

# Esoneri Svolti – Programmazione 2 (C)

Autore: **Scarlata Davide**

Anno: **Primo anno**

Corso: **Linguaggio C – Programmazione 2**


## ⚠️ Nota bene:

Le soluzioni non sono garantite corrette: **sono esercizi svolti da me** (studente) e **potrebbero contenere errori**.

Questo materiale è pensato come supporto allo studio e confronto personale, **non come riferimento ufficiale**.

---

## Collegamenti utili

-  [Moodle corso C](#)
-  [Piattaforma esami](#)
- 

---

## Come usare questa raccolta

- Puoi usare questi esercizi per **ripassare la teoria** attraverso il codice.
- Prova a **risolvere prima da solo** ogni esercizio, poi confronta con la mia soluzione.
- Se trovi un errore o vuoi suggerire una miglioria, **scrivimi** e ti fornirò il codice Markdown per effettuare un *push* della modifica.

---

## Ringraziamenti

Un ringraziamento speciale a **edo.js** e a chi ha collaborato alla creazione della "**Bibbia**" di **Programmazione 2**:

una risorsa fondamentale per comprendere e affrontare al meglio questo corso.

---



## Se vuoi offrirmi un caffè

- [paypal](#)
- [revolut](#)



## SIMULAZIONE

### Funzione `extract`

```
/**
@brief Estrae e restituisce una nuova stringa contenente i caratteri da s[j] a
s[i], in ordine inverso. Se i > j, restituisce stringa vuota.
*
* @param s La stringa di partenza.
* @param i L'indice iniziale.
* @param j L'indice finale .
* @return Una nuova stringa con i caratteri estratti in ordine inverso, o
una stringa vuota se i > j.
*/
char *extract(char *s, int i, int j);
```

## SOLUZIONE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char *extract(char *s, int i, int j) {
    if (i > j) {
        // restituisce stringa vuota se i > j
        char *empty = malloc(1);
        if (empty != NULL){
            empty[0] = '\0';
        }
        return empty;
    }

    int len = j - i + 1;
    char *res = malloc(len + 1); // +1 per il terminatore
    if (res == NULL){
        return NULL;
    }
}
```

```

    for (int k = 0; k < len; k++) {
        res[k] = s[j - k];
    }
    res[len] = '\0';
    return res;
}

int main() {
    char *str = "eccocosafare";
    int first = 4;
    int last = 7;
    char *result = extract(str, first, last);
    if (result != NULL) {
        printf("Risultato: %s\n", result); // atteso: "asoc"
        free(result); // libera memoria allocata
    }
    return 0;
}

```

# PRIMO APPELLO - 24/06/2024

## Esonero 1

```

/**
 * @brief Restituisce una nuova stringa ottenuta da `ac`:
 * - rimuovendo le cifre ('0'..'9')
 * - convertendo le minuscole in maiuscole
 * - riempiendo con '#' fino a lunghezza n
 */
char *change(char *ac, int n);

```

## soluzione

utilizzando i comandi delle librerie `stdlib.h` e `ctype.h` per la gestione della memoria e dei caratteri rispettivamente, la funzione `change` può essere implementata come segue:

```

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
char *change(char *ac, int n) {
    char *res = malloc(n + 1); // +1 per '\0'
    if (res == NULL) return NULL;

    int j = 0; // indice per il nuovo array

```

```

for (int i = 0; i < n; i++) {
    if (isdigit(ac[i])) {
        continue; // salta cifre
    }

    if (islower(ac[i])) {
        res[j++] = toupper(ac[i]);
    } else {
        res[j++] = ac[i];
    }
}

// Riempi il resto con '#'
while (j < n) {
    res[j++] = '#';
}

res[n] = '\0'; // chiusura stringa
return res;
}

int main() {
    char original[] = "a1b2c3XyZ";
    int len = sizeof(original) - 1; // esclude '\0'

    char *trasformata = change(original, len);
    if (trasformata != NULL) {
        printf("Risultato: %s\n", trasformata); // atteso: "ABCXYZ###"
        free(trasformata);
    }

    return 0;
}

```

senza utilizzare le libreria `ctype.h`, la funzione `change` può essere implementata come segue: (soluzione di un altro ragazzo non fatta da me )

