%s: Impossibile aprire il file %s: %s\n",
            argv[0], filename, strerror(errno));
        return 2;
    }

    fseek(file, 0, SEEK_END);
    long bytes = ftell(file);

    printf("Il file è lungo %ld bytes\n", bytes);
    fclose(file);
    return 0;
}
```

- APRIRE UN FILE

```c
#include <unistd.h>
#include <fcntl.h>

int main(int argc, char **argv){
    int fd = open(argv[1], O_WRONLY | O_APPEND | O_CREAT, 0644);

    char contents[13] = "Hello World\n";

    write(fd, contents, 12);
    close(fd);
    return 0;
}
```

- USO DI STAT

(...)

- ESEMPIO DI UTILIZZO DI FORK

```c
#include <stdio.h>
#include <unistd.h>

int main(){
    printf("Un solo processo con PID %d.\n", (int)getpid());
    printf("Chiamata a fork...\n");

    pid_t pid = fork();

    if(pid == 0)
        printf("Sono il processo figlio (PID: %d).\n", (int)getpid());
    else if (pid > 0)
        printf("Sono il genitore del processo con PID %d.\n", pid);
    else
        fprintf(stderr, "Si è verificato un errore nella chiamata a fork.\n");

    return 0;
}
```

- ESEMPIO DI FUNZIONE EXECL()

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

int main(){
    printf("Esecuzione di ls...\n");

    execl("/bin/ls", "ls", "-l", NULL);

    perror("La chiamata di execl ha generato un errore");
    return 1;
}
```

- UTILIZZO COMBINATO DI FORK E EXECL

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main(){
    pid_t pid = fork();

    switch(pid){
        case -1:
            perror("fork() failed");
            return 1;
        case 0:
            printf("Esecuzione di ls...\n");
            execl("/bin/ls", "ls", "-l", NULL);

            perror("exec failed");
            return 1;
        default:
            wait(NULL);
            printf("ls completed\n");
            return 0;
    }
}
```

- ESEMPIO AMBIENTE DI UN PROCESSO:

```c
#include <stdio.h>

int main(int argc, char **argv, char **envp){
    while(*envp){
        printf("%s\n", *envp);
        ++envp;
    }
    return 0;
}
```

- ESEMPIO USO DI GETENV()

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

int main(int argc, char **argv, char **envp){
    if(argc <= 1){
        while(*envp){
            printf("%s\n", *envp);
            ++envp;
        }
    } else {
        printf("%s=%s\n", argv[1], getenv(argv[1]));
    }
    return 0;
}
```

- ESEMPIO USO DI EXECVE()

```c
#include <stdio.h>
#include <unistd.h>

int main(){
    char *argv[2] = {"env2", NULL};
    char *envp[3] = {"var1=valore1", "var2=valore2", NULL};

    execve("./env2", argv, envp);
    perror("execve fallita");

    return 1;
}
```

- ESEMPIO USER ID E GROUP ID

```c
#include <stdio.h>
#include <unistd.h>

int main(){
    uid_t uid = getuid();
    gid_t gid = getgid();

    printf("UID=%d, GID=%d\n", uid, gid);
    return 0;
}
```

- CREAZIONE DI UNA PIPE

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>

#define MSGSIZE 14

int main(){
    int pipes[2] = { };
    if (pipe(pipes) == -1){
        perror("pipe call");
        return 1;
    }

    char msg[MSGSIZE] = { };
    pid_t pid = fork();
    switch (pid){
        case -1:
            perror("fork call");
            return 2;
        case 0:
            close(pipes[0]);
            write(pipes[1], "Hello, world!", MSGSIZE);
            break;
        default:
            close(pipes[1]);
            read(pipes[0], msg, MSGSIZE);
            printf("%s\n", msg);
            wait(NULL);
    }
    return 0;
}
```

- COMANDO KILL

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <signal.h>

int main(int argc, char **argv){
    if (argc < 2){
        fprintf(stderr, "specificare il PID di un processo\n");
        return 1;
    }
    char *endptr = NULL;
    pid_t pid = strtoll(argv[1], &endptr, 10);
    if (*endptr != 0){
        fprintf(stderr, "specificare il PID di un processo\n" );
        return 1;
    }

    if(kill(pid, SIGKILL) == -1){
        fprintf(stderr, "impossibile uccidere il processo %d: %s\n", pid, strerror(errno));
        return 2;
    }
    return 0;
}
```

- ESEMPIO DI SIGNAL HANDLING

```c
#include <stdio.h>
#include <unistd.h>
#include <signal.h>

void ahah(int x){
    printf("ahah you cannot terminate me!\n");
}

int main(){
    signal(SIGINT, ahah);
    signal(SIGTERM, ahah);
    signal(SIGKILL, ahah);

    while(1){
        printf("Hey apple!\n");
        sleep(1);
    }
    return 0;
}
```

- UTILIZZO DEI SOCKET CON FUNZIONI BIND() E LISTEN()

```c
#include <sys/types.h>
#include <sys/socket.h>

struct sockaddr_un{
    short sa_family;
    char sun_path[108];
};

int bind(int sockfd, const struct sockaddr *addr, size_t addr_len);
int listen(int sockfd, int queue_size);
```

- ESEMPIO

```c
#include <stdio.h>
#include <pthread.h>

void *print_msg(void *ptr);

int main(){
    char msg1[] = "Thread 1";
    char msg2[] = "Thread 2";

    pthread_t thread1, thread2;

    pthread_create(&thread1, NULL, print_msg, (void *)msg1);
    pthread_create(&thread2, NULL, print_msg, (void *)msg2);

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    return 0;
}

void *print_msg(void *ptr){
    char *arg = (char *) ptr;

    while(1)
        printf("%s\n", arg);

    return NULL;
}
```

- ESEMPIO DI SINCRONIZZAZIONE DELL'ACCESSO A UN CONTATORE

```c
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>

int n = 0;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

void *count(void *);

int main(){
    pthread_t th1, th2;
    pthread_create(&th1, NULL, count, NULL);
    pthread_create(&th2, NULL, count, NULL);

    pthread_join(th1, NULL);
    pthread_join(th2, NULL);

    printf("n: %d\n", n);
    return 0;
}

void *count(void *arg){
    int local_n = 0;
    do {
        usleep(50000);
        pthread_mutex_lock(&mutex);
        n += 1;
        local_n = n;
        pthread_mutex_unlock(&mutex);
        printf("n: %d\n", local_n);
    } while (local_n < 42);
    return NULL;
}
```