Capitolo 15

Schede riassuntive

15.1 Primo programma in C

Struttura di un file sorgente in C

```
/* programma: NomeFile.c
 * autore: NomeAutoreDelProgramma
 * BreveDescrizioneDelProgramma
 */

/* Inclusione delle librerie */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(void)
{
   /* Definizione delle variabili */
   . . . .

   /* Istruzioni eseguibili */
   . . . .

   exit(0) ;
}
```

Nota

Quando il programma riceve degli argomenti sulla linea di comando, allora la definzione della funzione `main` deve essere modificata come:

```
int main(int argc, char *argv[])
```

Librerie principali

Lettura/scrittura su terminale e su file	`#include <stdio.h>`
Interazione con sistema operativo	`#include <stdlib.h>`
Funzioni matematiche	`#include <math.h>`
Elaborazione di testi e stringhe	`#include <string.h>`
Analisi del tipo di caratteri	`#include <ctype.h>`
Valori minimi e massimi	`#include <limits.h>`

Definizione delle variabili

Definizione di variabili intere	`int i, j ;`
Definizione di variabili reali	`float r ;`

Istruzioni eseguibili

Assegnazione a variabile	`a = 0 ;`
	`b = 17 ;`
	`c = a + b ;`
	`b = b + 1 ;`
	`d = b * b - 4 * a * c ;`
	`e = b * (b - 4 * a) * c ;`
	`x1 = ( -b + sqrt(d) ) / ( 2*a ) ;`
	`nomevariabile = espressione ;`

Lettura (input) di numeri interi	`scanf("%d", &i) ;` ⇐ ricordare il '`&`'
Lettura (input) di numeri reali	`scanf("%f", &r) ;` ⇐ ricordare il '`&`'
Stampa (output) di messaggi e numeri	`printf("Numero␣%d,␣valore␣%f\n", i, r) ;`
Vai a capo nella stampa	`printf("\n") ;`

Espressioni aritmetiche

Le 4 operazioni	`+ - * /`
Le parentesi	`( (a + b) * (c / (d - e) ) )`
Resto della divisione	`%`

Funzioni definite in `math.h`

Valore assoluto	$y \leftarrow \lvert x \rvert$	`y = fabs(x) ;`
Radice quadrata	$y \leftarrow \sqrt{x}$	`y = sqrt(x) ;`
Radice cubica	$y \leftarrow \sqrt[3]{x}$	`y = cbrt (x) ;`
Elevamento a potenza	$y \leftarrow x^z$	`y = pow(x, z) ;`
Ipotenusa	$y \leftarrow \sqrt{x^2 + z^2}$	`y = hypot(x, z) ;`
Ceiling	$y \leftarrow \lceil x \rceil$	`y = ceil(x) ;`
Floor	$y \leftarrow \lfloor x \rfloor$	`y = floor(x) ;`
Arrotondamento	$y \leftarrow \lfloor x + 1/2 \rfloor$	`y = round(x) ;`
Troncamento verso 0	$y \leftarrow sign(x) \lfloor \lvert x \rvert \rfloor$	`y = trunc(x) ;`
Resto della divisione	$y \leftarrow Resto(x/z)$	`y = fmod(x, z) ;`
Esponenziale	$y \leftarrow e^x$	`y = exp(x) ;`
	$y \leftarrow 2^x$	`y = exp2(x) ;`
Logaritmo	$y \leftarrow \ln x$	`y = log(x) ;`
	$y \leftarrow \log_2 x$	`y = log2(x) ;`
	$y \leftarrow \log_{10} x$	`y = log10(x) ;`
Funzioni trigonometriche	$y \leftarrow \sin x$	`y = sin(x) ;`
	$y \leftarrow \cos x$	`y = cos(x) ;`
	$y \leftarrow \tan x$	`y = tan(x) ;`
Funzioni trigonometriche inverse	$y \leftarrow \arcsin x$	`y = asin(x) ;`
	$y \leftarrow \arccos x$	`y = acos(x) ;`
	$y \leftarrow \arctan x$	`y = atan(x) ;`
	$y \leftarrow \arctan(x/z)$	`y = atan2(x, z) ;`
Funzioni iperboliche	$y \leftarrow \sinh x$	`y = sinh(x) ;`
	$y \leftarrow \cosh x$	`y = cosh(x) ;`
	$y \leftarrow \tanh x$	`y = tanh(x) ;`
Funzioni iperboliche inverse	$y \leftarrow \sinh^{-1} x$	`y = asinh(x) ;`
	$y \leftarrow \cosh^{-1} x$	`y = acosh(x) ;`
	$y \leftarrow \tanh^{-1} x$	`y = atanh(x) ;`

15.2 Istruzioni di scelta in C

Espressioni condizionali

Confronto di uguaglianza	`==`	⇐ mai usare `=`
Confronto di disuguaglianza	`!=`	
Confronto di ordine	`< <= > >=`	
Congiunzione AND	`(a>0) && (b>0)`	
Disgiunzione OR	`(a>0) || (b>0)`	
Negazione NOT	`!(a+b<c)`	
Appartenenza ad intervalli $x \in [a, b]$	`a<=x && x<=b`	⇐ mai usare `a<=x<=b`
Esclusione da intervalli $x \notin [a, b]$	`x<a || x>b`	⇐ oppure `!(a<=x && x<=b)`

