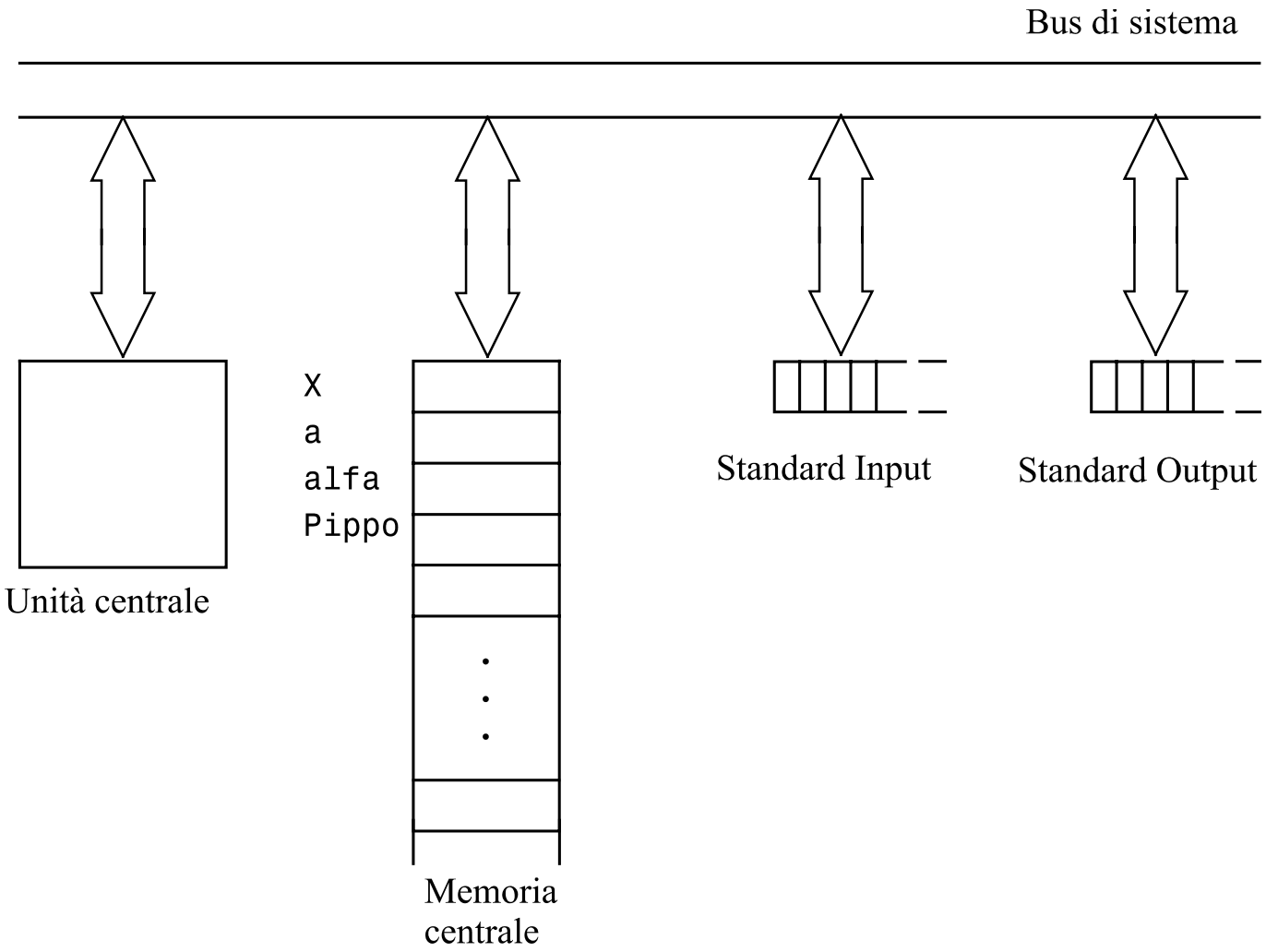
La macchina astratta del C

Terminologia ed elementi essenziali

- **Standard Input**
- **Standard Output**
- **Stringa**
- **Celle di memoria o variabili**
- **Identificatori simbolici**
- **Identificatori predefiniti e riservati**
- **Parole chiave**



Struttura sintattica di un programma C

Se per ora trascuriamo alcuni componenti (**direttive** o *macroistruzioni* per il pre-compilatore C e **dichiarazioni** delle costanti e delle variabili) che esamineremo in seguito, un programma C è composto da:

⇨ un'**intestazione** seguita da

⇨ una sequenza di **istruzioni** racchiusa tra i simboli **{** e **}**.

