

# **COMPUTER SCIENCE: software**

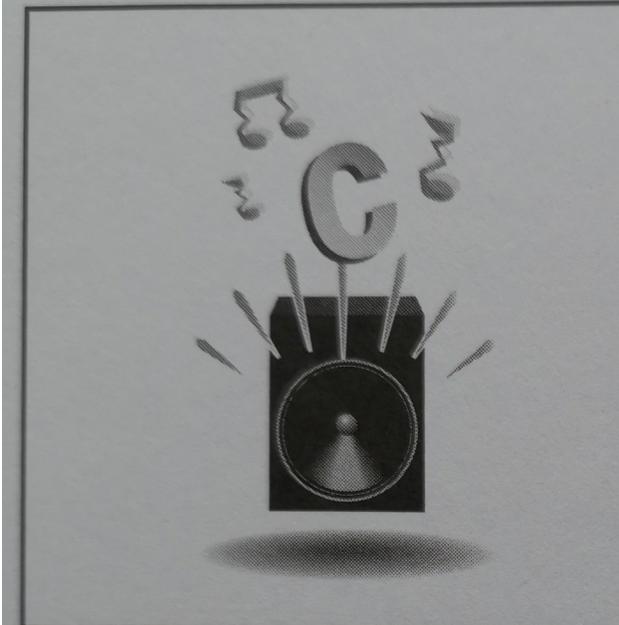
Piero Rivoira

Istituto Agrario Penna – Asti  
[piero.rivoira@yahoo.it](mailto:piero.rivoira@yahoo.it)

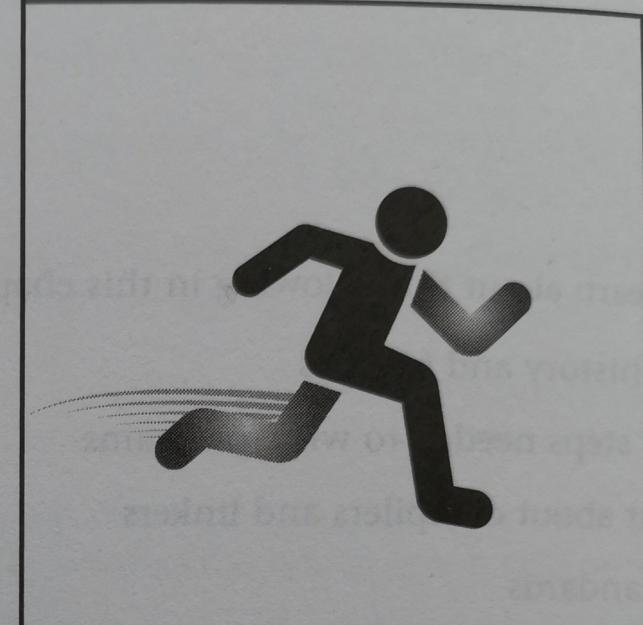
# Il linguaggio C

Le virtù del C:

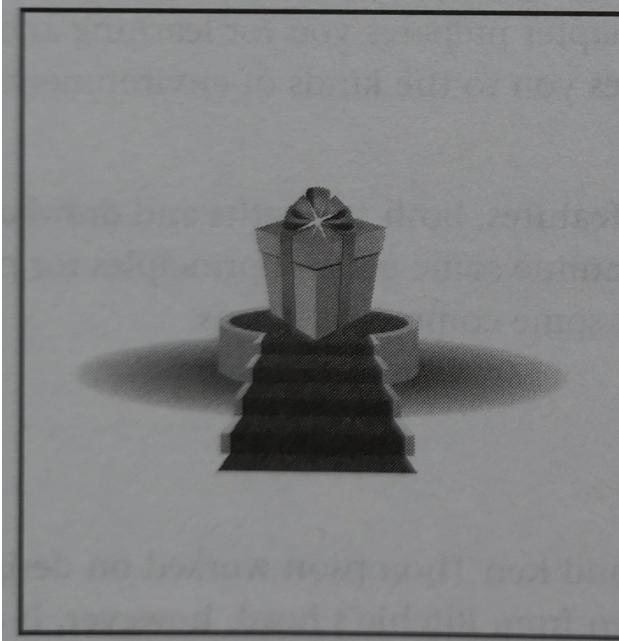
- Velocità ≈ a quella di un linguaggio di tipo assembly (sequenza di istruzioni interne utilizzate da una determinata architettura CPU: CPU di famiglie diverse hanno linguaggi assembly differenti)
- Un programma in C può essere impostato per la max velocità oppure per ottimizzare l'uso della memoria