Costrutto `if-else`

Costrutto condizionale semplice	`if (condizione)`
	`{`
	`  istruzioni 1 ;`
	`}`
	`else`
	`{`
	`  istruzioni 2 ;`
	`}`

Costrutto condizionale senza alternativa	`if (condizione)`
	`{`
	`  istruzioni 1 ;`
	`}`

Costrutti `if-else` multipli

Costrutti condizionali sequenziali

```
if ( condizione1 )
{
    istruzioni 1 ;
}
else
{
    istruzioni 2 ;
}

if( condizione2 )
{
    istruzioni 3 ;
}
else
{
    istruzioni 4 ;
}
```

Costrutti condizionali annidati

```
if ( condizione1 )
{
    istruzioni 1 ;
    if( condizione2 )
    {
        istruzioni 2 ;
    }
    else
    {
        istruzioni 3 ;
    }
    istruzioni 4 ;
}
else
{
    istruzioni 5 ;
    if( condizione3 )
    {
        istruzioni 6 ;
    }
    else
    {
        istruzioni 7 ;
    }
    istruzioni 8 ;
}
```

Costrutto condizionale con più alternative

```
if (condizione 1)
{
    istruzioni 1 ;
}
else if (condizione 2)
{
    istruzioni 2 ;
}
else
{
    istruzioni 3 ;
}
```

Costrutto `switch`

Espressione di tipo intero

```
switch (espressione)
{
    case 2:
        istruzioni 1 ;
    break ;

    case 20:
        istruzioni 2 ;
    break ;

    case 210:
        istruzioni 3 ;
    break ;

    default:
        istruzioni di default ;
    break ;
}
```

Espressione di tipo carattere

```
switch (carattere)
{
    case 'a':
    case 'A':
            istruzioni 1 ;
    break ;

    case 'b':
    case 'B':
        istruzioni 2 ;
    break ;

    case 'c':
    case 'C':
        istruzioni 3 ;
    break ;

    case '␣':
        istruzioni 4 ;
    break ;

    case '*':
        istruzioni 5 ;
    break ;

    default:
            istruzioni di default ;
    break ;
}
```

15.3 Cicli ed iterazioni in C

Struttura di un ciclo

1. **Inizializzazione**. Assegnazione del valore iniziale a tutte le variabili che vengono lette durante il ciclo (nella condizione o nel corpo).
2. **Condizione di ripetizione**. Condizione, di solito inizialmente vera, che al termine del ciclo diventerà falsa. Deve dipendere da variabili che saranno modificate all'interno del ciclo (nel corpo o nell'aggiornamento).
3. **Corpo del ciclo**. Le istruzioni che effettivamente occorre ripetere: sono lo scopo per cui il ciclo viene realizzato. Si possono usare e modificare le variabili inizializzate.
4. **Aggiornamento**. Modifica di una o più variabili in grado di aggiornare il valore della condizione di ripetizione (rendendola, prima o poi, falsa). Tengono "traccia" del progresso dell'iterazione.

Operatori di auto-incremento/decremento

Auto-incremento	`i++ ;` `++i ;`	equivale a `i = i + 1 ;`
Auto-decremento	`i-- ;` `--i ;`	equivale a `i = i - 1 ;`

Costrutti iterativi

Costrutto **`while`**	`while(condizione)`
	`{`
	`  corpo ;`
	`}`

Costrutto **`do-while`**	`do`
	`{`
	`  corpo ;`
	`}`
	`while(condizione) ;`

Costrutto **`for`**	`for( inizializzazione; condizione; incremento )`
	`{`
	`  corpo ;`
	`}`

Equivalenza `for-while`

```
for ( inizializz; condiz; aggiornamento )
{
    corpo ;
}
```

```
inizializz ;
while ( condiz )
{
    corpo ;
    aggiornamento ;
}
```

Ciclo infinito

```
for( ; ; )
{
    corpo ;
}
```

```
while( 1 )
{
    corpo ;
}
```

Numero di iterazioni noto a priori

Da 0 a $N-1$, crescente	`for( i=0 ; i<N ; i++)`
	`{`
	`  corpo ;`
	`}`
		`i = 0 ;`
		`while( i<N )`
		`{`
		`  corpo ;`
		`  i++ ;`
		`}`

Da 1 a N, crescente	`for( i=1 ; i<=N ; i++)`
	`{`
	`  corpo ;`
	`}`
		`i = 1 ;`
		`while( i<=N )`
		`{`
		`  corpo ;`
		`  i++ ;`
		`}`

Da $N-1$ a 0, decrescente	`for( i=N-1 ; i>=0 ; i--)`
	`{`
	`  corpo ;`
	`}`
		`i = N-1 ;`
		`while( i>=0 )`
		`{`
		`  corpo ;`
		`  i-- ;`
		`}`

Da N a 1, decrescente	`for( i=N ; i>0 ; i--)`
	`{`
	`  corpo ;`
	`}`
		`i = N ;`
		`while( i>0 )`
		`{`
		`  corpo ;`
		`  i-- ;`
		`}`