

IEIM 2018-2019

Esercitazione IX ***“Ripasso Generale”***

Alessandro A. Nacci

alessandro.nacci@polimi.it - www.alessandronacci.it

- Immagini
- Mercato
- Tris 1
- Tris 2
- Ricorsione



Es 0 - Immagini BN

- Si scriva un programma che sappia gestire una immagine con una risoluzione di 65536 pixels
- L'immagine è in scala di grigi: deve saper gestire 256 differenti tonalità di grigio.
- Il programma deve avere una funziona che crea una immagine completamente bianca
- Il programma deve avere una funziona che crea una immagine completamente nera



Es 0 - Immagini BN

- Si scriva un programma che sappia gestire una immagine con una risoluzione di 65536 pixels
- L'immagine è in scala di grigi: deve saper gestire 256 differenti tonalità di grigio.
- Il programma deve avere una funziona che crea una immagine completamente bianca
- Il programma deve avere una funziona che crea una immagine completamente nera



Es 0 - Immagini BN

```
int main()
{
    char image[256][256];
}
```

```
void makeBlack(char image[256][256])
{
    int i, j;
    for (i=0; i<256; i++)
    {
        for (j=0; j<256; j++)
        {
            image[i][j] = 0;
        }
    }
}
```

```
void makeWhite(char image[256][256])
{
    int i, j;
    for (i=0; i<256; i++)
    {
        for (j=0; j<256; j++)
        {
            image[i][j] = 255;
        }
    }
}
```



Es 0 - Immagini Colori

- Si scriva un programma che sappia gestire una immagine con una risoluzione di 65536 pixels
- L'immagine è a colori: deve saper gestire 256 differenti tonalità per ogni colore nello spazio RGB (Red, Green, Blue)
- Il programma deve avere una funziona che crea una immagine completamente bianca
- Il programma deve avere una funziona che crea una immagine completamente nera



Es 0 - Immagini Colori

```
typedef struct
{
    char R;
    char G;
    char B;
} t_pixel;

int main()
{
    t_pixel image[256][256];
}
```

```
void makeWhite(t_pixel image[256][256])
{
    int i, j;

    for (i=0; i<256; i++)
    {
        for (j=0; j<256; j++)
        {
            image[i][j].R = 255;
            image[i][j].G = 255;
            image[i][j].B = 255;
        }
    }
}
```

```
void makeBlack(t_pixel image[256][256])
{
    int i, j;

    for (i=0; i<256; i++)
    {
        for (j=0; j<256; j++)
        {
            image[i][j].R = 0;
            image[i][j].G = 0;
            image[i][j].B = 0;
        }
    }
}
```




Es 0 - Immagini Colori

Colour picker

#4286f4
rgb(66, 134, 244)

HEX #4286f4
RGB rgb(66, 134, 244)
HSV hsv(217°, 73%, 96%)
HSL hsl(217°, 90%, 61%)
CMYK cmyk(73%, 45%, 0%, 4%)

^ Show less

Feedback

Colour picker

#4ef442
rgb(78, 244, 66)

HEX #4ef442
RGB rgb(78, 244, 66)
HSV hsv(116°, 73%, 96%)
HSL hsl(116°, 90%, 61%)
CMYK cmyk(68%, 0%, 73%, 4%)

^ Show less

Feedback

Colour picker

#000000
rgb(0, 0, 0)

HEX #000000
RGB rgb(0, 0, 0)
HSV hsv(276°, 100%, 0%)
HSL hsl(276°, 0%, 0%)
CMYK cmyk(0%, 0%, 0%, 100%)

^ Show less

Feedback

<https://www.google.com/search?q=color+picker&oq=color+picker>

- Si scriva un programma per la gestione del carrello di un supermercato.
- Si definisca una struttura dati per rappresentare un prodotto acquistabile; tale struttura deve contenere:
 - Codice prodotto
 - Nome
 - Prezzo
- Si definisca una struttura dati per rappresentare il carrello:
 - Lista di prodotti
 - Numero di prodotti acquistati
 - Totale da pagare
- Viene chiesto all'utente di inserire i prodotti che devono essere acquistati. Tali prodotti vengono inseriti nel carrello e le informazioni del carrello vengono aggiornate. Alla fine si procede alla stampa dello scontrino e del totale da pagare.
- Vincolo:
 - Si strutturi il programma dividendolo in sottofunzioni ove possibile. In particolare si creino apposite funzioni per:
 - Leggere un prodotto acquistato
 - Inserire il prodotto nel carrello
 - Calcolare il totale da pagare
 - Stampare lo scontrino