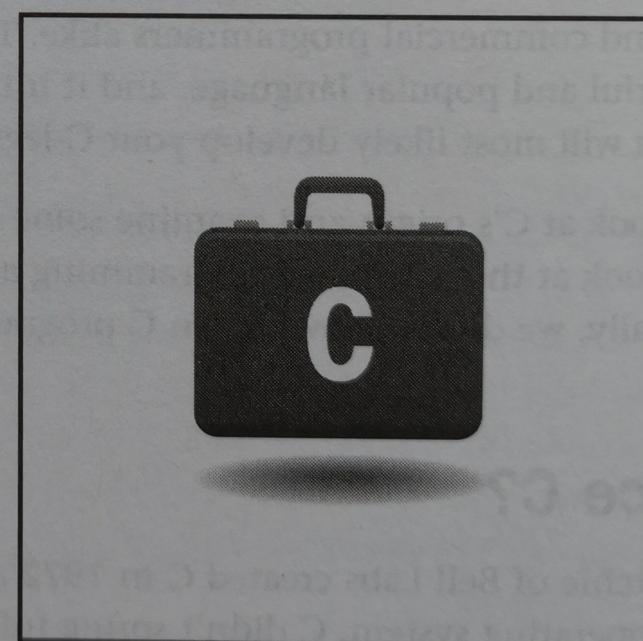
Powerful control structures



Fast



Compact code—small programs

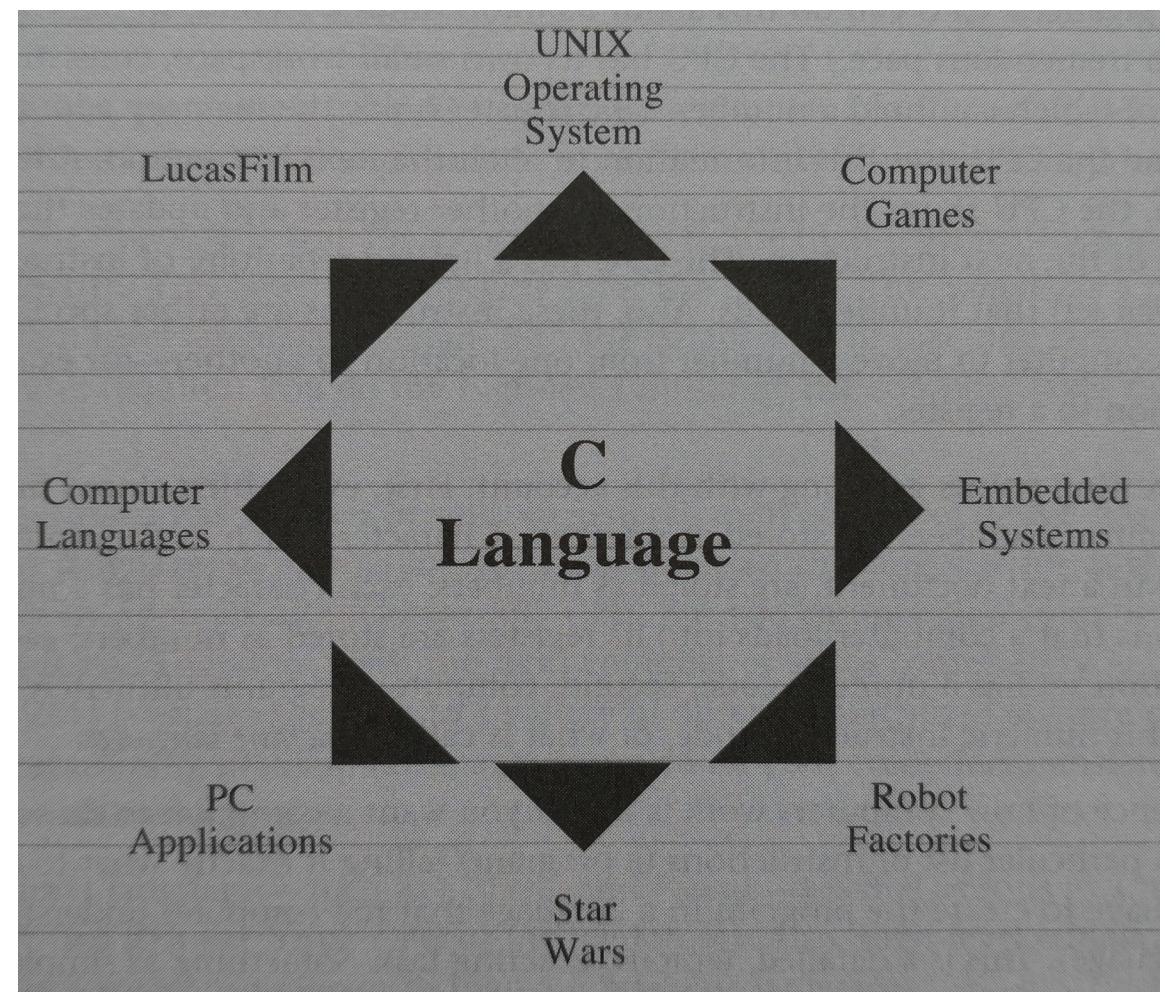


Portable to other computers

# Il linguaggio C

Le virtù del C:

- Portabilità: un programma scritto in C può essere eseguito su architettura diverse con minime modifiche
- Potenza e flessibilità: la > parte dei sistemi operativi della famiglia Unix sono scritti in C, così come gli interpreters di linguaggi interpretati come Perl, Python, Pascal e LISP
- Accesso all'hardware e possibilità di manipolare singoli bits in memoria



# Cosa succede quando un programma viene eseguito

La CPU legge un'istruzione dalla memoria e la esegue, poi legge l'istruzione successiva e la esegue e così via (una CPU gigahertz può effettuare miliardi di cicli di lettura – esecuzione di istruzioni al secondo)

La CPU ha diversi registri di memoria, ognuno dei quali contiene un numero:

- un registro contiene l'indirizzo di memoria dell'istruzione successiva
- l'istruzione appena eseguita viene memorizzata in un altro registro
- Il primo registro viene aggiornato con l'indirizzo dell'istruzione successiva
- La CPU comprende solo un limitato repertorio di istruzioni (*instruction set*)
- Una tipica istruzione consiste nello spostamento di un numero da un indirizzo di memoria ad un registro

# Precisazioni

- Tutto ciò che è memorizzato in un computer lo è sotto forma di numero: i numeri sono immagazzinati come numeri, i caratteri (come le lettere dell'alfabeto, i caratteri speciali ecc.) sono anch'essi immagazzinati come numeri, poiché **ogni carattere è identificato da un codice numerico**

## ASCII printable characters (character code 32-127)

Codes 32-127 are common for all the different variations of the ASCII table, they are called printable characters, represent letters, digits, punctuation marks, and a few miscellaneous symbols. You will find almost every character on your keyboard. Character 127 represents the command DEL.

