Esercizi su funzioni e procedure

Esercizio 1

Dati due numeri interi a e b, scrivere una funzione che restituisca il risultato di a^b; proporre quindi un adeguato main di prova.

Si ricorda che se b=0 allora $a^b=1$ e se b<0 allora $a^b=(1/a)^{-b}$.

La **funzione potenza**

- prende **in ingresso** due valori interi (a, b)
- restituisce **in uscita** un valore reale.

Il passaggio di entrambi i parametri è per valore perché né a né b devono essere modificati

Esercizio 1 (cont.)

```
#include<stdio.h>

/* dichiarazione */
float potenza(int, int);

/* cliente */
void main (){
int base, esponente;
printf("Base: "); scanf("%d",&base);
printf("Esponente: "); scanf("%d",&esponente);
printf("Risultato di %d^%d: %f", base, esponente,
   potenza(base,esponente));}
/* funzione */
float potenza(int a, int b){
float base, ris; int i;
if(b<0){b=-b; base=1/(float) a;}
   else base=a;
for(i=1,ris=1;i<=b;i++) ris=ris*base;
return ris;}
```


Esercizio 2

Dato un carattere alfabetico minuscolo:

- scrivere una funzione che restituisca il corrispondente carattere maiuscolo;
- scrivere una funzione che, fornito un offset in input (valore intero), restituisca il carattere “shiftato” (con rotazione) in avanti di offset posizioni;
- scrivere un main che chiede all’utente il carattere e il tipo di operazione a cui è interessato (maiuscolo o shifting) e, dopo aver controllato che l’input sia corretto, stampa il risultato dell’operazione richiesta.

Esercizio 2 (cont.)

- la funzione **maiuscolo**
  - prende in **ingresso** un carattere (car)
  - restituisce **in uscita** un carattere (car_maiu)
- la funzione **shift**
  - prende in **ingresso** un carattere (car) e un offset
  - restituisce in **uscita** un carattere (car_shift)
- Il main
  - chiede e controlla il carattere in input
  - chiede il tipo di operazione (se l'operazione richiesta è uno shift allora chiede l'offset)
  - chiama il corrispondente funzione e stampa il risultato

Esercizio 2 (cont.)

```
#include<stdio.h>

/* dichiarazioni */
charmaiuscolo(char);
charshift(char, int);

/* cliente */
voidmain ()
{
 char car;
 int offset, op;
 do {printf("Carattere: "); scanf("%c",&car);}
 while(!(car>='a' &&car<= 'z'));
 do {printf("Operazione(1-maiuscolo, 2-shift): ");
 scanf("%d",&op);}
 while(!(op==1 || op ==2));
```

Esercizio 2 (cont.)

```
 if (op==1)
 printf("Carattere maiuscolo: %c",maiuscolo(car));
 else
 {printf("offset: "); scanf("%d",&offset);
 printf("Risultato operazione di shift: %c \n",
  shift(car,offset));}
}
/* servitori */
char maiuscolo(char c)
{return c+'A'-'a';}
char shift(char c, int i){
int ris=c+i;
if(ris>'z') return (ris%'z')+'a'-1;
return ris;}
```


Esercizio 3

Data una sequenza di numeri verificare quali di questi è primo.

Un possibile **approccio top-down** alla soluzione del problema:

1. ricevi in input un numero;
2. verifica se è primo;
3. stampa il risultato;
4. se l'utente ha un altro numero da controllare allora torna al passo (1).

Il passo 2 può essere implementato con una funzione `IsPrime` che riceve in input un intero (`n`) e restituisce un valore booleano.

Variante

Data una sequenza di numeri verificare quali di questi è primo e per quelli non primi restituire il primo divisore.

