Dal Pascal al C

Parte 1ª

Introduzione

- Il linguaggio di programmazione Pascal fu sviluppato alla fine degli anni sessanta dal Prof. Niklaus Wirth

- Il linguaggio C fu ideato agli inizi degli anni settanta da Brian W. Kernighan e da Dennis M. Ritchie

Un ciclo while ed una istruzione di scrittura

Commento ai programmi **Somma_10.pas** e **Somma_10.cpp** (1)

- Le parentesi graffe aperte **{** (**Alt + 123** oppure **Shift + AltGr + [**) e chiuse **}** (**Alt + 125** oppure **Shift + AltGr +]**) del C corrispondono alle parole chiave **begin** e **end** del Pascal
- I commenti che nel Pascal sono racchiusi tra parentesi graffe o tra **(*** e ***)**, nel C sono racchiusi tra i simboli **/*** e ***/**
- Nelle istruzioni di assegnazione (ad es. **somma=0** e **somma=somma+1**) il simbolo "**=**" nel C sostituisce il simbolo "**:=**" del Pascal
- Nel Pascal dichiarazione delle variabili inizia per **var**, nel C non esiste corrispondente
- Nel Pascal l'identificatore di tipo segue la lista degli identificatori di variabile (**i, somma: integer**), nel C l'identificatore di tipo precede la lista degli identificatori di variabile (**int i, somma**)
- La sintassi del **while** è leggermente diversa nei due linguaggi
- L'intestazione del programma nel Pascal è composta dalla parola chiave **program** seguita da un identificatore (il nome del programma), nel C si usa la parola chiave **main** seguita da una coppia di tonde **()**, il C non ammette l'assegnazione di un nome al programma

Un ciclo while ed una istruzione di scrittura

Commento ai programmi **Somma_10.pas** e **Somma_10.cpp** (2)

- Il **write** del Pascal è sostituito dal **printf** del C
- **write('La somma vale: ', somma)** ha l'effetto di stampare la stringa racchiusa tra apici e poi il contenuto della variabile somma; nel C le stringhe sono racchiuse tra doppi apici, la stringa "La somma vale: %d" contiene anche i caratteri %d che specificano il formato dei dati che seguiranno (%d indica che sarà stampato il contenuto di una variabile intera (la variabile somma))
- Il comando del C **#include <stdio.h>** specifica che nel programma saranno utilizzate le funzioni standard di input-output (comando necessario ad esempio per utilizzare la funzione printf); l'estensione .h sta ad indicare che stdio.h è un file di intestazione (*header*); **#include** del C è analogo allo **uses** del Pascal
- A differenza del Pascal, nel C il programma non termina con un punto
- Nel C il punto e virgola è il terminatore di ogni singola istruzione e non il separatore tra istruzioni (è necessario un punto e virgola anche prima di una parentesi graffa chiusa)
- Nel C non si deve usare il punto e virgola dopo i commenti e dopo le dichiarazioni di inclusione precedute da **#include**
- Al contrario del Pascal, il C è un linguaggio **case-sensitive**

Un ciclo while ed una istruzione di scrittura

Commento ai programmi **Somma_10a.pas** e **Somma_10a.cpp** (3)

Per evitare che, in fase di esecuzione, la finestra dell'output si chiuda immediatamente e non lasci vedere i risultati, si può introdurre:

- nel Pascal l'istruzione **readln** o **repeat until keypressed** che però necessita della inclusione della unit Crt mediante **uses crt**
- nel C l'istruzione **system("PAUSE")** o **system("pause")**

In entrambi i linguaggi si determina una pausa di attesa che può essere interrotta premendo un tasto qualsiasi.

Nel C è necessario aggiungere il riferimento alla libreria **stlib.h** tramite il comando **#include <stlib.h>**.