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
32	040	20	00100000	SP	&#32;		Space
33	041	21	00100001	!	&#33;	&excl;	Exclamation mark
34	042	22	00100010	"	&#34;	&quot;	Double quotes (or speech marks)
35	043	23	00100011	#	&#35;	&num;	Number sign
36	044	24	00100100	\$	&#36;	&dollar;	Dollar
37	045	25	00100101	%	&#37;	&percnt;	Per cent sign
38	046	26	00100110	&	&#38;	&amp;	Ampersand
39	047	27	00100111	'	&#39;	&apos;	Single quote
40	050	28	00101000	(	&#40;	&lparen;	Open parenthesis (or open bracket)
41	051	29	00101001	)	&#41;	&rparen;	Close parenthesis (or close bracket)
42	052	2A	00101010	*	&#42;	&ast;	Asterisk
43	053	2B	00101011	+	&#43;	&plus;	Plus
44	054	2C	00101100	,	&#44;	&comma;	Comma
45	055	2D	00101101	-	&#45;		Hyphen-minus
46	056	2E	00101110	.	&#46;	&period;	Period, dot or full stop
47	057	2F	00101111	/	&#47;	&sol;	Slash or divide
48	060	30	00110000	0	&#48;		Zero
49	061	31	00110001	1	&#49;		One
50	062	32	00110010	2	&#50;		Two
51	063	33	00110011	3	&#51;		Three
52	064	34	00110100	4	&#52;		Four
53	065	35	00110101	5	&#53;		Five
54	066	36	00110110	6	&#54;		Six
55	067	37	00110111	7	&#55;		Seven
56	070	38	00111000	8	&#56;		Eight
57	071	39	00111001	9	&#57;		Nine
58	072	3A	00111010	:	&#58;	&colon;	Colon
59	073	3B	00111011	;	&#59;	&semi;	Semicolon
60	074	3C	00111100	<	&#60;	&lt;	Less than (or open angled bracket)

# Precisazioni

- Le istruzioni che il processore carica nei suoi registri vengono immagazzinate come numeri (ogni istruzione ha il proprio codice numerico)
- Ogni programma (istruzioni, caratteri ecc.) deve, quindi, essere tradotto in una serie di codici numerici detta **linguaggio macchina**
- Se si desidera che il computer faccia una determinata cosa, gli si deve fornire una lista di istruzioni (un programma) che gli dica esattamente cosa fare e come farlo in un linguaggio che il pc sia in grado di comprendere, ossia un linguaggio di basso livello (*low level programming language*)

## Esempio: somma di due numeri

- Copia il numero che si trova nell'indirizzo di memoria 2000 nel registro 1
- Copia il numero che si trova nell'indirizzo di memoria 2004 nel registro 2
- Somma il contenuto del registro 1 al contenuto del registro 2 e memorizza il risultato nel registro 1
- Copia il contenuto del registro 1 nell'indirizzo di memoria 2008

Ognuna delle suddette operazioni è identificata da un codice numerico

# Linguaggi di alto livello

I linguaggi di programmazione di alto livello, come il C, rendono la programmazione un lavoro molto più semplice, divertente e meno noioso:

- Ogni istruzione non dev'essere espressa da un codice numerico
- Le istruzioni che si usano sono molto più simili al modo di pensare umano rispetto al linguaggio macchina: invece di preoccuparsi dei passaggi precisi che la CPU dovrà eseguire per svolgere una determinata operazione, quest'ultima può essere espressa in un modo più astratto. Per sommare due numeri, per es., si scriverà semplicemente:

