Laboratorio 2 A

***ESERCITAZIONE:* Preparazione al II compitino.**

Prof. Patrizio Dazzi • a.y. 2025/26

https://pages.di.unipi.it/dazzi/

patrizio.dazzi@unipi.it

Informazioni sul compitino "prova in itinere"

Compitino 2 - Informazioni e Modalità

- Martedì 25, in aula, carta e penna;
- Registrazione obbligatoria, entro lunedì alle 14:00[a]
- È consentita la consultazione delle slide;

[a] https://forms.office.com/e/B857Dz2JjN

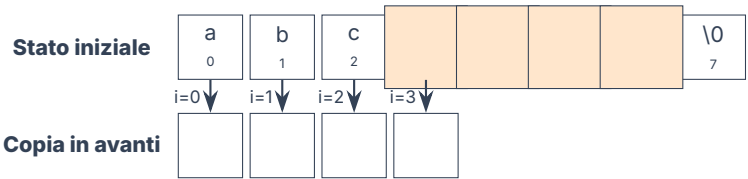
Puntatori in C

Obiettivi dell'esercizio

Puntatori – Implementare `mymemmove` una versione minimale di `memmove(void *dst, const void *src, size_t n)` che gestisce correttamente il caso di aree sovrapposte. Scrivere un piccolo `main` che:

- allochi dinamicamente un buffer, lo inizializzi con una stringa,
- applichi `mymemmove` con regioni parzialmente sovrapposte,
- mostri il risultato prima/dopo.

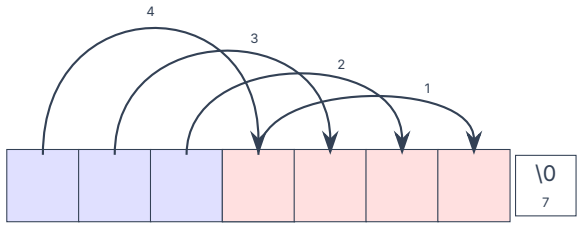
Non copiare in avanti quando le aree si sovrappongono



Nei passi `i=0..2` scrivo in `dst[0..2] = buf[3..5]` lasciando intatti `src[1..3]`. Al passo `i=3`, però, leggo `src[3] = buf[3]`, che è già stato sovrascritto al passo `i=0` con `'a'`. Risultato: in `buf[6]` finisce `'a'`, non `'d'`.

Morale: con sovrapposizione e `dst` > `src` serve copiare *all'indietro* (come fa `memmove`).

`mymemmove(buf+3, buf, 4)`: overlap e direzione di copia



SRC = buf[0..3] **DST** = buf[3..6]

Con $n = 4$**:** si copiano i byte `a b c d`.

Stato finale: `[0..7] = a b c a b c d \0` $\Rightarrow$ stringa `"abcabcd"`.

Se vuoi includere anche il terminatore, usa $n = 5$ (o `strlen(src)+1`).

blocco sorgente

blocco destinazione

ordine delle copie: dall'ultimo byte al primo

Possibile Soluzione (mymemmove)

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

void *mymemmove
      (void *dst, void *src, size_t n) {
    char *d = (char*)dst;
    char *s = (char*)src;
    if (d == s || n == 0) return dst;
    if (d < s || d >= s + n) {
      for (size_t i = 0; i < n; ++i)
        d[i] = s[i];
    } else {
      for (size_t i = n; i > 0; --i)
        d[i-1] = s[i-1];
    }
    return dst;
}
```

```c
int main(void) {
    char *msg = "abcdefg";
    size_t len = strlen(msg) + 1;
    char *buf = malloc(len);
    if (!buf){ perror("malloc"); return 1;}

    memcpy(buf, msg, len);
    printf("prima: %s\n", buf);

    mymemmove(buf+3, buf, 4); // copia 'a
    ','b','c','\0'