- L'approccio è lo stesso.
- Il passo 2 può essere implementato con una funzione `IsPrime` che riceve in input un intero (`n`) e restituisce un valore booleano oltre al primo divisore (se il numero non è primo) ***in un parametro passato per indirizzo***.

```
Bool IsPrime(int n, int *primo_div);
```

Esercizio 3 (cont.)

```
#include<stdio.h>
#include<math.h>
/* dichiarazioni */
typedef int Bool;
Bool IsPrime(int);
/* cliente */
void main () {
 int num; Bool continua;
 do {
   printf("Numero: ");scanf("%d",&num);
   if(IsPrime(num))
   printf("%d e' primo ",num);
   else
   printf("%d non e' primo\n",num);
   printf("Vuoicontinuare (0=fine, 1=continuo)?");
   scanf("%d",&continua);}
 while(continua);}
```


Esercizio 3 (cont.)

```
/* funzione */
Bool IsPrime(int num){
  int i;
  double max;
  if (num>=1 && num<=3)
     return 1;
  if (num%2==0)
     return 0;
  max = sqrt(num);
  for(i=3; i<=max; i+=2)
  if (num%i==0)
       return 0;
  return 1;
}
```

Esercizio 3 - Variante

```
#include<stdio.h>
#include<math.h>
/* dichiarazioni */
typedef int Bool;
Bool IsPrime(int, int *);

void main () {
  int num, div; Bool continua;
  do {
  printf("Numero: ");
  scanf("%d",&num);
  if(IsPrime(num,&div))
  printf("%d e' primo ",num);
  else
  printf("%d non primo: %d div.\n",num,div);
  printf("Vuoicontinuare (0=fine, 1=continuo)?");
  scanf("%d",&continua);}
  while(continua);
}
```


Esercizio 3 – Variante (cont.)

```
Bool IsPrime(int num, int* primo_div){
  int i;
  double max;
  if (num>=1 &&num<=3)
     return 1;
  if (num%2==0){
     *primo_div=2;
     return 0;}      /* no numeri pari */
  max = sqrt(num);
  for(i=3; i<=max; i+=2)
     if (num%i==0){
        *primo_div=i;
        return 0;}
  return 1;
}
```

Esercizio 4

Dato una linea di testo scrivere un programma che ne esamina i caratteri e conta il numero di caratteri che rientrano nelle seguenti categorie: vocali, consonanti, cifre, spazi e "altri" caratteri.

Un possibile **approccio top-down** alla soluzione del problema:

1) acquisisci una riga da tastiera e memorizzala in un vettore
2) per ogni carattere della riga

   2.1) trasforma il carattere in un formato standard (ad es. in maiuscolo attraverso la funzione `chartoupper(char)` contenuta nella libreria `ctype.h`)

   2.2) analizza il carattere e aggiorna il contatore della categoria a cui appartiene il carattere
3) stampa il risultato a video

Il passo 2.2 può essere implementato con una procedura `scan_car` che riceve in input un carattere e aggiorna il contatore della categoria di appartenenza

```
void scan_car(char l, int* pv, int* pc, int* pd, int* pw,
int* po);
```


Esercizio 4 (cont.)

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

/* dichiarazioni */
void scan_car(char, int*, int*, int*, int*, int*);

/* cliente */
void main (){
char line[80];
char c;
int count;
int voc=0, cons=0, dig=0, ws=0, altri=0;

printf("Digitare la riga:");
gets(line);
for(count=0; (c = line[count])!='\0'; count++){
      c = toupper(c);
      scan_car(c, &voc, &cons, &dig, &ws, &altri);}
```

Esercizio 4 (cont.)

```
printf("Num. vocali: %d\n",voc);
printf("Num. consonanti: %d\n",cons);
printf("Num. cifre: %d\n",dig);
printf("Num. spazi bianchi: %d\n",ws);
printf("Num. altri caratteri: %d\n",altri);}

/* funzione */
void scan_car(char c, int* pv, int* pc, int* pd, int* pw, int*
   po){
if(c=='A'||c=='E'||c=='I'||c=='O'||c=='U'){(*pv)++; return;}
if(c>='A'&&c<='Z'){(*pc)++; return;}
if(c>='0'&&c<='9'){(*pd)++; return;}
if(c==' ') {(*pw)++; return;}
(*po)++;
}
```


Esercizio 4: modalità di lettura alternative

L'acquisizione della riga si può realizzare con `gets()` (funzione in `<stdio.h>`)

```
void main (){
char line[80];
char c;
int count;
int voc=0, cons=0, dig=0, ws=0, altri=0;

printf("Digitare la riga:");
gets(line);
for(count=0; (c = line[count])!='\0'; count++)
…}
```

… oppure si può acquisire carattere per carattere con `getchar()` (funzione in `<stdio.h>`):