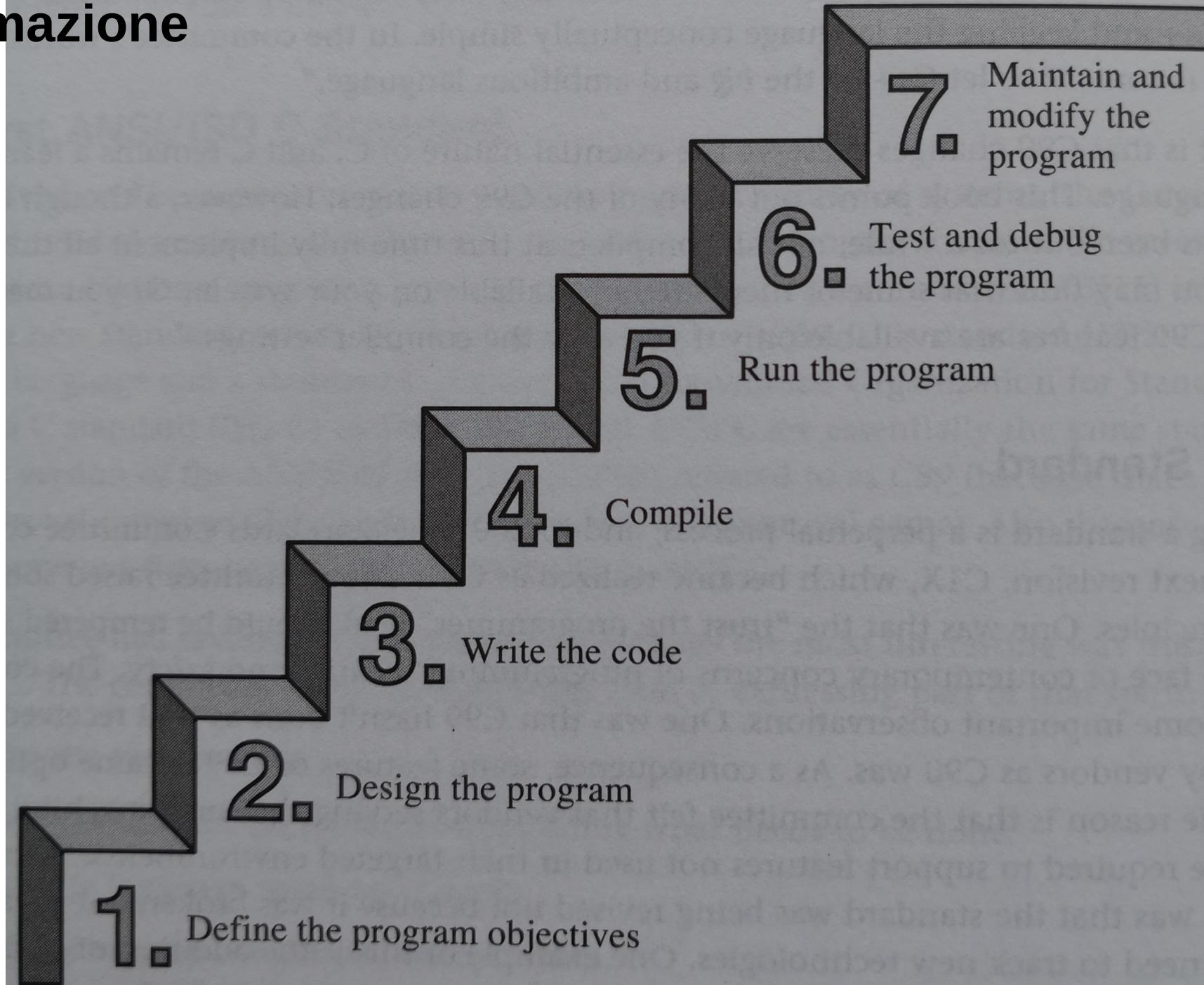
```
totale = mio + tuo;
```

- Un'espressione come questa è del tutto incomprensibile per un pc
- Qui entra in gioco il **compilatore**

# Il compilatore

- Programma che traduce il programma scritto in un linguaggio di alto livello (come il C) nella sequenza dettagliata di istruzioni in linguaggio macchina che il pc è in grado di eseguire
- Ogni architettura ha il proprio linguaggio macchina: un programma scritto in linguaggio macchina per un ARM (Advanced RISC Machine) Cortex-A57 è incomprensibile ad un Intel Core i7, la il compilatore appropriato provvederà a tradurre il nostro programma scritto in C (o in qualsiasi altro linguaggio ad alto livello) nel linguaggio macchina desiderato
- I linguaggi di alto livello, come il C, il C++, Java e Pascal, esprimono le istruzioni in una forma più astratta che non dipende da una particolare CPU e dal suo set di istruzioni

# Le fasi della programmazione



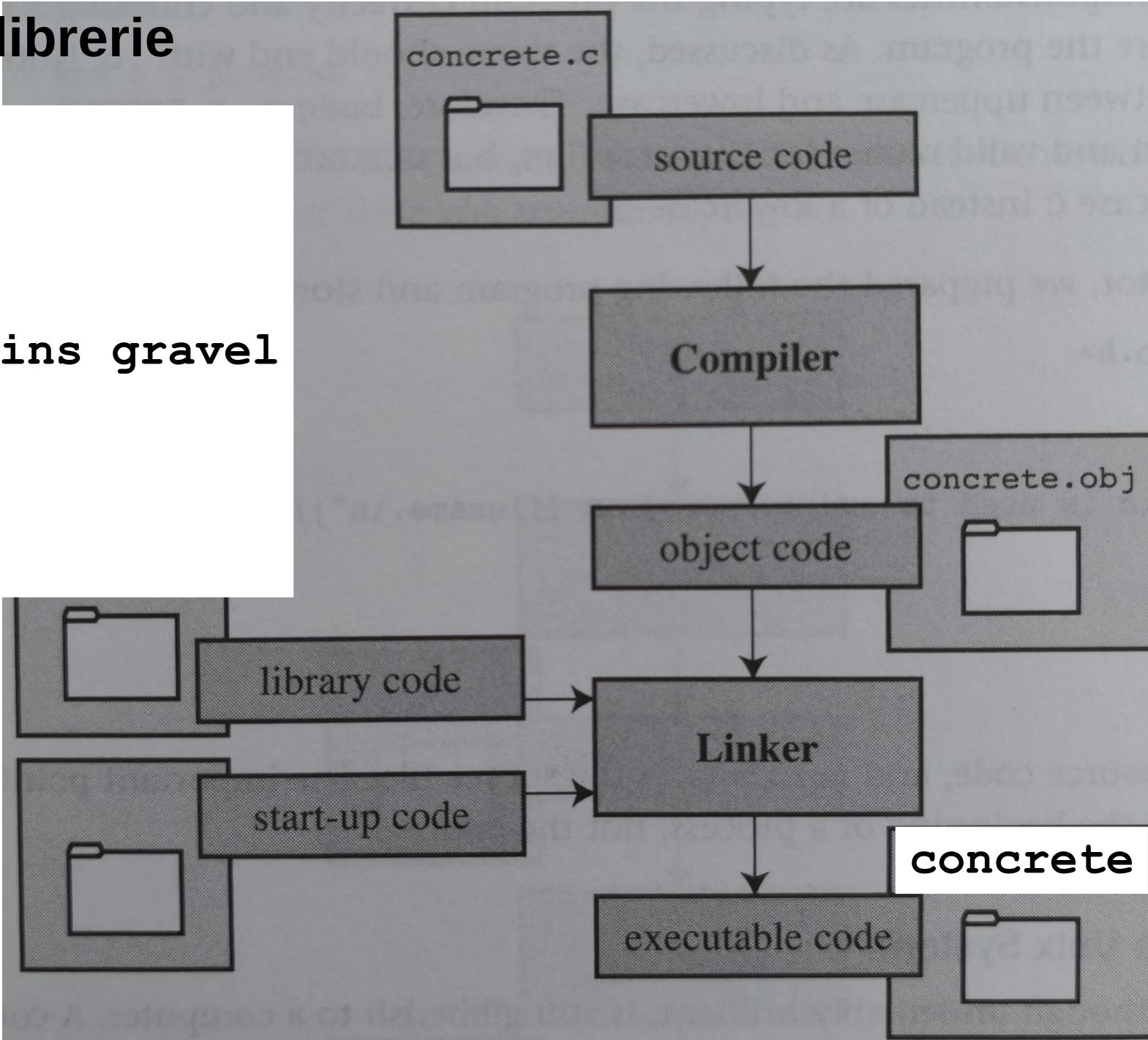
# Files oggetto, eseguibili e librerie

```
#include <stdio.h>
// concrete.c
int main(void)
{
    printf("Concrete contains gravel
and cement.\n");

    return 0;
}
```