```

char *change(char *ac, int n){
    //controllo se array passato vuoto
    if(ac == NULL)
        return NULL;
    //crea nuovo array
    char* newArray = (char*)malloc(sizeof(char) * n);
    //controlla se anche questo è vuoto
    if(newArray == NULL)

```

```

        return NULL;
    size_t i=0;
    size_t j=0;
    while (ac[i] != '\0' && i<n){
        //controlla se il carattere è un numero
        if(ac[i]== '1' || ac[i]== '2' || ac[i]== '3' || ac[i]== '4'
        || ac[i]== '5' || ac[i]== '6' || ac[i]== '7' || ac[i]== '8'
        || ac[i]== '9'){
            i++;
        }
        else{
            //controlla se il carattere è 'k' o 'h'
            if(ac[i]== 'h' || ac[i]== 'k'){
                if(ac[i]== 'h')
                {
                    newArray[j] = 'H';
                    i++;
                    j++;
                }
                else if(ac[i]== 'k')
                {
                    newArray[j] = 'K';
                    i++;
                    j++;
                }
            }
            //se non è numero, non è 'k' o 'h', mette la lettera puntata
            else{
                newArray[j] = ac[i];
                i++;
                j++;
            }
        }
    }
    //riempie gli spazi restanti con '#'
    while(newArray[j] == '\0' && j<n){

        newArray[j] = '#';
        j++;
    }
    return newArray;
}

#define N 12
int main(){

```

```

char ac[N] = {'a', '1', 'b', '2', 'c', '3', 'd', '4', 'k', '5', 'h', '\0'};

char* newAc = change(ac, N);
}

```

## esonero 2

```

/*
crea una funzione in c che, data una stringa, crea un array dove toglie i
numeri dall'array che le viene passato
e li sostituisce con le lettere maiuscole 'A' 'B', ovvero:
ciao1c2a4o2 diventa ciaoAcBaAoB (sostituisce consecutivamente i numeri con A e
B)
crea un main con una stringa che richiami una funzione per risolvere il
problema, la quale dovrà
usare la malloc. La funzione chiamata avrà come input un puntatore all'array e
il numero di elementi.
*/
char* sostituisciNumeri(const char* str, int lunghezza);

```

## soluzione

```

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
char* sostituisciNumeri(const char* str, int lunghezza) {
    char* risultato = malloc(lunghezza + 1);
    if (risultato == NULL) {
        return NULL; // errore allocazione
    }

    char lettera = 'A'; // prima lettera maiuscola da usare

    for (int i = 0; i < lunghezza; i++) {
        if (isdigit((unsigned char)str[i])) {
            risultato[i] = lettera++;
            if (lettera > 'Z') { // ricomincia da 'A' se supera 'Z'
                lettera = 'A';
            }
        } else {
            risultato[i] = str[i];
        }
    }
}

```

```

    risultato[lunghezza] = '\0'; // terminatore stringa
    return risultato;
}
int main() {
    char stringa[] = "ciao1c2a4o2";
    int len = sizeof(stringa) - 1; // esclude '\0'

    char *modificata = sostituisciNumeri(stringa, len);
    if (modificata != NULL) {
        printf("Stringa originale: %s\n", stringa);
        printf("Stringa modificata: %s\n", modificata);
        free(modificata);
    } else {
        printf("Errore allocazione memoria\n");
    }

    return 0;
}

```

## esonero 3

```

/**
 * @brief Duplica ogni cifra nell'array di caratteri passato come input.
 * Alloca dinamicamente e restituisce la nuova stringa.
 *
 * @param str array di caratteri di input
 * @param lunghezza lunghezza dell'array
 * @return char* nuova stringa (allocata dinamicamente), oppure NULL in caso
di errore
 */
char* duplicaNumeri(const char* str, int lunghezza);

```

## soluzione