Es I - Mercato - Strutture Dati

```
#include <stdio.h>

#define MAX_PRODOTTI 10
#define MAX_NOME 50

typedef struct{
    int codice;
    char nome[MAX_NOME];
    float prezzo;
} prodotto;

typedef struct{
    prodotto prodotti[MAX_PRODOTTI];
    int numeroProdotti;
    float totale;
} carrello;
```



Es I - Mercato - Prototipi Funzioni

```
prodotto leggiProdotto();  
void inserisciProdotto(carrello *c, prodotto p);  
void calcolaTotale(carrello *c);  
void stampaScontrino(carrello *c);
```

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

```
#define MAX_PRODOTTI 10
```

```
#define MAX_NOME 50
```

```
typedef struct{  
    int codice;  
    char nome[MAX_NOME];  
    float prezzo;  
} prodotto;
```

```
typedef struct{  
    prodotto prodotti[MAX_PRODOTTI];  
    int numeroProdotti;  
    float totale;  
} carrello;
```



Es I - Mercato - Funzioni

```
prodotto leggiProdotto() {  
    prodotto p;  
  
    printf( "Codice: " );  
    scanf( "%d", &p.codice );  
    printf( "Nome: " );  
    scanf( "%s", p.nome );  
    printf( "Prezzo: " );  
    scanf( "%f", &p.prezzo );  
  
    return p;  
}
```



Es I - Mercato - Funzioni

```
void inserisciProdotto(carrello *c, prodotto p){  
    c->prodotti[c->numeroProdotti]=p;  
    c->numeroProdotti+=1;  
}
```



Es I - Mercato - Funzioni

```
void calcolaTotale(carrello *c){  
    int i;  
  
    c->totale=0;  
  
    for(i=0;i<c->numeroProdotti;i++)  
        c->totale+=c->prodotti[i].prezzo;  
}
```



Es I - Mercato - Funzioni

```
void stampaScontrino(carrello *c){  
    int i;  
  
    for(i=0;i<c->numeroProdotti;i++)  
        printf("%d\t%s\t%f\n",c->prodotti[i].codice, c->prodotti[i].nome, c->prodotti[i].prezzo);  
  
    printf("\n\tTOTALE:\t%f\n",c->totale);  
}
```




Es I - Mercato - Main

```
int main(){
    carrello c;
    prodotto p;

    c.numeroProdotti=0;

    int i;
    char s;

    do{
        printf("Vuoi comprare un prodotto (s/n)? ");
        scanf(" %c",&s);

        if(s=='s' || s=='S'){
            p=leggiProdotto();
            inserisciProdotto(&c,p);
        }
    }while((s=='s' || s=='S') && c.numeroProdotti<MAX_PRODOTTI);

    calcolaTotale(&c);
    stampaScontrino(&c);

    return 0;
}
```


L'impiccato

Esercizio 3

Il gioco dell'impiccato

- Scrivere un programma che permetta di giocare al gioco dell'impiccato



A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

```
bash
es5 — bash — 106x27

c o _ _ _ _ _
Inserisci una lettera
m
Hai a disposizione ancora 8 tentativi
c o m _ _ _ _
Inserisci una lettera
r
Hai a disposizione ancora 8 tentativi
c o m _ _ _ r
Inserisci una lettera
t
Hai a disposizione ancora 8 tentativi
c o m _ _ t _ r
Inserisci una lettera
p
Hai a disposizione ancora 8 tentativi
c o m p _ t _ r
Inserisci una lettera
u
Hai a disposizione ancora 8 tentativi
c o m p u t _ r
Inserisci una lettera
e
Hai a disposizione ancora 8 tentativi
c o m p u t e r
Complimenti!
MacBook-Pro-di-Alessandro-Nacci:es5 alessandronacci$
```



Vediamo cosa ci serve...