2 fasi:

- **Compiling**: conversione del codice sorgente in un file intermedio, il file oggetto
- **Linking**: combina l'oggetto con altri files (librerie ecc.) per creare l'eseguibile



# Il file oggetto

Il file oggetto, pur contenendo codice in linguaggio macchina, non è ancora un programma completo poiché gli mancano due elementi:

- **startup code**: sequenza di istruzioni che fa da interfaccia fra il nostro programma ed il sistema operativo (lo startup code dipende, quindi, dal tipo di sistema operativo)
- **library code**: quasi tutti i programmi scritti in C usano dei blocchi di codice, detti funzioni, che fanno parte della libreria standard del C. Per es., `concrete.c` usa la funzione `printf()`; tuttavia il file oggetto non contiene il codice di questa funzione ma solo la chiamata di essa. Il codice della funzione si trova in un altro file, detto **library** (libreria).

Il ruolo del **linker** è di mettere insieme questi tre elementi, il codice sorgente che abbiamo scritto, il codice di startup del nostro sistema operativo e, infine, il codice delle funzioni chiamate (dal programma), contenuto nelle librerie (da cui il linker estrae solo il codice delle funzioni richieste), in un unico file, l'**eseguibile**

# La formattazione dell'output

```
#include <stdio.h>
// ex3.c
int main()
{
    int age = 10;
    int height = 72;
    printf("I am %d years old.\n", age);
    printf("I am %d inches tall.\n", height);

    return 0;
}
```

# La formattazione dell'output

Includiamo l'header file `stdio.h`, il quale dice al compilatore che il programma userà le funzioni standard di Input/Output, una delle quali è `printf`

Dichiariamo le variabili di tipo `int age e height`, ed assegnamo loro un valore

Chiamiamo la funzione `printf()` per mandare all'output standard (il monitor del pc) l'età e la statura del decenne più alto del pianeta

Nell'argomento di `printf()` includiamo una stringa di formattazione (`%d`) e, fuori dal testo compreso fra le " ", il nome della variabile il valore della quale dovrà essere messo, da `printf`, al posto della stringa di formattazione

# Operatori aritmetici

Operatori aritmetici	
Operatore	Descrizione
+	somma
-	sottrazione
*	moltiplicazione
/	divisione
%	modulo
++	incremento
--	decremento

# Operatori di relazione

## Operatori di relazione

Operatore	Descrizione
<code>==</code>	uguale
<code>!=</code>	diverso
<code>&gt;</code>	maggiore di
<code>&lt;</code>	minore di
<code>&gt;=</code>	uguale o maggiore di
<code>&lt;=</code>	uguale o minore di

# Operatori logici

Operatori logici	
Operatore	Descrizione
&&	AND logico
	OR logico
!	NOT logico
? :	ternario logico

# Operatori di assegnamento

## Operatori di assegnamento

Operatore	Descrizione
=	assegna uguale
+=	assegna uguale o maggiore
-=	assegna uguale o minore
*=	assegna uguale o prodotto
/=	assegna uguale o rapporto
%=	assegna uguale o modulo
<<=	assegna uguale o sposta a sinistra
>>=	assegna uguale o sposta a destra
&=	assegna AND o uguale
^=	assegna XOR o uguale
=	assegna OR o uguale

# Operatori di dati

## Operatori di dati

Operatore	Descrizione
<b>sizeof()</b>	dammi la dimensione di
<b>[ ]</b>	array subscript
<b>&amp;</b>	indirizzo di
<b>*</b>	valore di
<b>-&gt;</b>	structure deference
<b>.</b>	structure reference

# Strutture sintattiche

Parole chiave	
Operatore	Descrizione
<b>auto</b>	assegna ad una variabile locale un contesto locale
<b>break</b>	esci da uno statement
<b>case</b>	caso possibile in uno switch-statement
<b>char</b>	dato di tipo carattere
<b>const</b>	rendi non modificabile una variabile
<b>continue</b>	continua fino alla fine di un loop
<b>default</b>	opzione di default in uno switch-statement
<b>do</b>	inizia un loop di tipo do-while
<b>double</b>	tipo di dato double floating-point
<b>else</b>	opzione else in un if-statement
<b>enum</b>	definisci un insieme di costanti di tipo <b>int</b>
<b>extern</b>	dichiara che il nome di un oggetto è definito esternamente
<b>float</b>	Tipo di dato floating-point

# Strutture sintattiche

## Parole chiave

Operatore	Descrizione
<b>for</b>	
<b>free</b>	
<b>goto</b>	
<b>if</b>	
<b>int</b>	
<b>long</b>	
<b>malloc</b>	
<b>register</b>	
<b>return</b>	
<b>short</b>	
<b>signed</b>	
<b>sizeof</b>	
<b>static</b>	