L'**intestazione** è costituita dall'identificatore predefinito **`main`** seguito da una coppia di parentesi **( )** (per il momento vuote)

Le **istruzioni** sono *frasi* del linguaggio di programmazione; ognuna di esse termina con il simbolo **;**

Un **Commento**, che non è influente dal punto di vista esecutivo perchè non considerato dal programma di traduzione, è delimitato dai simboli **`/*`** e **`*/`**

In questo caso il commento può estendersi su più linee e apparire in qualsiasi parte del programma.

Alternativamente

```
// commento
```

in questo caso il commento può estendersi su una sola linea.

L'esecuzione inizia da **`main()`**

Il *corpo del programma* è racchiuso tra parentesi graffe **`{`** e **`}`**

Ogni istruzione deve terminare con un carattere di punto e virgola
Il programma è composto da una serie di **istruzioni** che verranno eseguite *in sequenza*.

```
/*Calcolo area rettangolo*/
#include <stdio.h>
 main()
{ int base; int altezza; int area;

  base = 3;
  altezza = 7;
  area = base*altezza;

  printf("%d\n", area);
}
```

Il C distingue tra lettere maiuscole e minuscole.

Se si scrive **`MAIN()`** o **`Main()`** non si fa riferimento a **`main()`**

Le principali istruzioni del C

Istruzione di assegnazione

```
x = 23;
w = 'a';
y = z;
r3 = (alfa*43-xgg)*(delta-32*ijj);
x = x+1;
```

Istruzioni composte

Istruzioni di ingresso e uscita o I/O (scanf e printf)

La dichiarazione di costanti e variabili

Tutto ciò che viene usato va dichiarato. In prima istanza:
Dichiarazione delle variabili;
Dichiarazione delle costanti.

Perché questa fatica?
Aiuta la **diagnostica** (ovvero *segnalazione di errori*):

```
x = alfa;
alba = alfa + 1;
```

– Senza dichiarazione, alba è una nuova variabile e se non è stata dichiarata sarà segnalato un errore!

Principio importante: meglio un po' più di fatica nello scrivere un programma che la maggiore fatica nel leggerlo e capirlo!

Una dichiarazione di variabili consiste in:
– uno **specificatore di tipo**, seguito da una lista di uno o più **nomi** o **identificatori di variabili** separati da una virgola **,**
– il carattere **;** per indicare il termine della dichiarazione

Le variabili possano essere dichiarate

- prima di **`main()`** e la parentesi graffa aperta
  parte dichiarativa globale (global declarative part), che contiene la dichiarazione di tutti gli elementi che sono *condivisi* dal programma principale e dai sottoprogrammi;
- dopo **`main()`**
  parte dichiarativa locale (del main) (local declarative part), che elenca tutti gli elementi che fanno parte del programma, con le loro principali caratteristiche.

Perciò la struttura generale di un programma in C è la seguente

```
direttive al pre-compilatore C
dichiarazione di variabili globali
main
{
dichiarazione di variabili locali
 istruzione1
 istruzione2
 istruzione3
 ...
 istruzioneN
}
```

Dichiarazioni di variabili

Il ***nome*** di una variabile la identifica.
il suo ***tipo*** ne definisce la dimensione e l'insieme delle operazioni che si possono effettuare su di essa.
La ***dimensione*** può variare rispetto all'implementazione.

Dichiarazioni di variabili di tipo intero

```
int base;
int altezza;
int area;
```

oppure

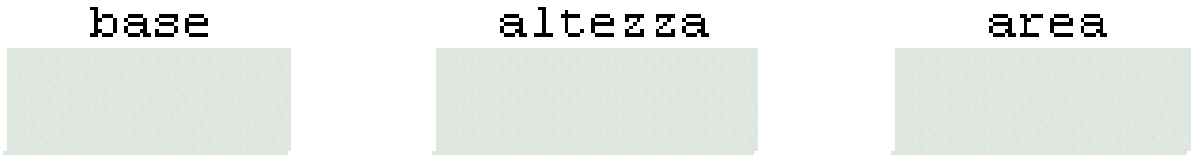
```
int base, altezza, area;
```

`int` specifica che si tratta di una variabile numerica di tipo *intero*.

