Astrazione in C: la funzione

- ⟨tipo-risult⟩ indica il tipo del risultato della funzione. Può essere
  - `void`: assenza di risultato, cioè la funzione rappresenta una procedura
  - un altro tipo ad eccezione del tipo array
- ⟨param-formali⟩ rappresenta i parametri formali della funzione
  - `void` indica assenza di parametri
  - ogni parametro è indicato con tipo e nome

Il blocco della funzione

- La parte ⟨dichiarazioni⟩ del blocco è detta *parte dichiarativa locale*; la parte ⟨istruzioni⟩ è detta anche *parte esecutiva*, o *corpo della funzione*
- La parte esecutiva spesso comprende una istruzione `return` con la sintassi

  ⟨istr-return⟩ $\longrightarrow$ **`return`** ⟨espr⟩

  dove ⟨espr⟩ ha il tipo ⟨nome-tipo-risult⟩
- `return` restituisce il risultato della funzione come valore di ⟨espr⟩ e restituisce immediatamente il controllo alla chiamante; pertanto, se è eseguita, è sempre l'ultima istruzione eseguita dalla funzione
- Il valore assunto da ⟨espr⟩ viene comunque convertito a ⟨nome-tipo-risult⟩

Chiamata delle funzioni

- Sintassi

$$\langle \text{chiamata-funzione} \rangle \longrightarrow \langle \text{nome-funz} \rangle \texttt{(} \langle \text{param-attuali} \rangle \texttt{)}$$
$$\mid \langle \text{nome-funz} \rangle \texttt{()}$$
$$\langle \text{param-attuali} \rangle \longrightarrow \langle \text{espr} \rangle \mid \langle \text{espr} \rangle \texttt{,} \langle \text{param-attuali} \rangle$$

- Il collegamento tra parametri attuali e parametri formali
  - è stabilito secondo l'ordine di comparizione
  - è sempre per valore
- $\langle \text{chiamata-funzione} \rangle$ è una $\langle \text{espr} \rangle$: quindi può essere parametro attuale—è consentita la composizione di funzioni

Chiamata delle funzioni

Esempio.

```
int MinimoInteri(int i, int j) {
/* I, J "parametri formali" */
  if (i<j)
    return i;
  else
    return j;
}
main() { /* chiamante */
  int M;
  int A=5, B=6;
  M = MinimoInteri(A,B);
  /* la funzione MinimoInteri e' la chiamata
  A, B sono parametri attuali, compatibili per assegnamento con i
  parametri formali */
}
```

Chiamata delle funzioni

- Una chiamata di funzione è una espressione → può essere usata come parametro attuale nella chiamata di un'altra funzione, ad esempio `printf`

```
printf("%d",MinimoInteri(A,B);
```

oppure della stessa `MinimoInteri`

```
main() {
  int M;
  int A=5, B=6;
  M = MinimoInteri(A,MinimoInteri(B,3));
}
```

Chiamata delle funzioni

Esempio. Scrivere una funzione per il calcolo del prodotto di due numeri interi non negativi X e Y utilizzando l'operatore $+$

- *inizializza P a 0*
- *inizializza W a X*
- *finché W è diverso da 0*
  - *P=P+Y*
  - *W=W-1*
- *restituisci P*

```
int Prodottolnteri1(int X, int Y) {
  int P=0; /* X deve essere >= 0 */
  int W=X;
  while (W>0) {
    P = P + Y;
    W = W -1;
  }
  return P;
}
```

- X, Y non sono modificati nel corpo della funzione

Chiamata delle funzioni

- Non utilizziamo la variabile di appoggio `W` ma decrementiamo direttamente `X`

```
                                int ProdottoInteri2(int X, int Y) {
- inizializza P a 0                 int P=0;
- finché X è diverso da 0           while (X>0) {
  - P=P+Y                             P = P + Y;
  - X=X-1                             X = X -1;
                                    }
  - restituisci P                   return P;
                                }
```

Chiamata delle funzioni

Esempio. Visualizzare il minimo di tre interi acquisiti da input, utilizzando una funzione che calcola il minimo di due interi

```
int MinimoInteri(int i, int j)
{
  if (i<j)
    return i;
  else
    return j;
}
main()
{
  int a,b,c;
  printf("Primo intero="); scanf("%d",&a);
  printf("Secondo intero="); scanf("%d",&b);
  printf("Terzo intero="); scanf("%d",&c);
  printf("%d\n",MinimoInteri(a,MinimoInteri(b,c)));
}
```

Struttura di un programma C

- Un programma C è costituito da
  - Una parte dichiarativa globale opzionale
  - la definizione della funzione `main()`
  - un insieme di definizioni di funzioni, eventualmente vuoto
- `main()` è abbreviazione di `void main(void)`: è l'intestazione della funzione `main`
  - `main` può avere parametri di input per comunicare con il sistema operativo: infatti è possibile eseguire il programma con argomenti che vengono passati come parametri di `main`
- Non si può definire una funzione all'interno di un'altra