Un ciclo while ed una istruzione di scrittura

Commento ai programmi **Somma_10b.pas** e **Somma_10b.cpp** (4)

- A differenza del programma **Somma_10a.pas**, il **Somma_10b.pas** introduce l'istruzione **writeln** al posto di **write**
- Il comportamento della **writeln** del Pascal è realizzato in C con la funzione printf, inserendo un '\n' (*new line*, il carattere speciale del C che fa saltare ad una nuova riga) alla fine della stringa che fa da primo parametro alla funzione.

Un ciclo for ed istruzioni di lettura e scrittura

Commento ai programmi **Som10num.pas** e **Som10num.cpp**

- L'istruzione **for** è presente sia nel Pascal che nel C, ma con una sintassi ed una semantica differenti. Nella istruzione **for(i=1; i<=10; i=i+1)** i è la variabile di controllo del ciclo; i=1 specifica il valore assegnato ad i all'inizio dell'esecuzione del ciclo; se i<=10 risulta vera il ciclo continua ad essere eseguito, altrimenti il ciclo termina; i=i+1 determina la modalità di variazione della variabile di controllo.
- Le istruzioni di lettura si presentano in Pascal ed in C in modo diverso, la funzione **scanf("%d",&num)**, analogamente alla printf, ha due argomenti separati da una virgola. Il primo argomento, una stringa, specifica con la stessa notazione della printf il tipo del dato da leggere, in questo caso un intero. Il secondo argomento, &num, specifica che il dato letto deve essere assegnato alla variabile num e per fare questo si fornisce alla scanf tramite &num l'indirizzo della locazione di memoria in cui memorizzare il dato letto.

Una istruzione if ed istruzioni di lettura e scrittura

Commento ai programmi **Max.pas** e **Max.cpp**

- L'istruzione **if** è presente sia nel Pascal che nel C, ma con una sintassi leggermente differente. In C l'espressione che condiziona l'esecuzione dell' **if** va messa obbligatoriamente tra parentesi tonde e manca la parola chiave **then**. Da notare il punto e virgola prima della parola chiave **else**. Tale punto e virgola è necessario in C in quanto è il terminatore dell'istruzione che precede l'**else**.
- Il comportamento della **writeln** del Pascal è realizzato in C con la funzione printf, inserendo un '\n' (*new line*, il carattere speciale del C che fa saltare ad una nuova riga) alla fine della stringa che fa da primo parametro alla funzione.

Numeri interi

In C si distinguono **numeri interi** e **numeri con separatore decimale**. Come nel Pascal, un intero è costituito da una sequenza di cifre; nel C vi sono però alcune possibilità in più:

- È possibile usare anche numeri in base 8 ed in base 16. Se la prima cifra è diversa da zero (o il numero è 0) un numero è considerato in base 10; se un numero inizia con lo 0 viene interpretato come ottale; se i primi due caratteri sono 0x il numero viene interpretato come esadecimale.
- È possibile utilizzare dei caratteri postfissi per specificare il tipo di un intero: “l” o “L” individuano un intero di tipo **long**, “u” o “U” individuano un intero di tipo **unsigned** (un tipo senza segno); i caratteri postfissi possono essere utilizzati anche contemporaneamente (non è importante l’ordine) per definire un intero **unsigned long**.

Esempi di numeri interi sintatticamente corretti:

15 -45 4196L 654u 077 0334L 0xFFFu 0x89AB

Numeri floating point

Un numero con separatore decimale mobile (*floating point*) può essere scritto, in Pascal come in C, sia in notazione standard che in notazione scientifica (mantissa ed esponente). Mentre, però, nel Pascal esiste solo il tipo **real**, nel C esistono i tipi primitivi **float** e **double**. Se nessun carattere postfisso è presente, il numero è considerato **double**; i caratteri “f” e”F” indicano che il numero è **float**.