Dichiarazione e inizializzazione

```
int base = 3, altezza = 7, area;
```

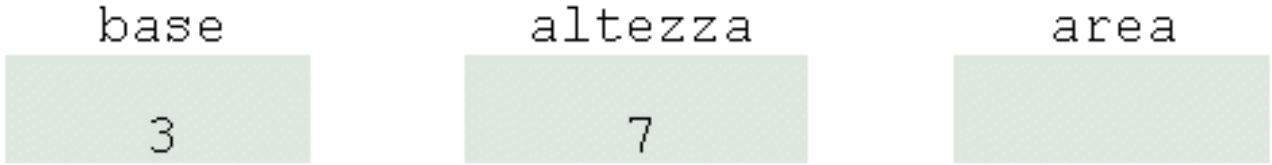
Nell'esempio, a ogni dichiarazione di variabile corrisponde anche la sua *dimensione*: le viene cioè riservato uno spazio adeguato in memoria centrale:



`base = 3;`

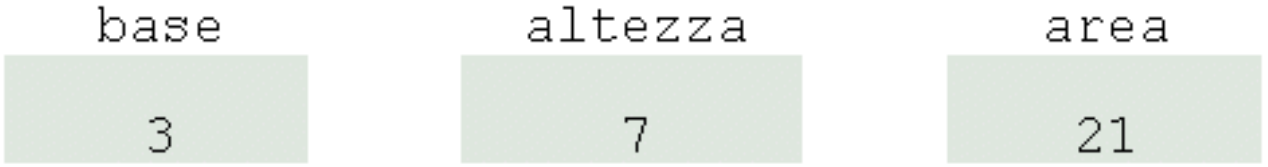
assegna alla variabile **`base`** il valore 3.

L'***assegnazione*** è dunque realizzata mediante l'operatore **=**



`area = base*altezza;`

assegna alla variabile **`area`** il prodotto dei valori di **`base`** e **`altezza`**.



Dichiarazioni di costanti

Una costante valore può essere rappresentata con un nome simbolico attraverso la direttiva **#define** al pre-compilatore C

```
#define BASE 3
```

BASE non può essere modificata

Per convenzione utilizziamo caratteri maiuscoli per le costanti
Per calcolare l'area del rettangolo con base 102 e altezza 34, dobbiamo modificare solo le **define**:

```
#define BASE 102
#define ALTEZZA 34
```

La dichiarazione dei dati, e quindi la dichiarazione del loro tipo, consente di conoscere:

- l'insieme dei valori ammissibili
- l'insieme delle operazioni applicabili
- la quantità di memoria necessaria
- gli eventuali errori d'uso.

Operazioni per dati di tipo **int**

=	Assegnamento di un valore **int** a una variabile **int**
+	Somma (tra **int** ha come risultato un **int**)
-	Sottrazione (tra **int** ha come risultato un **int**)
*	Moltiplicazione (tra **int** ha come risultato un **int**)
/	Divisione con troncamento della parte non intera (risultato **int**)
%	Resto della divisione intera
==	Relazione di uguaglianza
!=	Relazione di diversità
<	Relazione "minore di"
>	Relazione "maggiore di"
<=	Relazione "minore o uguale a"
>=	Relazione "maggiore o uguale a"

I nomi o identificatori delle variabili

Gli identificatori o nomi delle variabili:

- devono iniziare con una lettera o con un carattere di sottolineatura _
- possono contenere lettere, cifre e _
- un identificatore non può essere una parola chiave del linguaggio o un nome di funzione

Le principali istruzioni del C

Istruzioni composte

compound statement o blocco

```
{z = x + 1; y = 13 + x;}
```

Istruzione condizionale (selezione)

```
if(x == 0) z = 5; else y = z + w*y;
if(x == 0) {z = 5;} else {y = z + w*y;}
if ((x+y)*(z-2) > (23+v)) {z = x + 1; y = 13 + x;}
if ((x == y && z >3) || w != y) z = 5; else {y = z + w*y; x = z;}
```

Istruzioni scorrette:

```
if (x == 0) else y = z; y = 34;
if (x == 0) a; else b + c;
```

Istruzione iterativa (ciclo o loop)

```
while (x >= 0) x = x - 1;
while (z != y) {y = z - x; x = x*3;}
```

Espressioni condizionali

```
x == 0
(x+y)*(z-2) > (23+v)
(x == y && z >3) || w != y
x >= 0
z != y
```

Istruzioni di ingresso e uscita o I/O (scanf)

Ogni programma che utilizza al suo interno le istruzioni di I/O deve dichiarare l'uso di tali funzioni nella parte direttiva che precede il programma principale

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

stdio (*standard input/output*) è la libreria standard delle funzioni che consentono in C di leggere e scrivere.

#include è una **direttiva** data ad una parte del compilatore, chiamata *preprocessore*

Per immettere/leggere valori:

```
scanf("%d", &x);
```

%d indica che s'intende immettere un valore intero in formato decimale. **&x** indica l'indirizzo di memoria della variabile **x** in cui dovrà essere immesso il valore intero.