# Strutture sintattiche

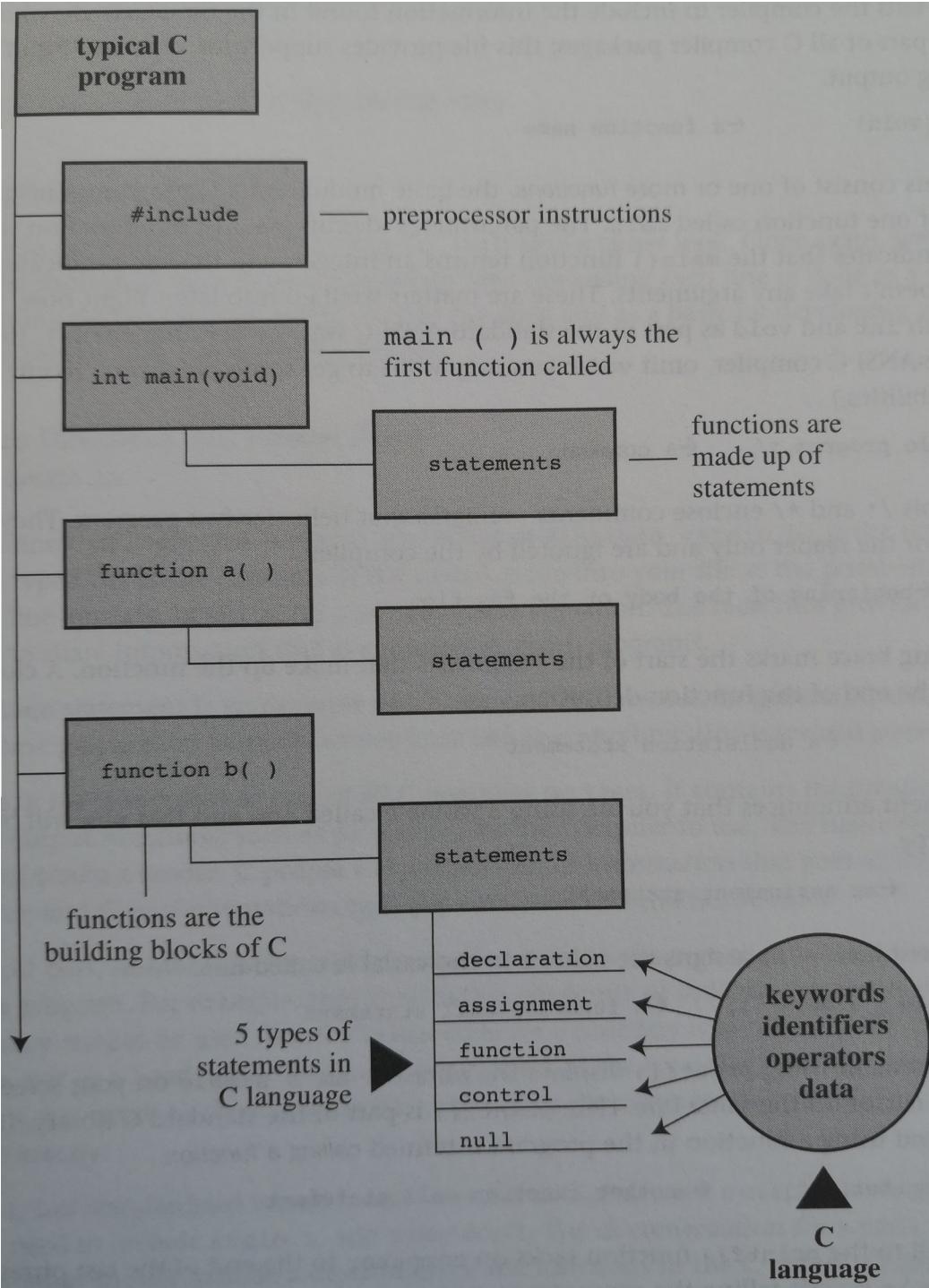
## Parole chiave

Operatore	Descrizione
<code>struct</code>	
<code>switch</code>	
<code>typedef</code>	
<code>union</code>	
<code>unsigned</code>	
<code>void</code>	
<code>volatile</code>	
<code>while</code>	

# Anatomia di un programma in C

```
#include <stdio.h>
// first.c
int main(void)
{
    int num = 1;
    printf("I am a simple computer.\n");
    printf("My favourite number is %d because
it is the first.\n", num);

    return 0;
}
```



# Variabili e tipi di dati

```
#include <stdio.h>
// ex7.c
int main(int argc, char *argv[])
{
    int distance = 100;
    float power = 2.345f;
    double super_power = 56789.4532;
    char first_name[] = "Piero";
    char last_name[] = "Rivoira";
    printf("You are %d miles away.\n", distance);
    printf("You have %f levels of power.\n", power);
    printf("You have %f awesome super powers.\n", super_power);
    printf("I have a first name %s.\n", first_name);
    printf("I have a last name %s.\n", last_name);
    printf("My whole name is %s %s\n", first_name, last_name);
    int bugs = 100;
    double bug_rate = 1.2;
    printf("You have %d bugs at the imaginary rate of %f.\n", bugs, bug_rate);
    long universe_of_defects = 1L * 1024L * 1024L * 1024L;
    printf("The entire universe has %ld bugs.\n", universe_of_defects);
    double expected_bugs = bugs * bug_rate;
    printf("You are expected to have %f bugs\n", expected_bugs);
    double part_of_universe = expected_bugs / universe_of_defects;
    printf("That is only a %e portion of the universe\n", part_of_universe);
    int nul_byte = 0;
    int care_percentage = bugs * nul_byte;
    printf("Which means you should care %d%%.\n", care_percentage);
    return 0;
}
```