- L'idea è quella di avere
 - un dizionario di parole da indovinare
 - un numero massimo di tentativi
 - lo stato delle parola (quali lettere sono state indovinate?)

Dichiarazione ed inizializzazione variabili

```
//Dizionario di parole
char dizionario[5][20]={"computer", "proiettore", "aula", "ciao", "telecomando"};

int i;
int tentativi=0;
int found=0;

//Puntatore alla parola da indovinare, scelta casualmente
char* parola_da_indovinare;
parola_da_indovinare = dizionario[scegliParola()];

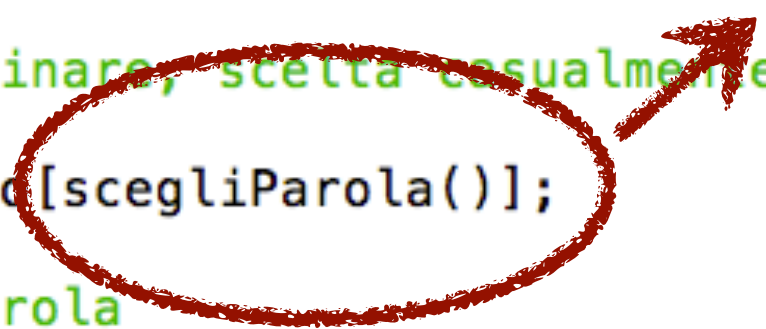
//Campo di gioco, stato della parola
char stato_parola[20];

//Lunghezza della parola da indovinare
int len = strlen(parola_da_indovinare);

char lettera;

for(i=0;i<len;i++)
    stato_parola[i]='_';
```

come la implemento?



```
//Inizializzo il campo da gioco con '_'
```



Scegli parola

```
int scegliParola(){  
    int num;  
    srand(time(0));  
    //Estraggo un numero tra 0 e "numero di parole del dizionario"  
    num=rand()%5;  
    return num;  
}
```



Ragioniamo sulle funzioni necessarie...

- Ora ci servono ancora alcune funzionalità:
 - Controllare se una data lettera è corretta o meno
 - Sostituire le lettere indovinate al posto dei trattini ‘_’
 - Stampare a schermo lo stato attuale della parola
 - Controllare se il giocatore ha vinto



Controllo e sostituzione lettera

```
int selezionaLettere(char * parola_da_indovinare,
                    char* stato_parola, char lettera)
{

    int i;
    int count=0;

    //Segno nel campo da gioco, tutte le posizioni
    //corrispondenti alla lettera scelta dall'utente
    for(i=0;i<strlen(parola_da_indovinare);i++)
        if(parola_da_indovinare[i]==lettera)
        {
            stato_parola[i]=lettera;
            count++;
        }

    if(count==0)
        return 0;
    else
        return 1;

}
```


Controllo vincita & stampa a schermo

```
//Controllo se la parola è stata completata
```

```
int check(char* stato_parola, int len){
```

```
    int i;
```

```
    for(i=0;i<len;i++)
```

```
        if(stato_parola[i]=='_')
```

```
            return 0;
```

```
    return 1;
```

```
}
```

```
void stampa(char* parola, int len, int tentativi){
```

```
    int i;
```

```
    printf("Hai a disposizione ancora %d tentativi\n",  
          TENTATIVI - tentativi);
```

```
    for(i=0;i<len;i++)
```

```
        printf("%c ", parola[i]);
```

```
    printf("\n");
```

```
}
```




Gestione del gioco

```
while(!found && tentativi<TENTATIVI){  
    //Chiedo lettere, finchè non indovina la parola o esaurisce i tentativi  
  
    stampa(stato_parola, len, tentativi);  
    printf("Inserisci una lettera\n");  
    scanf("%c", &lettera);  
    getchar();  
  
    int result = selezionaLettere(parola_da_indovinare, stato_parola, lettera);  
    if(result==0)  
        tentativi++;  
    else if(check(stato_parola, len))  
        found=1;  
}  
  
stampa(stato_parola, len, tentativi);  
if(found==1)  
    printf("Complimenti!\n");  
else  
    printf("Hai superato i tentativi a disposizione\n");  
  
return 0;
```

**Tutte il materiale sarà
disponibile sul mio sito
internet!**

alessandronacci.it

See You Next Time!