```
void main (){
char c;
int count;
int voc=0, cons=0, dig=0, ws=0, altri=0;

printf("Digitare la riga:");
for(count=0; (c = getchar())!='\n'; count++)
```

Esercizio 5

Scrivere una funzione che concatena la stringa s2 alla stringa s1 e proporre un adeguato main di prova per stringhe lunghe al più caratteri.

La funzione

- prende in **ingresso** due stringhe (s1,s2)
- restituisce in s1 il risultato della concatenazione

NOTA

Le stringhe sono vettori e quindi sono sempre passate ad una funzione/procedura per indirizzo. Ogni modifica al parametro formale s1 è una modifica al corrispondente parametro attuale.

Esercizio 5 (cont.)

```
#include <string.h>

void concatena(char s1[], char s2[]){
int i,j;
for(i=strlen(s1),j=0;j<strlen(s2);i++,j++)
  s1[i]=s2[j];
s1[i]='\0';
}

int main(){
char s1[51], s2[51];

scanf("%s %s",s1,s2);
concatena(s1,s2);
printf("%s", s1);}
```

Esercizio 6

Dato un vettore di dimensione N di stringhe lunghe al più 10 caratteri e due stringhe s1 e s2, scrivere una funzione che sostituisce ogni occorrenza di s1 con s2.

La funzione

- prende in **ingresso** un vettore vett di stringhe lunghe al più 10 caratteri, la dimensione N del vettore e due stringhe (s1,s2)
- restituisce vett modificato

CODIFICA

```
#include<string.h>
typedef char stringa[11];

void sost(stringa vett[], int N, stringa s1, stringa s2){
int i;
for(i= 0;i<N;i++)
   if(strcmp(vett[i],s1)==0)
strcpy(vett[i],s2);
}
```

ESERCIZIO Proporre un adeguato main di prova!

Esercizio 7

Dato vettore vett1 di interi di dimensione N e due interi min e max

- scrivere una funzione che restituisce nel vettore vett2 gli elementi di vett1 compresi tra min e max e la dimensione del vettore
- proporre un adeguato main di prova dove vett1 è inizializzato da tastiera e contiene al più 100 elementi.

La funzione/procedura `preleva` deve

- prendere in **ingresso** un vettore di interi vett1, la dimensione N del vettore e due interi (min,max)
- restituire in vett2 un sottoinsieme degli elementi di vett1
- restituire la dimensione del vettore vett2

Esercizio 7 (cont.)

NOTA

- Il vettore vett2 è passato a `preleva` per indirizzo.
- Per la dimensione del vettore, consideriamo due soluzioni alternative:
  - preleva è una funzione che restituisce in uscita la dimensione del vettore

    ```
    int preleva(int vett1[],int N, int min,
    int max, int vett2[]);
    ```

  - preleva è una procedura che restituisce la dimensione in un parametro

    ```
    void preleva((int vett1[], int N, int
    max, int min, int vett2[], int* M);
    ```


Esercizio 7 (cont.)

```
#include<stdio.h>
int preleva(int vett1[], int N, int min, int max,
  int vett2[]);

void main () {
int origin[100], dimO, dest[100], dimD, m, n;
int i;

printf("Numero elementi: "); scanf("%d",&dimO);
for(i=0;i<dimO;i++){
  printf("Elemento %d-esimo: ", i+1);
  scanf("%d",&origin[i]);}
```

Esercizio 7 (cont.)

```
printf("Elemento minimo: "); scanf("%d",&m);
printf("Elemento massimo: "); scanf("%d",&n);
dimD=preleva(origin,dimO,m,n,dest);
for(i=0;i<dimD;i++) printf("Elemento %d-esimo: %d\n",
  i+1, dest[i]);}

int preleva(int vett1[], int N, int min, int max, int
  vett2[]){
int i, j;

for(i=0, j=0;i<N;i++)
if((min<=vett1[i])&&(vett1[i]<=max)){
      vett2[j]=vett1[i];
      j++;}
return j;}
```


Esercizio 8

Dati

- una collezione di 10 libri caratterizzati dall' autore (al più 20 caratteri), dal titolo (al più 30 caratteri), dall'editore (al più 30 caratteri) e dal numero di pagine, inizializzati da tastiera e memorizzati in un vettore
- un autore

scrivere

- una procedura per l'inizializzazione del vettore con dati forniti da tastiera,
- una procedura che stampi a video il titolo dei libri dell'autore specificato.

Esercizio 8 (cont.)