Istruzioni di ingresso e uscita o I/O (printf)

Per visualizzare/stampare valori:

```
printf("%d\n", y);
```

printf stampa ciò che è racchiuso tra parentesi tonde e doppi apici.

% specifica che il carattere seguente (**d**) definisce il formato di stampa di **y**; **d** (decimal) indica il formato di stampa: un intero nel sistema decimale. **\n** tra doppi apici provoca un salto a linea nuova dopo la visualizzazione. In effetti, la sequenza **\n** corrisponde a un solo carattere, quello di linea nuova (newline).

N.B.

Le istruzioni printf successive alla prima iniziano a scrivere a partire dalla posizione del video che segue quella occupata dall'ultimo carattere visualizzato dalla printf immediatamente precedente.

Istruzioni di ingresso e uscita o I/O (printf)

Per visualizzare più variabili con una sola **printf**

```
printf("%d %d %d", base, altezza, area);
```

Per visualizzare degli a-capo a piacere:

```
printf("%d\n%d\n%d", base, altezza, area);
```

Per visualizzare dei commenti:

```
printf("Base: %d\nAltezza: %d\nArea: %d", base, altezza, area);
```

Per inserire delle espressioni:

```
printf("Area: %d", 10*13);
printf("Area: %d", base*altezza);
```

Per definire il numero di caratteri riservati per un valore:

```
printf("%5d%5d%5d", base, altezza, area);
```

%5d riserva un campo di cinque caratteri per il valore, che sarà sistemato a cominciare dalla destra di ogni campo (**-5d%** da sinistra).

Istruzioni di ingresso e uscita o I/O (printf)

Sequenze di escape, sequenze di caratteri con funzioni speciali:

`\n`	va a linea nuova
`\t`	salta di una tabulazione
`\b`	ritorna un carattere indietro (*backspace*)
`\a`	suona il campanello della macchina
`\\`	stampa il carattere `\`
`\"`	stampa il carattere `"`

I primi, semplici, programmi in C

```
/*Programma NumeroMaggiore – prima versione */
  #include <stdio.h>
  main()
{   int x, y, z;
  scanf("%d", &x);
  scanf("%d", &y);
  if (x > y) z = x; else z = y;
  printf("il numero più grande tra %d e %d è: %d", x, y, z);
}
```

```
/*Programma NumeroMaggiore – seconda versione */
  #include <stdio.h>
  main()
{   int x, y;
  scanf("%d", &x);
  scanf("%d", &y);
  if (x > y) printf("il numero più grande tra %d e %d è: %d", x,y,x);
  else printf("il numero più grande tra %d e %d è: %d", x,y,y);
}
```

I primi, semplici, programmi in C

```
/* ProgrammaCercaIlPrimoZero */
    #include <stdio.h>
    main()
{       int uno, dato;
    uno = 1;
    scanf (“%d”, &dato);
    while (dato !=0) scanf(“%d”, &dato);
    printf(“trovato zero %d\n”, uno);
}

/* Programma SommaSequenza */
    #include <stdio.h>
    main()
{       int numero, somma;
    somma = 0;
    scanf(“%d”, &numero);
    while (numero != 0)
        {
        somma = somma + numero;
        scanf(“%d”, &numero);
        }
        printf(“\n somma dei numeri letti %d”, somma);
}
```

I primi, semplici, programmi in C

```
/* Programma per la valutazione di un triangolo */
#include <stdio.h>
main()
{       int X, Y, Z;
   /*Lettura dei dati di ingresso */
    scanf(“%d”, &X); scanf(“%d”, &Y); scanf(“%d”, &Z);
   /* Verifica che i dati possano essere le lunghezze dei lati
            di un triangolo */
    if ((X < Y + Z) && (Y < X + Z) && (Z < X + Y))
       /*Distinzione tra i vari tipi di triangolo */
       if (X == Y && Y == Z)
       printf("I dati letti corrispondono a un triangolo equilatero");
       else
          if (X == Y || Y == Z || X == Z)
          printf("I dati letti corrispondono a un triangolo isoscele");
          else
          printf("I dati letti corrispondono a un triangolo scaleno");
    else
    printf("I dati letti non corrispondono ad alcun triangolo");
}
```

Nota sull'istruzione if

```
if (C1) if (C2) S1; else S2;
```

l'istruzione `else S2;` è diramazione del primo o del secondo `if` ?

Convenzione: il primo ramo `else` viene attribuito all'ultimo `if`. Altrimenti, scriviamo esplicitamente:

```
if (C1) {if (C2) S1;} else S2;
```