```

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
char* duplicaNumeri(const char* str, int lunghezza) {
    // Calcolo la lunghezza necessaria per la nuova stringa
    int new_len = 0;
    for (int i = 0; i < lunghezza; i++) {
        if (isdigit((unsigned char)str[i])) {
            new_len += 2; // duplico la cifra
        } else {

```

```

        new_len += 1;
    }
}
// Alloco la nuova stringa
char* risultato = malloc(new_len + 1); // +1 per terminatore
if (!risultato) return NULL; // errore allocazione

// Costruisco la nuova stringa duplicando le cifre
int j = 0;
for (int i = 0; i < lunghezza; i++) {
    if (isdigit((unsigned char)str[i])) {
        risultato[j++] = str[i];
        risultato[j++] = str[i];
    } else {
        risultato[j++] = str[i];
    }
}

risultato[j] = '\0'; // terminatore
return risultato;
}

int main() {
    char input[] = "abc1c";
    int len = sizeof(input) - 1; // esclude '\0'

    char* output = duplicaNumeri(input, len);
    if (output != NULL) {
        printf("Originale: %s\n", input);
        printf("Modificata: %s\n", output);
        free(output);
    } else {
        printf("Errore allocazione memoria.\n");
    }
    return 0;
}

```

## secondo appello 24/07/2024

### esonero 2

```

/**
 * @brief Data una stringa str di lunghezza n e un numero m >= 0,
 * restituisce una nuova stringa di lunghezza m (o meno se m > n),
 * con i caratteri di str presi dagli ultimi m caratteri ma in ordine
 * inverso.

```



```
* Se m == 0 o n == 0, restituisce stringa vuota.  
* **/  
char* trasforma(char* str, int n, int m);
```

## soluzione

```
#include <stdlib.h>  
#include <stdio.h>  
char* trasforma(char* str, int n, int m) {  
    if (m == 0 || n == 0) {  
        // restituisco stringa vuota allocata  
        char* vuota = malloc(1);  
        if (vuota) vuota[0] = '\0';  
        return vuota;  
    }  
  
    int len = (m < n) ? m : n; // la lunghezza effettiva da copiare  
  
    char* result = malloc(len + 1);  
    if (!result) return NULL; // errore allocazione  
  
    // riempio result con i caratteri presi da str, dall'ultimo verso i precedenti  
    // posizione 0 di result = str[n-1]  
    for (int i = 0; i < len; i++) {  
        result[i] = str[n - 1 - i];  
    }  
  
    result[len] = '\0';  
    return result;  
}  
int main() {  
    char s[] = "eccocosafare";  
    int n = 12;  
    int m = 5;  
  
    char* res = trasforma(s, n, m);  
    if (res) {  
        printf("Risultato: \"%s\"\n", res); // output atteso: "erafa"  
        free(res);  
    }  
  
    return 0;  
}
```

# terzo appello 24/09/2024

## esonero 1

```
/**
 * @brief Combina due array a e b di lunghezza n in un nuovo array di
 * lunghezza 2*n,
 * alternando elementi da a e b come descritto:
 * a[0], b[n-1], a[1], b[n-2], ..., a[n-1], b[0].
 */
int *combine(int *a, int *b, int n);
```

## soluzione

```
#include <stdio.h>
#include <stdlib.h>
int *combine(int *a, int *b, int n) {
    int *result = malloc(sizeof(int) * 2 * n);
    if (!result){
        return NULL; // errore allocazione
    }
    for (int i = 0; i < n; i++) {
        result[2*i] = a[i];
        result[2*i + 1] = b[n - 1 - i];
    }
    return result;
}
int main() {
    int a[] = {1, 2, 3, 4};
    int b[] = {6, 7, 8, 9};
    int n = 4;

    int *res = combine(a, b, n);
    if (res != NULL) {
        for (int i = 0; i < 2*n; i++) {
            printf("%d ", res[i]);
        }
        printf("\n");
        free(res);
    } else {
        printf("Errore allocazione memoria\n");
    }
}
```

```
    return 0;  
}
```