

Classi di problemi

	Senza vettori	Con vettori/matrici
Numerici	Equaz. 2° grado Serie e successioni numeriche ...	Statistiche per gruppi Operazioni su insiemi di numeri Generazione numeri primi Somme/prodotti matriciali
Codifica	Conversioni di base (es. binario/decimale) Crittografia di testo ...	Conversioni tra basi numeriche Ricodifica testi utilizzando tabelle di conversione
Elab. Testi	Manipolazione stringhe Menu con scelta Grafico di funzione (asse X verticale)	Conteggio caratteri in testo Grafico funzione (asse X orizzontale) Formattazione testo (centrare, eliminare spazi)
Verifica/ selezione	Verifica di ordinamento/congruenza di dati Verifica mosse di un gioco Filtro su elenco di dati Ricerca massimo o minimo Ordinamento parziale	Verifica di unicità (o ripetizione) di dati Selezione di dati in base a criterio di accettazione Ricerca di dato in tabella (in base a nome/stringa) Ordinamento per selezione

Problemi su valori non numerici

- I caratteri in C sono in genere codificati secondo lo standard ASCII:
 - Internamente il codice ASCII utilizza 8 bit
 - NON è necessario conoscere le codifiche: è sufficiente ricordare che i sottoinsiemi di caratteri alfabetici (maiuscoli e minuscoli) e numerici sono contigui (e ordinate)
 - Esternamente (input/output) i caratteri sono visualizzati in forma testuale

Codice ASCII

- Codifica standard: il testo è l'informazione più scambiata
- Usa 8 bit per rappresentare:
 - 52 caratteri alfabetici (a...z A...Z)
 - 10 cifre (0...9)
 - Segni di interpunzione (,;!?...)
 - Caratteri di controllo, ad esempio:
 - '\n' line feed = a capo
 - '\t' tab = tabulazione

Codice ASCII

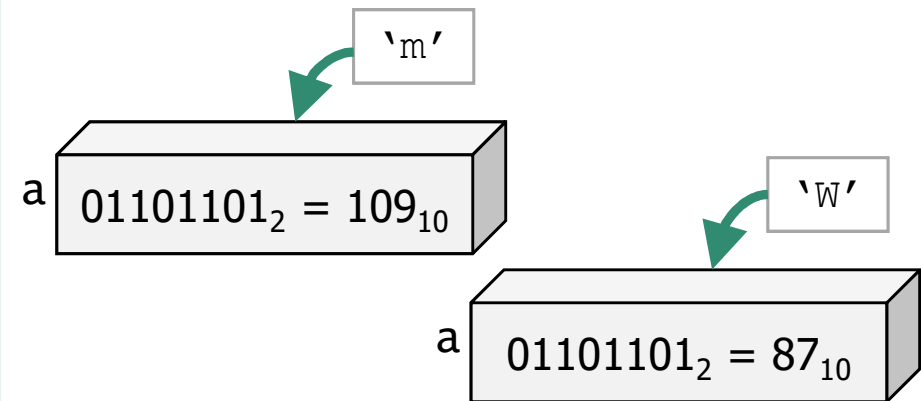
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Tipo char

- Tipo fornito direttamente dal C per rappresentare caratteri
 - Dimensione = 8 bit = 1 Byte sempre
 - Definita dalla codifica ASCII

```
#include <stdio.h>
int main(void) {
    char a;
    a = 'm';
    printf("variabile a come char: %c\n", a);
    printf("variabile a come int: %d\n", a);
    a = 'W';
    printf("variabile a come char: %c\n", a);
    printf("variabile a come int: %d\n", a);
    return 0;
}
```

```
❏ ./main
variabile a come char: m
variabile a come int: 109
variabile a come char: W
variabile a come int: 87
❏
```



Problemi su valori non numerici

- Le operazioni aritmetiche, di confronto tra caratteri, e il loro ordine, rispettano la codifica ASCII
 - Per gli operatori aritmetici i codici sono assimilati a binari interi su 8 bit
 - Esempi:
 - 'C' - 'A' vale 2
 - 'Z' - 'A' vale 25
 - Attenzione: maiuscole e minuscole → 'Z' e 'a' non sono contigui, devo lavorare separatamente su maiuscole e minuscole
 - Si possono usare gli operatori relazionali (==, !=, >, <, >=, <=)
 - Basati sul codice ASCII, ad esempio 'A' è minore di 'C'