    printf("dopo:  %s\n", buf);
    free(buf);
    return 0;
}
```

Stringhe in C

Obiettivi dell'esercizio

Stringhe – Implementare `normalize(char *s)` che:

- rimuove spazi iniziali/finali,
- compatta sequenze di spazi interni in un singolo spazio,
- converte le lettere in minuscolo (locale ASCII),
- opera in-place con soli puntatori (senza array ausiliari grandi).

Scrivere un test che legga una riga con `fgets` e stampi prima/dopo.

Possibile Soluzione (normalize)

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

void normalize(char *s) {
    // Trim left
    char *p = s;
    while (*p && isspace((unsigned char)*p)) p++;
    char *w = s; int in_sp = 0;
    for (; *p; ++p) {
        unsigned char c = (unsigned char)*p;
        if (isspace(c)) { in_sp = 1; }
        else {
            if (in_sp && w != s) *w++=' ';
            *w++ = (char)tolower(c);
            in_sp = 0;
        }
    }
    // Rimuovi eventuale spazio finale
    if (w > s && *(w-1) == ' ') --w;
    *w = '\0';
}
```

```c
int main(void) {
    char buf[256];
    if (!fgets(buf, sizeof(buf), stdin)) return 0;

    // rimuovi eventuale '\n' finale
    char *nl = buf; while (*nl) { if (*nl=='\n'){*nl='\0';break;} nl++; }
    printf("prima:  '%s'\n", buf);
    normalize(buf);
    printf("dopo:   '%s'\n", buf);
    return 0;
}
```

File e Directory in C

Obiettivi dell'esercizio

File/Directory – Scrivere un programma che legge un file di testo riga per riga e ne produce una copia su un nuovo file, numerando le righe.

- Usare solo funzioni standard della libreria C: `fopen`, `fgets`, `fprintf`, `fclose`.
- Il programma deve ricevere due argomenti: file di input e file di output.
- Ogni riga dell'output deve avere il formato: `<numero>: <contenuto>`.

Possibile Soluzione (file numerati)

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

int main(int argc, char *argv[]) {
    if (argc != 3) {
    fprintf(stderr, "Uso: %s <input> <output>\n", argv[0]);
    return 1;
    }


    FILE *in = fopen(argv[1], "r");
    if (!in) { perror("Errore apertura input"); return 1; }
    FILE *out = fopen(argv[2], "w");
    if (!out) { perror("Errore apertura output"); fclose(in); return 1; }


    char line[1024];
    int num = 1;
    while (fgets(line, sizeof(line), in)) {
    fprintf(out, "%4d: %s", num++, line);
    }


    fclose(in);
    fclose(out);
    printf("Copiato %d righe in %s\n", num - 1, argv[2]);
    return 0;
}
```

Strutture dati: Liste

Obiettivi dell'esercizio

Liste – Implementare una lista semplicemente concatenata di interi con le seguenti operazioni:

- `push_front`, `print`, `reverse`,
- `remove_all(head, x)` che rimuove tutte le occorrenze di *x*,
- test: leggi interi da `stdin` fino a EOF, stampa, rimuovi *x* passato su riga di comando, stampa, fai `reverse`, stampa.

Possibile Soluzione (liste)

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int v; struct Node *next;
} Node;
void push_front(Node **head, int v){
    Node *n = malloc(sizeof(*n));
    if(!n){perror("malloc"); exit(1);}
    n->v=v; n->next=*head; *head=n;
}
void print(const Node *h){
    for(;h;h=h->next) printf("%d ", h->v);
    printf("\n");
}
void reverse(Node **head){
    Node *prev=NULL, *cur=*head, *nx;
    while(cur){ nx=cur->next; cur->next=prev; prev=cur; cur
    =nx; }
    *head=prev;
}
```

```c
void remove_all(Node **head, int x){
    Node **pp = head;
    while(*pp){
        if((*pp)->v == x){
            Node *tmp = *pp; *pp = (*pp)->next; free(tmp);
        } else {
            pp = &((*pp)->next);
        }
    }
}

void free_all(Node *h){
    while(h){ Node *t=h; h=h->next; free(t);} }

int main(int argc, char **argv){
    if(argc!=2){fprintf(stderr,"Uso: %s <x>\n",argv[0]);
    return 1;}
    int x = atoi(argv[1]);
    Node *head=NULL; int val;
    while (scanf("%d", &val)==1) push_front(&head, val);
    print(head);
    remove_all(&head, x); print(head);
    reverse(&head); print(head);
    free_all(head); return 0;
}
```

Q & A

Thank you for your attention!
Any Questions?

patrizio.dazzi@unipi.it