# Variabili e tipi di dati

```
/* platinum.c -- your weight in platinum */
#include <stdio.h>
int main(void)
{
    float weight;          /* user weight in kg */
    float weight_pounds;   /* user weight in pounds */
    float value;           /* platinum equivalent */

    printf("Are you worth your weight in platinum?\n");
    printf("Let's check it out.\n");
    printf("Please enter your weight in kilos with eight decimal digits: ");

    /* get input from the user */
    scanf("%8f", &weight);
    printf("Your weight is %.8f kilos.\n", weight);
    weight_pounds = weight * 2.20462262185;
    printf("Your weight is %.8f pounds.\n", weight_pounds);
    printf("Current platinum's price is 29.39 €/g.\n");
    value = weight * 29390.00;
    printf("Your weight in platinum is worth €%.8f.\n", value);
    printf("You are easily worth that! If platinum prices drop,\n");
    printf("eat more to maintain your value.\n");

    return 0;
}
```

# Variabili e tipi di dati

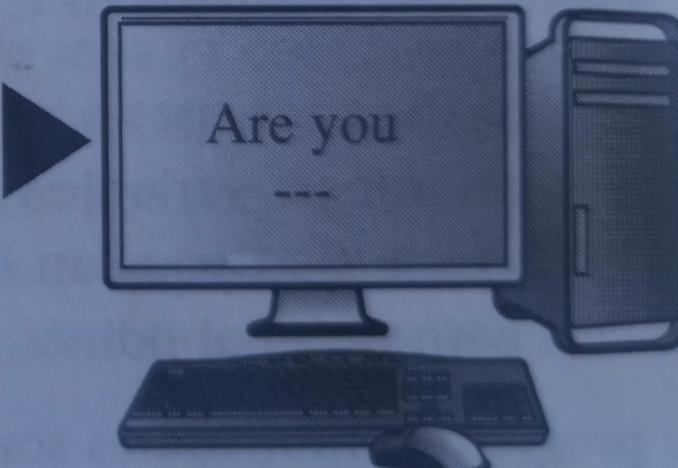
## Body

```
/*platinum.c*/  
.  
.  
.  
int main(void)  
{  
.  
.  
.  
.  
.  
.  
scanf("-----")  
.  
.  
.  
.  
.  
printf("Are you--")  
printf("-----")  
.  
.  
return 0;  
}  
.
```

getting keyboard input



displaying program output



# Variabili e tipi di dati

```
/* bases.c -- prints a whole number in decimal, octal, and hex */
#include <stdio.h>
int main(void)
{
    int value;

    printf("Please enter a whole number.\n");

    /* get input from the user */
    scanf("%d", &value);
    printf("The number you entered is %d.\n", value);
    printf("dec = %d; octal = %o; hex = %x\n", value, value, value);
    printf("dec = %d; octal = %#o; hex = %#x\n", value, value, value);

    return 0;
}
```

# Multiple Integer Types

Tipo	N° di bits	Numero massimo
<code>short</code>	16	<pre>&gt;&gt;&gt; print(bin(32767)) 0b1111111111111111</pre>
<code>int</code>	16÷32	<pre>&gt;&gt;&gt; print(bin(-32767)) -0b1111111111111111</pre>
<code>unsigned short</code> <code>unsigned int</code>	16	<pre>&gt;&gt;&gt; print(bin(65535)) 0b1111111111111111</pre>
<code>long</code>	32	<pre>&gt;&gt;&gt; print(bin(2147483647)) 0b11</pre>
<code>unsigned long</code>	32	<pre>&gt;&gt;&gt; print(bin(4294967295)) 0b11</pre>
<code>long long</code>	64	<pre>&gt;&gt;&gt; print(0b11) 9223372036854775807</pre>
<code>unsigned long long</code>	64	<pre>&gt;&gt;&gt; print(0b11) 18446744073709551615</pre>

# Multiple Integer Types

```
/* defines.c -- uses defined constants from limit.h and float */
#include <stdio.h>
#include <limits.h> // integer limits
#include <float.h> // floating-point limits
int main(void)
{
    printf("Some number limits for this system:\n");
    printf("Biggest int: %d\n", INT_MAX);
    printf("Smallest long long: %lld\n", LLONG_MIN);
    printf("Biggest long: %ld\n", LONG_MAX);
    printf("One byte = %d bits on this system.\n", CHAR_BIT);
    printf("Largest double: %e\n", DBL_MAX);
    printf("Smallest normal float: %e\n", FLT_MIN);
    printf("float precision = %d digits\n", FLT_DIG);
    printf("float epsilon = %e\n", FLT_EPSILON);

    return 0;
}
```

# Escape sequences

```
/* escape.c - use escape characters */
#include <stdio.h>
int main(void)
{
    float salary;

    printf("Please enter a whole number.\n");

    printf("\aEnter your desired monthly salary:");
    printf(" $_____ \b\b\b\b\b\b\b\b");
    scanf("%f", &salary);
    printf("\n\t%.2f a month is $%.2f a year.", salary, salary * 12.0);
    printf("\rGee!\n");

    return 0;
}
```

# Character strings

```
/* talkback.c -- nosy, informative program */
#include <stdio.h>
#include <string.h>          // for strlen() prototype
#define DENSITY 62.4 // human density in lbs per cu ft
int main()
{
    float weight, volume;
    int size, letters;
    char name[40];           // name is an array of 40 chars
    printf("Hi! What's your first name?\n");
    scanf("%s", name);
    printf("%s, what's your weight in kilograms?\n", name);
    scanf("%f", &weight);
    weight = weight * 2.20462262185;
    printf("%s, your weight is %f pounds!\n", name, weight);
    size = sizeof name;
    letters = strlen(name);
    printf("%s, your name is %d letters long\n", name, letters);

    return 0;
}
```