Problemi su valori non numerici

```
int main(void) {
    int b0, b1, n, p, cifra, fine=0;
    char c;

    printf("b0 (2..9): "); scanf("%d",&b0);
    printf("b1 (2..9): "); scanf("%d\n",&b1);

    n = 0;
    while (!fine) {
        scanf("%c",&c);
        if (c== ' ' || c== '\n') {
            converti(n,b1); n=0;
        }
        else {
            cifra = c - '0';
            if (cifra >= 0 && cifra < b0)
                n = b0*n + cifra;
            else fine=1;
        }
    }
}
```

- Ottengo la conversione del carattere in intero calcolando la distanza dal carattere '0' nella tabella ASCII

- Esempio:

- $c = '0' \rightarrow \text{cifra} = '0' - '0' = 48 - 48 = 0$
- $c = '3' \rightarrow \text{cifra} = '3' - '0' = 51 - 48 = 3$
- $c = '9' \rightarrow \text{cifra} = '9' - '0' = 57 - 48 = 9$

Problemi di codifica di caratteri

■ Perché?

- Può essere necessario determinare il codice ASCII di un carattere:
 - Basta una funzione di conversione di un numero a binario
- Si può effettuare una ricodifica per:
 - Crittografare/decrittografare o compattare/scompattare un testo

■ Che Algoritmi?

- I problemi di codifica si risolvono spesso mediante algoritmi con:
 - Gestione del singolo carattere mediante manipolazioni (eventualmente numeriche) sul codice del carattere (codifica/decodifica)
 - Iterazioni per trattare sequenze di caratteri

Crittografia semplice

■ Formulazione:

- Crittografare il contenuto di un file testo, immagazzinando il risultato in un secondo file
- La crittografia consiste nel modificare i codici dei caratteri alfabetici e numerici, secondo le regole seguenti:
 - Ogni codice numerico n (0..9) viene trasformato nel codice complemento-a-9: $9-n$
 - ogni codice alfabetico ch viene trasformato scambiando maiuscole e minuscole, e facendo il complemento-a- z : $'a' + 'z' - ch$

■ Soluzione:

- Leggere iterativamente i caratteri dal primo file
- A seconda del tipo di carattere applicare la codifica opportuna
- Scrivere il carattere via via ottenuto sul secondo file

Crittografia semplice

■ Struttura dati:

- Due variabili di tipo puntatore a FILE fpin, fpout per gestire i due file in lettura e scrittura
- Una stringa nomefile per i nomi dei file
- Una variabile char per lettura e ri-codifica dei caratteri

■ Algoritmo:

- Acquisizione dei nomi di file e loro apertura
- Iterazione di lettura di un carattere – ri-codifica – scrittura nel secondo file
 - La ri-codifica viene fatta mediante selezione del sottoinsieme di caratteri, e relativa operazione (sfruttando l'aritmetica dei codici)
 - Ad es., il nuovo codice della 'c' si ottiene sommando ad 'A' la differenza tra 'z' e 'c'

Codice

```
#define MAXRIGA 30
int main(void) {
    char ch, nomefile[MAXRIGA+1];
    FILE *fpin, *fpout;

    printf("nome file in ingresso: ");
    scanf("%s", nomefile);
    fpin = fopen(nomefile, "r");
    printf("nome file in uscita: ");
    scanf("%s", nomefile);
    fpout = fopen(nomefile, "w");
```

```
    while (fscanf(fpin, "%c", &ch) == 1) {
        if (ch>= '0' && ch<= '9')
            ch = '0' + ('9' - ch);
        else if (ch>= 'a' && ch<= 'z')
            ch = 'A' + ('z' - ch);
        else if (ch>= 'A' && ch<= 'Z')
            ch = 'a' + ('Z' - ch);
        fprintf(fpout, "%c", ch);
    }
    fclose(fpin); fclose(fpout);
}
```