```
#include <stdio.h>
#include <string.h>
#define N 10
typedef struct {
   char autore[21];
   char titolo[31];
   char editore[31];
   int pagine;}Libro;
void acquisisci(Libro l[], int dim);
void stampa_libri(Libro l[], int dim, char nome[]);

void main(){
Libro libri[N];
char autore[21];

acquisisci(libri,N);
printf("Autore: "); scanf("%s",autore);
stampa_libri(libri,autore);}
```


Esercizio 8 (cont.)

```
void acquisisci(Libro l[], int dim){
int i;

for (i=0;i<dim;i++)
{ printf("Nome autore: "); scanf("%s",l[i].autore);
  printf("Titolo libro: "); scanf("%s", l[i].titolo);
  printf("Nome editore: "); scanf("%s",l[i].editore);
  printf("Numero pagine: "); scanf("%d", &l[i].pagine);}
}

void stampa_libri(Libro l[],int dim, char nome[]){
int i;
for(i=0;i<dim;i++)
  if(strcmp(l[i].autore,nome)==0)
      printf("%s",l[i].titolo);
}
```

Esercizio 9

Dati:

- una collezione di 10 libri caratterizzati dall' autore (al più 20 caratteri), dal titolo (al più 30 caratteri), dall'editore (al più 30 caratteri) e dal numero di pagine, inizializzati da tastiera e memorizzati in un vettore
- un autore

scrivere un funzione che copi in un vettore i soli libri dell'autore specificato con un numero di pagine inferiore a 100.

NOTA

Oltre al vettore, la funzione deve restituire anche la sua dimensione:

```
int copia(Libro l1[], int dim, Libro l2[], char nome[]);
```


Esercizio 9 (cont.)

```
int copia(Libro l1[], int dim, Libro l2[],
  char nome[])
int i,j;
for(i=0,j=0;i<N;i++)
  if((strcmp(l1[i].autore,nome)==0)
       && l1[i].pagine< 100){
    l2[j]=l1[i];
    j++;}
return j;}
```

Esercizio 10

- Scrivere un programma che dati due vettori di al più N elementi di tipo reale, presenti all'utente il seguente menù per la gestione di vettori:

  1 – stampare i vettori
  2 – stampare il vettore somma
  3 – calcolare il prodotto scalare dei due vettori
  4 – calcolare il modulo dei due vettori
  5 – calcolare l'angolo fra i due vettori
  6 – fine programma

Il programma deve consentire all'utente di eseguire una o più operazioni fino a quando non digita l'opzione di uscita (6).
Al termine di ogni operazione eseguita il programma deve stampare il contenuto del vettore ottenuto.

Esercizio 10 (cont)

```
void stampa_vettore (float vett1[], int n)
//stampa un vettore di float
{
   int i;

   for(i=0;i<n;i++)
     printf("%f\t",vett1[i]);
   printf("\n");

}
```

- Lettura e scrittura di un vettore

```
void leggi_vettore (float vett1[], int n)
// legge un vettore di float
{

   int i;

   printf("Dammi il vettore:  ");
   for(i=0;i<n;i++)
     scanf("%f",&vett1[i]);
}
```

Esercizio 10 (cont)

```
float prod_scalare_vettori(float vett1[], float vett2[], int n)
// prodotto scalare fra vett1 e vett2
{
   int i;
   float prod=0;

   for(i=0;i<n;i++)
      prod+=vett1[i]*vett2[i];

  return prod;
}
```

- Prodotto scalare e modulo

```
float modulo(float vett1[], int n)
// modulo di vett1
{
   float prod=0;

   return
   sqrt(prod_scalare_vettori(vett1,vett1,n)
   );
}
```


Esercizio 10 (cont)

```
int angolo_vettori(float *angolo, float vett1[], float vett2[], int n)
// la funzione ritorna VERO se è possibile fare il calcolo
// Se è possibile fare il calcolo, il risultato è messo in "angolo"
{
   int i;
   float mod1,mod2;

   mod1=modulo(vett1,n);
   if(mod1==0) return 0;

   mod2=modulo(vett2,n);
   if(mod2==0) return 0;