Esempi di numeri interi sintatticamente corretti:

0E0 123456.345 3.28e3 456.23e-23 6E+2 201e1 .648e-12

Identificatori

- La sintassi degli identificatori C è la stessa degli identificatori Pascal. L'unica differenza tra i due linguaggi è che mentre nel Pascal solitamente non c'è distinzione tra maiuscole e minuscole, nel C gli identificatori aa ed AA sono due identificatori diversi: il C è un linguaggio **case sensitive**.
- Non c'è restrizione sul numero di caratteri che formano un identificatore.
- Analogamente al Pascal esistono varie parole chiave che **non** possono essere usate come identificatori:

argc	**argv**	**auto**	**break**	**case**	**char**	**const**
continue	**default**	**do**	**double**	**else**	**enum**	**envp**
extern	**float**	**for**	**goto**	**if**	**int**	**long**
main	**register**	**return**	**short**	**signed**	**sizeof**	**static**
struct	**switch**	**static**	**struct**	**typedef**	**union**	**unsigned**
void	**volatile**	**while**				

Caratteri speciali

Accanto ai caratteri alfanumerici tradizionali è possibile utilizzare in C vari caratteri speciali. Un carattere speciale è denotato da un \ seguito da uno o più caratteri:

\b backspace (una battuta indietro)
\f form feed (salto pagina)
\n new line (a capo riga nuova)
\r carriage return (ritorno a capo della stessa riga)
\t tab (tabulazione)
\’ single quote (apice singolo)
\” double quote (apice doppio)
\0 end of string (fine stringa)
\\ backslash (\) (barra contraria)
\a bell (segnalazione acustica)
\ddd (carattere ASCII associato all’ottale ddd)
\xdd (carattere ASCII associato all’esadecimale dd)

Ad esempio i caratteri speciali ‘\104’ e ‘\x44’ corrispondono al carattere D.

Stringhe

Una serie di caratteri (speciali e non) racchiusi tra doppi apici forma una stringa (es. “esempio di stringa”). In pratica l’apice del Pascal è sostituito nel C dal doppio apice. Per evitare ambiguità all’interno della stringa non può essere utilizzato il carattere doppio apice se non è preceduto da un \; per analoghi motivi il carattere \ deve essere rappresentato tramite il carattere speciale ‘\\’. I caratteri speciali possono essere utilizzati all’interno delle stringhe come qualsiasi altro carattere.

Commenti

All’interno del testo di un programma è possibile inserire commenti. I commenti sono delimitati nel C da /* e */. Ad esempio: /* questo è un commento */. Come nel Pascal i commenti non possono essere nidificati.

Struttura di un programma Pascal

La struttura di un programma Pascal si può così riassumere:

- l'intestazione del programma è seguita da un blocco, nel blocco compaiono nell'ordine:
  - dichiarazione **unit** (librerie) con **uses**
  - dichiarazione **etichette** con **label**
  - dichiarazione **costanti** con **const**
  - dichiarazione **tipi** con **type**
  - dichiarazione **variabili** con **var**
  - dichiarazione **procedure** con **procedure**
  - dichiarazione **funzioni** con **function**
  - **istruzioni**
- il programma si conclude con un punto
- le parti dichiarazioni sono fra loro ben distinte e devono rispettare l'ordine mostrato
- la parte istruzioni inizia con un **begin** e termina con un **end**; a loro volta le procedure e le funzioni dichiarate nel blocco contengono un blocco; ciò consente di scrivere programmi la cui nidificazione di procedure e funzioni è teoricamente illimitata

Struttura di un programma C

Nel C la situazione è profondamente diversa e la strutturazione del codice molto più limitata e molto meno rigida rispetto al Pascal:

- nel C non esistono procedure e l'unico strumento per scrivere sottoprogrammi è la funzione
- le componenti principali sono una intestazione e un blocco-C
- l'intestazione è "di solito" composta esclusivamente da **main**()
- il blocco-C è delimitato da una coppia di parentesi graffe e contiene sia dichiarazioni di tipi e variabili che istruzioni
- il blocco-C non può contenere dichiarazioni di funzioni
- prima di **main**() e dopo la fine del blocco-C possono essere usate altre dichiarazioni e/o definizioni (di tipi, variabili, funzioni)
- le funzioni C non possono essere annidate (sono tutte allo stesso livello, vengono dichiarate prima o dopo il blocco-C preceduto da **main**)
- nel C manca una parte dichiarazione costanti ed etichette
- un programma C si presenta come una sequenza di dichiarazioni e il programma principale è una funzione contraddistinta dal particolare nome **main**