Problemi di text-processing

- Problemi nei quali occorre manipolare sequenze di caratteri e/o stringhe
 - Esempio: costruzione o modifica di testo, creazione di messaggio in un dato formato
- Scopi possibili:
 - Riconoscimento di un testo (input: menu per input di comandi)
 - Visualizzazione di un testo (output: grafica «elementare»)
 - Elaborazione di un testo (modifica: formattazione)
- Le operazioni possono essere effettuate a livello di:
 - Singoli caratteri (livello più basso)
 - Stringhe (livello più alto: una stringa corrisponde a una parola o frase)

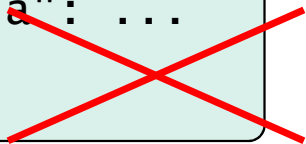
Problemi di text-processing

- Il C consente input/output sia a livello di carattere che di stringa
- La manipolazione di un testo a livello di stringa dipende dalla disponibilità di funzioni di libreria (o altre scritte dall'utente)
- Il confronto tra testi si effettua:
 - A livello di caratteri, mediante gli operatori relazionali (==, !=, >, <, >=, <=)
 - A livello di stringhe si richiede la funzione strcmp (o strncmp)
- Il costrutto di selezione if:
 - E' il più generale (switch può essere espresso in termini di if e non viceversa)
 - Utilizzabile sempre (pur di formulare correttamente l'espressione di controllo)

Problemi di text-processing

- Il costrutto switch richiede selezione in base a valori costanti
- **Non** può quindi essere applicato:
 - A costanti stringa (perchè sarebbero confrontati i puntatori e non i contenuti delle stringhe)

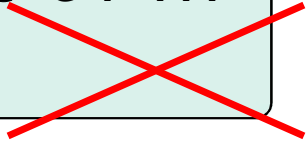
```
switch (stato) {  
    case "Italia": ...  
    ...  
}
```



```
if strcmp (stato, "Italia") == 0) {  
    ...  
}  
else if (...)  
    ...
```

- A insiemi o intervalli di caratteri (se non mediante enumerazione di tutti i caratteri possibili)

```
switch (car) {  
    case 'A': case 'B': case 'C': ...  
    ...  
}
```



```
if (car >= 'A' && car <= 'Z') {  
    ...  
}  
else if (...)  
    ...
```

Problemi di text-processing

- Il costrutto switch richiede selezione in base a valori costanti

- **No**

- A

Si possono usare le funzioni di libreria (ctype.h): isalpha, isupper, islower, isdigit, isalnum, isxdigit, ispunct, isgraph, isprint, isspace, iscntrl

contenuti delle

```
italia") == 0) {
```

- A insiemi o intervalli di caratteri (se non mediante enumerazione di tutti i caratteri possibili)

```
switch (car) {  
  case 'A': case 'B': case 'C': ...  
  ...  
}
```

```
if (isalpha(car)) {  
  ...  
}  
else if (...)  
  ...
```

Funzioni su Caratteri (<ctype.h>)

<i>FUNZIONE</i>	<i>DEFINIZIONE</i>
<code>int isalnum (char c)</code>	Se c è lettera o cifra
<code>int isalpha (char c)</code>	Se c è lettera
<code>int isascii(char c)</code>	Se c è lettera o cifra
<code>int isdigit (char c)</code>	Se c è una cifra
<code>int islower(char c)</code>	Se c è minuscola
<code>int isupper (char c)</code>	Se c è maiuscola
<code>int isspace(char c)</code>	Se c è spazio,tab,\n
<code>int iscntrl(char c)</code>	Se c è di controllo
<code>int isgraph(char c)</code>	Se c è stampabile, non spazio
<code>int isprint(char c)</code>	Se c è stampabile
<code>int ispunct(char c)</code>	Se c è di interpunzione
<code>int toupper(char c)</code>	Converte in maiuscolo
<code>int tolower(char c)</code>	Converte in minuscolo