# Funzioni multiple

```
#include <stdio.h>
// two_func.c
void butler(void);
int main(int argc, char *argv[])
{
    printf("I will summon the butler function.\n");
    butler();
    printf("Yes. Bring me some tea and writeble DVDs.\n");
    return 0;
}

void butler(void)
// definizione della funzione butler()
{
    printf("You rang, sir?\n");
}
```

# If, Else-if, Else

SINTASSI

```
if(TEST)
{
    CODE;
}
else if(TEST)
{
    CODE;
}
else
{
    CODE;
}
```

# If, Else-if, Else

```
#include <stdio.h>
// ex8.c
int main(int argc, char *argv[])
{
    int i = 0;
// dichiara il contatore i ed inizializzalo a zero
    if (argc == 1)
// se la funzione main ha un solo argomento
    {
        printf("You have only one argument. You suck.\n");
    }
    else if (argc > 1 && argc < 4)
// se il numero degli argomenti è compreso fra 1 e 4; && -> AND logico
    {
        printf("Here's your arguments:\n");

        for (i = 0; i < argc; i++)
// partendo da i = 0, stampa in output l'argomento[0] della funzione main (il primo dell'array argv[0])
// incrementa il contatore i di 1 (uno) e stampa in output il secondo argomento (argv[1])
// ripeti finché i < del numero degli argomenti
        {
            printf("%s ", argv[i]);
        }
        printf("\n");
    }
else
{
    printf("You have too many arguments. You suck.\n");
}
return 0;
}
```

# While-Loop

```
#include <stdio.h>
// ex9.c
int main(int argc, char *argv[])
{
    int i = 0;
// dichiara il contatore i ed inizializzalo a zero
    while (i < 25)
// fino a quando i è minore di 25
    {
        printf("%d\n", i);
        i++;
    }

    return 0;
}
```

# For-Loop

```
#include <stdio.h>
// ex10.c
int main(int argc, char *argv[])
// la funzione main ha due argomenti:
// 1) argc → N° di argomenti che l'utente inserisce (tipo int)
// 2) *argv[] → nomi degli argomenti stessi (array di stringhe di caratteri, ognuna delle quali può essere letta singolarmente tramite un indice
// compreso fra parentesi quadre
{
    int i = 0;
// go through each string in argv
// why am I skipping argv[0]?

for (i = 1; i < argc; i++)
{
    printf("arg %d: %s\n", i, argv[i]);
}
// let's make our own array of strings
char *states[] = {"California", "Oregon", "Washington", "Texas"};
int num_states = 4;
for (i = 0; i < num_states; i++)
{
printf("state %d: %s\n", i, states[i]);
}

return 0;
}
```

# La funzione `strlen()`

```
/* praise2.c */
#include <stdio.h>
#include <string.h>
// l'header file <string.h> contiene diverse funzioni utili per manipolare stringhe di caratteri, come strlen(),
#define PRAISE "You are an extraordinary being."
int main(void)
{
    char name[20];
// dichiariamo l'array name, formato da 20 celle di memoria, come ci dice l'operatore sizeof.
// di queste solo 5 sono necessarie per contenere il nome Piero; la sesta contiene il carattere null, la presenza del quale
// dice alla funzione strlen() quando smettere di contare
```



```
printf("What's your name?\n");
scanf("%s", name);
printf("Hello %s. %s\n", name, PRAISE);
printf("Your name of %zd letters occupies %zd memory cells.\n", strlen(name), sizeof name);
printf("The phrase of praise has %zd letters", strlen(PRAISE));
printf("and occupies %zd memory cells.\n", sizeof PRAISE);

return 0;
}
```

# Le costanti

```
/* pizza.c */
#include <stdio.h>
#include <math.h>
#define PI 3.141593
int main(void)
{
    float area, circum, radius, square_radius;
    printf("What is the radius of your pizza?\n");
    scanf("%f", &radius);
//    square_radius = powf(10.0, 2.0);
//    area = PI * square_radius;
    area = PI * radius * radius;
    circum = 2.0 * PI * radius;
    printf("Your basic pizza parameters are as follows:\n");
    printf("circumference = %1.2f, area = %1.2f\n", circum, area);

    return 0;
}
```

## Il modificatore \* con printf() e scanf()

```
/* varwid.c - uses variable-width output field */
#include <stdio.h>
int main(void)
{
    unsigned width, precision;
    int number = 256;
    double weight = 242.5;
    printf("Enter a field width:\n");
    scanf("%d", &width);
    printf("The number is :%*d:\n", width, number);
    printf("Now enter a width and a precision:\n");
    scanf("%d %d", &width, &precision);
    printf("Weight = %.*f", width, precision, weight);
    printf("Done!\n");

    return 0;
}
```

# Exponential growth

```
/* wheat.c – exponential growth */
#include <stdio.h>
#define SQUARES 64          // squares on a checkerboard
int main(void)
{
    const double CROP = 2E16;      // world wheat production in wheat grains
    double current, total;
    int count = 1;
    printf("square      grains      total      ");
    printf("fraction of \n");
    printf("           added      grains      ");
    printf("world total\n");
    total = current = 1.0;        // start with one grain
    printf("%4d %13.2e %12.2e %12.2e\n", count, current, total, total/CROP);
    while (count < SQUARES)
    {
        count = count + 1;
        current = 2.0 * current;      // double grains on next square
        total = total + current;      // update total
        printf("%4d %13.2e %12.2e %12.2e\n", count, current, total, total/CROP);
    }
    printf("Done!\n");

    return 0;
}
```

# Switch Statements

Solo espressioni che producano come risultato numeri interi, utilizzati per saltare dalla cima dello switch alla parte corrispondente al risultato

```
#include <stdio.h>
// ex .c
int main(int argc, char *argv[])
{
    int i = 0;

    return 0;
}
```