   //calcolo dell'angolo in radianti
   *angolo=acos(prod_scalare_vettori(vett1,vett2,n)/(mod1*mod2));

   //calcolo dell'angolo in gradi
   *angolo=(*angolo)*180.0/3.14159265;

   return 1;
}
```

- Angolo fra vettori

Esercizio 10 (cont)

```
int main (){   . . .
   printf("Quanti valori vuoi immettere?  "); scanf("%d",&n_elementi);
   leggi_vettore(vett1,n_elementi);
   leggi_vettore(vett2,n_elementi);
   ok=1;
   while(ok){
    system("cls\n");
     printf("MENU\n");   .  . .
     printf("5) Stampa dell'angolo fra i due vettori\n");
     scanf("%d",&scelta);
    switch(scelta){
          .  .  .
        case 5: system("cls\n");
                if(angolo_vettori(&angolo,vett1,vett2,n_elementi))
                       printf("L'angolo e: %f\n",angolo);
                  else
                       printf("Calcolo non possibile\n");
                     getchar(); getchar();
                  break;
          .  .  .
```

- Main: chiamata delle letture di due vettori
- Main: calcolo dell'angolo fra due vettori

Esercizio 11

- Scrivere un programma che data una matrice di al più M*N elementi di tipo reale, presenti all'utente il seguente menù per la gestione dei vettori riga:

  1 – stampare un vettore a scelta dell'utente
  2 – stampare la matrice
  3 – calcolare il modulo di un vettore a scelta dell'utente
  4 – calcolare il prodotto scalare di due vettori a scelta
  5 – calcolare l'angolo fra due vettori a scelta
  6 – fine programma

Il programma deve consentire all'utente di eseguire una o più operazioni fino a quando non digita l'opzione di uscita (6).
Al termine di ogni operazione eseguita il programma deve stampare il contenuto del vettore ottenuto.

Esercizio 11 (cont)

```
void stampa_matrice(float mat[][NC], int n, int m)
// stampa la matrice
{ int i,j;

   for(i=0;i<n;i++){
     for(j=0;j<m;j++)
       printf("%f\t",mat[i][j]);
     printf("\n");
   }
}
```

- Lettura e scrittura di una matrice a due dimensioni

```
void leggi_matrice(float mat[][NC], int n, int m)
// legge la matrice
{ int i,j;

   for(i=0;i<n;i++){
     printf("Dammi la riga n. %d :  ",i);
     for(j=0;j<m;j++)
       scanf("%f",&mat[i][j]);
   }
}
```


Esercizio 11 (cont)

```
int main ()
{  ...
   int i,scelta,ok,n_righe,n_colonne,vett1,vett2;
   float mat[NR][NC],angolo;
   . . .
   leggi_matrice(mat,n_righe,n_colonne);
   ok=1;
   while(ok){
    system("cls\n");
     printf("MENU\n");
     printf("1) Stampa di un vettore riga a scelta\n");
     . . .
    switch(scelta){
        case 1: system("cls\n");
                do{
                     printf("Dammi un vettore:  ");
                     scanf("%d",&vett1);
                }while(vett1<0||vett1>=n_righe);
                stampa_vettore(mat[vett1],n_righe);
                getchar(); getchar();
              break;
```

Esercizio 11 (cont)

```
int main ()
{   ...
    int i,scelta,ok,n_righe,n_colonne,vett1,vett2;
    float mat[NR][NC],angolo;
    . . .
    leggi_matrice(mat,n_righe,n_colonne);
    ok=1;
    while(ok){
     system("cls\n");
      printf("MENU\n");
      printf("5) Stampa dell'angolo fra due vettori a scelta\n");
      . . .
     switch(scelta){
          case 5: system("cls\n");
          printf("Dammi il primo vettore:   ");scanf("%d",&vett1);
          printf("Dammi il secondo vettore: ");scanf("%d",&vett2);
          if(angolo_vettori(&angolo,mat[vett1],mat[vett2],n_righe))
              printf("L'angolo e: %f\n",angolo);
           else
              printf("Calcolo non possibile\n");
           getchar(); getchar(); break;
```