Tipi di dati del linguaggio C

	Tipo	Intervallo	Bit utilizzati
caratteri	**unsigned char**	da 0 a 255	8 bit
	char	da -128 a 127	8 bit
interi	**unsigned short int**	da 0 a 65535	16 bit
	short int	da -327688 a 32767	16 bit
	int	da -2147483648 a 2147483647	32 bit
	unsigned long int	da 0 a 4294967295	32 bit
	long int	da -2147483648 a 2147483647	32 bit
reali	**float**	da 1.17549435×10^{-38} a $3.40282347\times10^{+38}$	32 bit
	double	da $2.2250738585072014\times10^{-308}$ a $1.7976931348623157\times10^{+308}$	64 bit
	long double	da 3.4×10^{-4932} a $1.1\times10^{+4932}$	80 bit

Tipi primitivi e tipi semplici

- Nel C come nel Pascal, vengono forniti dei tipi primitivi per rappresentare interi, numeri con separatore decimale e caratteri ed è possibile definire tipi enumerati.
- Nel Pascal agli interi corrisponde il solo tipo **integer**. Nel C gli interi sono rappresentati dal tipo **int** e dalle sue varianti ottenute con i prefissi **signed** / **unsigned** e **short** / **long**.
- Nel Pascal per rappresentare i numeri con il separatore decimale mobile è disponibile il solo tipo **real**. Nel C si possono utilizzare i tipi primitivi **float** e **double**; in alcuni compilatori si può utilizzare il tipo **long double**.
- I caratteri possono essere rappresentati nel C tramite il tipo primitivo **char**; anche i char possono essere **signed** / **unsigned**.
- Nel C a differenza di ciò che accade nel Pascal, non esiste il tipo primitivo **boolean**: nei programmi si fa corrispondere il valore 0 alla costante boolean false e tutti i valori diversi da 0 alla costante boolean true.
- Nel C esiste il tipo **void** che non ha analogie con il Pascal: il tipo void è un tipo il cui dominio dei valori è l'insieme vuoto; si attribuisce il tipo void ad oggetti (es. variabili o funzioni) il cui valore non è significativo.
- Mentre nel Pascal la definizione di un tipo enumerato non fa uso di alcuna parola chiave, nel C si usa la parola chiave **enum**.

Esempi di dichiarazioni di variabili

- **int numero;**
- **short int numeropiccolo;** oppure **short numeropiccolo;**
- **long int numeogrande;** oppure **long numeogrande;**
- **unsigned int numeropositivo;**
- **unsigned long int numeropositivogrande;**
- **char ch;**
- **unsigned char ch1;**
- **signed char ch2;**
- **double numeroreale;**
- **float numerorealepiccolo;**
- **long double numerorealegrande;**
- **char nome[30];** (stringa di 29 caratteri, il carattere **terminatore di stringa \0** viene aggiunto in automatico)
- **enum { rosso, blu, verde=5, giallo, bianco } a, b, c;**

  Il C tratta le variabili di tipo enumerato come variabili di tipo intero, creando una corrispondenza biunivoca tra gli identificatori elencati nella definizione del tipo e un insieme di interi; nell'esempio, rosso è associato all'intero 0 e blu a 1; al verde dovrebbe essere associato l'intero 2, ma il programmatore, attraverso l'assegnazione verde=5 altera il comportamento standard, forzando la corrispondenza tra verde e 5; di seguito giallo è associato a 6 e bianco a 7; in C scrivere **a=0** è come scrivere **a=rosso** e **a=5** è uguale a **a=verde**; il compilatore C non effettua alcun controllo se alla variabile **a** si assegna **3** o **10**, per cui ad es. l'istruzione **a=10** è lecita.