Stringhe

- Non sono un tipo di dato, ma un caso particolare di vettore (di char)
 - Esempi: `char nome[N];`
- Contengono il **terminatore** di stringa, il carattere `'\0'` (codice ASCII 0) posto dopo l'ultimo carattere significativo:
 - Di solito il vettore è sovradimensionato, il `'\0'` indica quale sia la parte utilizzata
- Vi si può operare con operazioni atomiche (su tutta la stringa), che richiedono stringhe (con `'\0'`) come operandi e generano stringhe (con `'\0'`) come risultato:
 - Le costanti stringa (parole tra doppi apici) sono vettori di char con terminatore
 - Funzioni di IO: `gets/puts`, `fgets/fputs`, IO formattato con `%s`
 - Funzioni disponibili includendo `<string.h>`: le più frequenti sono: `strlen`, `strcpy`, `strcmp`, `strncmp`, `strcat`

Esempi

```
char words[NP][MAXL], first[MAXL], last[MAXL], firstAndLast[2*MAXL];
// read words from file
int n;
for (n=0; fscanf(fin, "%s", words[n]) != EOF; n++);
// verify order
int i, sorted = 1;
for (i=1, i<n && sorted; i++) {
    if (strcmp(words[i-1], words[i]) > 0) { // if words[i-1] > words[i]
        sorted = 0;
    }
}
// copy first and last word
strcpy(first, words[0]); // WARNING: first = words[0] is WRONG!
strcpy(last, words[n-1]); // WARNING: last = words[n-1] is WRONG!
// join first and last
strcpy(firstAndLast, first); // copy first
strcat(firstAndLast, last); // append last
```

Stringhe come vettori di char

- Vi si può anche operare (SE NECESSARIO/OPPORTUNO) come vettore (sui singoli caratteri): può essere necessario se non c'è un'operazione atomica equivalente o se c'è (ci sono) ma è complicato o inefficiente
- Esempi:
 - Rimozione del '\n' lasciato dalla fgets

```
fgets(s,MAXL,fin);
if (s[strlen(s)-1]=='\n')
    s[strlen(s)-1] = '\0';
```
 - Conversione di una parola in maiuscolo (esistono toupper e tolower ma SOLO per caratteri singoli, non per stringhe)

```
void stringToUpper(char s[]) {
    int i, len = strlen(s);
    for (i=0; i<len; i++)
        s[i] = toupper(s[i]);
}
```
 - Altro: conteggio vocali, rimozione o sostituzione di caratteri, ecc.

Funzioni per Stringhe (<string.h>)

<i>FUNZIONE</i>	<i>DEFINIZIONE</i>
<code>char* strcat (char* s1, char* s2);</code>	concatenazione s1+s2
<code>char* strchr (char* s, int c);</code>	ricerca di c in s
<code>int strcmp (char* s1, char* s2);</code>	confronto
<code>char* strcpy (char* s1, char* s2);</code>	s1 <= s2
<code>int strlen (char* s);</code>	lunghezza di s
<code>char* strncat (char* s1,char* s2,int n);</code>	concat. n car. max
<code>char* strncpy (char* s1,char* s2,int n);</code>	copia n car. max
<code>char* strncmp(char* dest,char* src,int n);</code>	cfr. n car. max

Selezione a menu

- La selezione a menu consiste nell'effettuare una scelta, tra le varie disponibili, per eseguire un'azione (un calcolo, una chiamata di funzione o altro):
 - Spesso la scelta si basa su un'informazione testuale (un comando o opzione)
 - L'elenco delle possibili opzioni è visualizzata a video (se si tratta di I/O tastiera/video)
 - Occorre prevedere un caso di errore (o scelta non valida)
 - Il menu viene di solito iterato (una delle opzioni indica fine/uscita)

Menu con scelta su un carattere

- È il caso più semplice: si prevedono un certo numero di casi, ognuno selezionato da una costante carattere distinta
 - L'iniziale di un comando oppure il comando stesso
- Eventuali complicazioni:
 - Può esser necessario saltare un certo numero di spazi prima di individuare il carattere di selezione
 - Se il carattere è alfabetico, è possibile che occorra ignorare la differenza maiuscolo/minuscolo
- Soluzione: i costrutti if e switch sono entrambi adatti (di solito si utilizza switch)

Menu con scelta su un carattere

■ Formulazione:

- Scrivere una funzione che, iterativamente, acquisisca da tastiera una stringa (al massimo 50 caratteri, contenente eventuali spazi)
 - Il primo carattere diverso da spazio costituisce il selettore
 - Se il carattere è 'u' (uscita), occorre terminare l'iterazione
 - Se il carattere è uno tra 'A', 'L', 'T' (eventualmente minuscoli), occorre attivare, rispettivamente, le funzioni fA, fL, fT, passando loro come parametro il resto della stringa (oltre il carattere selettore)
 - Ogni altro carattere va segnalato come errato

Codice

```
void menuCarattere (void) {  
    const int MAXL=51;  
    char riga[MAXL];  
    int i, continua=1;  
  
    while (continua) {  
        printf("comando (A/L/T, U=uscita): ");  
        gets(riga);  
        for (i=0; riga[i]!='\0'; i++);  
        ...  
    }
```

```
        ...  
        switch (toupper(riga[i])) {  
            case 'A' : fA(&riga[i+1]); break;  
            case 'L' : fL(&riga[i+1]); break;  
            case 'T' : fT(&riga[i+1]); break;  
            case 'U' : continua=0; break;  
            default: printf("comando errato\n");  
        }  
    }  
}
```


Menu con scelta su una parola

- È il caso meno semplice: si prevedono un certo numero di casi, ognuno selezionato da una costante stringa distinta
 - La prima parola di un comando oppure il comando stesso
- Eventuali complicazioni:
 - Può esser necessario saltare un certo numero di spazi prima di individuare la parola di selezione
 - Se i caratteri sono alfabetici, è possibile che occorra ignorare la differenza maiuscolo/minuscolo
- Soluzione: è necessario il costrutto if

Menu con scelta su una parola

■ Formulazione:

- Scrivere una funzione che, iterativamente, acquisisca da tastiera una stringa (al massimo 50 caratteri, contenente eventuali spazi)
 - La prima parola diversa da spazio costituisce il selettore
 - Se la parola è "fine", occorre terminare l'iterazione
 - Se la parola è uno tra "cerca", "mod", "sta" (ignorare differenza maiuscole/minuscole), occorre attivare, rispettivamente, le funzioni cerca, sostituisci, stampa, passando loro come parametro il resto della stringa (oltre la parola di selezione)
 - ogni altra parola va segnalata come errata

Codice

```
void menuParola (void){
    const int MAXL=51;
    char comando[MAXL], riga[MAXL];
    int i, continua=1;
    while (continua) {
        printf("comando (cerca/modifica");
        printf("stampa/uscita): ");
        scanf("%s", comando); /* comando */
        for (i=0; i<strlen(comando); i++)
            comando[i] = toupper(comando[i]);
        gets(riga); /* resto della riga */
        ...
    }
```

```
...
    if (strcmp(comando, "CERCA")==0) {
        cerca(riga);
    } else if (strcmp(comando, "MOD")==0) {
        sostituisci(riga);
    } else if (strcmp(comando, "STA")==0) {
        stampa(riga);
    } else if (strcmp(comando, "FINE")==0) {
        continua=0;
    } else {
        printf("comando errato\n");
    }
}
```

Elaborazione testi - livello carattere

- Un testo può essere costruito o modificato a livello di caratteri perchè:
 - Il problema viene posto a livello di singoli caratteri (e non parole/stringhe)
 - Non ci sono quindi alternative
 - Il problema potrebbe essere risolto (anche parzialmente) a livello di stringhe, ma si sceglie di lavorare a livello di caratteri
 - Motivi: varianti di funzioni di libreria o miglior gestione di casi particolari
- Attenzione! Nei casi di manipolazione mista (caratteri e stringhe) occorre gestire il terminatore di stringa ('\0')

Costruzione di figure/grafici

- Si tratta di figure formate da caratteri testuali (es. video visto come matrice di caratteri di 25 righe e 80 colonne)
- NON si tratta di grafica basata su punti (pixel), per cui occorrono apposite librerie o linguaggi
 - Non scopo di questo corso
- La visualizzazione di caratteri su video (o su file testo) viene effettuata in modo sequenziale (successione di caratteri e righe)
- Un disegno o grafico può essere costruito (e visualizzato):
 - Riga per riga (esempi in questa unità)
 - Su una matrice di caratteri, da cui si passa successivamente a visualizzare le righe (unità successiva)

Visualizzazione di un rettangolo

- Formulazione:

- Scrivere una funzione che, ricevuti come parametri due numeri interi (identificati con b, h) tracci su video un rettangolo di caratteri '*', avente base e altezza, rispettivamente, di b e h asterischi

- Esempio: per b=5 e h=4 occorre visualizzare

```
*****  
*      *  
*      *  
*      *  
*****
```

- Algoritmo: E' sufficiente una coppia di costrutti iterativi annidati per gestire la stampa dei caratteri organizzandoli per righe e colonne

Codice

```
void rettangolo (int b, int h) {
    int i, j;

    for (i=0; i<h; i++) {
        for (j=0; j<b; j++)
            if (i!=0 && i!=h-1 && j!=0 && j!=b-1)
                printf(" ");
            else
                printf("*");
        printf("\n");
    }
}
```

b=5, h=4

```
*****
*       *
*       *
*       *
*****
```

Visualizzazione di una parabola

■ Formulazione:

- Data la parabola di equazione $y = ax^2 + bx + c = 0$ si scriva un programma che:
 - Acquisisca da tastiera i coefficienti a, b, c
 - Acquisisca un intero n ($n > 0$), e i valori degli estremi (x_0, x_n) di un intervallo per le ascisse
 - Acquisisca da tastiera i valori estremi (y_{\min}, y_{\max}) di un intervallo per le ordinate
 - Suddivida l'intervallo $[x_0, x_n]$ in n sotto-intervalli della stessa lunghezza, delimitati dalle ascisse $x_0, x_1, x_2, \dots, x_n$
 - Calcoli i valori di $y(x_i)$ per ognuna delle ascisse $x_i = x_0..x_n$
 - Stampi su file un grafico (con asse delle ascisse verticale) che rappresenti la funzione nel rettangolo del piano cartesiano compreso negli intervalli $[x_0, x_n], [y_{\min}, y_{\max}]$ una riga per ogni valore di ascissa, una colonna per ogni unità sulle ordinate

Visualizzazione di una parabola

- Esempio: se si acquisissero da tastiera i valori:
 - $a=1.0$, $b=2.0$, $c=1.0$, $n=5$, $x_0=0.0$, $x_n=5.0$, $y_{\min}=0.0$, $y_{\max}=50.0$
- Verrebbero calcolati
 - $y(0.0)=1.0$, $y(1.0)=4.0$, $y(2.0)=9.0$, $y(3.0)=16.0$, $y(4.0)=25.0$, $y(5.0)=36.0$
 - Contenuto del file:



Visualizzazione di una parabola

- Struttura dati: sono sufficienti variabili scalari, per rappresentare:
 - Variabile di tipo puntatore a FILE (fpout)
 - Coefficienti: a, b, c (float)
 - Numero di intervalli: n (int)
 - Intervalli: x0, xn, ymin, ymax (float)
 - Dati intermedi: passo (lunghezza degli n intervalli), x, y (float)
 - Contatori per grafico: i, j (int)
- Algoritmo:
 - Input dati e calcolo passo (= lunghezza intervalli)
 - Iterazione su $x=x_0..x_n$
 - Calcolo $y(x)$
 - Se è nell'intervallo $[ymin,ymax]$ converti in intero (j) e visualizza un asterisco dopo j spazi

Codice

```
#include <stdio.h>
#include <math.h>
int main(void) {
    float a,b,c,x,passo,x0,xn,y,ymin,ymax;
    int i, j, n;
    FILE *fpout = fopen("out.txt", "w");
    printf("Coefficienti (a b c): ");
    scanf("%f%f%f",&a,&b,&c);
    printf("Numero di intervalli: ");
    scanf("%d",&n);
    printf("Intervallo per ascisse: ");
    scanf("%f%f",&x0,&xn);
    printf("Intervallo per ordinate: ");
    scanf("%f%f",&ymin,&ymax);
    ...
}
```

```
...
passo = (xn-x0)/n;
for (i=0; i<=n; i++) {
    x = x0 + i*passo;
    y = a*x*x + b*x + c;
    if (y<ymin || y>ymax)
        continue;
    for (j=round(y-ymin); j>0; j--)
        fprintf(fpout, " ");
    fprintf(fpout, "*\n");
}
fclose(fpout);
}
```