Corso introduttivo sui microcontrollori

A. S. 2007 – 2008

Richiami fondamentali sul linguaggio C

Il CCS Pic-C Compiler

Prerequisiti fondamentali => Conoscenze generali di programmazione in linguaggio C

Nicola Amoroso

namoroso@mrscuole.net

http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler

I Microcontrollori, di solito, sono programmati utilizzando il linguaggio assembler, questo è un linguaggio costituito da diverse istruzioni mnemoniche ed è specifico per ogni microcontrollore; **l'assembler per un tipo di microcontrollore non può essere utilizzato per un altro microcontrollore**.

In genere i programmi assembler producono codice abbastanza ottimizzato, nel senso che occupa meno risorse rispetto al codice generato con un altro linguaggio e spesso i tempi di esecuzione sono inferiori rispetto ai tempi di esecuzione di un codice generato con altro linguaggio; si dice che il linguaggio **assembler** è un **linguaggio a basso livello** in quanto il codice prodotto, dopo la compilazione, è ottimizzato per ottenere le migliori prestazioni con i tempi minori di esecuzione.

Presenta però diversi svantaggi; è un linguaggio come suol dirsi ostico nel senso che le istruzioni non sono tutte immediatamente comprensibili e di facile utilizzo e questo rallenta notevolmente lo sviluppo di progetti e lavori specialmente di una certa entità; abbastanza difficoltoso è anche la fase di verifica cioè come suol dirsi di "Debug". Il linguaggio assembler è molto utilizzato da quei programmatori con una certa esperienza e nei casi in cui si richiede un utilizzo "ottimizzato" delle risorse del sistema.

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler

I microcontrollori possono essere programmati utilizzando un **linguaggio ad alto livello**, ad esempio è possibile utilizzare compilatori in **BASIC**, in **PASCAL**, in **JAVA** o in linguaggio **C**, la maggior parte di questi compilatori generano codice macchina nativo che può direttamente essere caricato nella memoria del microcontrollore mediante opportuno “programmatore”.

In genere questi linguaggi hanno delle istruzioni molto più vicine al nostro normale linguaggio e quindi sono più facilmente “ricordabili” (da questo vengono anche definiti linguaggio ad alto livello); però il codice prodotto non è ottimizzato per utilizzare con le migliori prestazioni il nostro sistema, hanno però la caratteristica di permettere la “portabilità” del codice tra varie famiglie di microcontrollori nel senso che un codice (ad alto livello), generato per una certa famiglia di microcontrollori Microchip, può facilmente essere riadattato ad una famiglia diversa della stessa casa produttrice e addiritura si può avere il riutilizzo, con semplici adattamenti, anche per microcontrollori prodotti da case diverse; in questo modo ci si rende conto che lo sviluppo di progetti di una certa dimensione richiede un impiego di risorse ben inferiori rispetto all’uso del linguaggio assembler.

In questa ottica il linguaggio C è uno dei più utilizzati nella programmazione dei microcontrollori.

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler

In questa lezione analizzeremo i principi fondamentali per programmare microcontrollori PIC in linguaggio C.

C è uno dei più diffusi linguaggi di programmazione ad **alto livello** per microcontrollori, in particolare esistono diversi compilatori prodotti da case diverse, per i microcontrollori Microchip PIC ad 8, 16 ed ora anche 32 bit.

Analizzeremo brevemente le risorse di uno dei compilatori, in linguaggio C, più utilizzati per la programmazione dei microcontrollori MicroChip PIC, il **Pic-C Compiler** prodotto dalla Americana **C**ustom **C**omputer **S**ervice Inc. [http://www.ccsinfo.com].

Il CCS PCWH Pic-C compiler è un compilatore efficiente che può essere utilizzato per programmare tutta la serie 12xxx, 16xxx e 18xxx di microcontrollori MicroChip PIC, e anche possibile estendere questo compilatore con l'opzione di sviluppo per microcontrollori MicroChip Pic a 16 bit (compilatore PCWHD).
Il compilatore è equipaggiato con un editor integrato che rende lo sviluppo di codice relativamente facile.
Il sistema integrato permette anche la programmazione/debug on line mediante opportuno programmatore/debugger CCS ICD-U (versione USB) CCS ICD-S (versione seriale) collegato ad una opportuna porta del PC e collegato mediante standard ICD2 con la board di sviluppo (es. AnxaPic v4.0 revA).

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler

Il sistema di sviluppo include un buon numero di funzioni di libreria (built-in functions) che agevolano notevolmente la scrittura e lo sviluppo di codice sorgente; questa caratteristica “embedded” è un po’ comune a questo tipo di sistemi di sviluppo per microcontrollori, la possibilità di disporre di funzioni “preconfezionate” che rendono più semplice e più veloce lo sviluppo di un progetto.

Caratteristiche principali del CCS PCWH (PCWHD) Pic-C compiler:

- Supporto della aritmetica in virgola mobile (floating point)
- Disponibilità di driver per i più comuni display LCD text e grafici
- Disponibilità di un largo numero di funzioni matematiche
- Costo non eccessivamente elevato
- Disponibilità di un Forum di supporto facilmente accessibile e molto valido per i contenuti

Lo sviluppo di un programma è relativamente facile. I programmi utente sono normalmente sviluppati su un PC utilizzando l'editor integrato. Dopo la compilazione Il codice ottenuto può essere impiegato per la simulazione mediante un simulatore visual tipo Proteus VSM (*http://www.Labcenter.co.uk*). Una delle caratteristiche principali di questo compilatore è quello di produrre codice ad alto livello (*.cof) che permette la simulazione con opportuni software di tipo visual.
Dopo la simulazione mediante opportuno programmatore (CCS ICD-U, Microchip ICD2, Microchip pickit2, etc…) il codice può essere “caricato” nella flash memory program del controllore ed eseguito correttamente.

Programmare i microcontrollori in linguaggio C

La struttura di un programma in C

Tutti i programmi in C possono contenere:
direttive del preprocessore, identificatori, definizioni, espressioni, istruzioni e funzioni

Direttive del preprocessore
Una direttiva del preprocessore è un comando al preprocessore C (il quale è Invocato automaticamente come il primo passo nel compilare un programma). Le direttive di preprocessore più comuni sono la direttiva **#define**, che permette di sostituire testo al posto di un identificativo specificato, e la direttiva **#include**, che include del testo di un archivio esterno in un programma.

Identificatori
Un identificatore stabilisce i nomi delle variabili, delle funzioni e le etichette utilizzate nel programma. Le variabili globali sono dichiarate esterne alle singole funzioni e sono visibile dalla fine della dichiarazione alla fine del programma (visibilità globale). Una variabile locale viene dichiarato all'interno di una funzione e la visibilità è dalla fine della dichiarazione alla fine della funzione.

Definizioni (Tipi di dati)
Una definizione stabilisce il tipo di dato per una variabile o una funzione. Una definizione assegna (riserva) anche le locazioni di memoria necessarie per variabili e funzioni.

Espressioni
Un'espressione è una combinazione di operatori e operandi che produce un singolo valore.

Istruzioni
Le istruzioni controllano il flusso o l'ordine di esecuzione in un programma C.

Programmare i microcontrollori in linguaggio C

La struttura di un programma in C

Funzioni

Una funzione è una raccolta di dichiarazioni, definizioni, espressioni e istruzioni che eseguono un compito specifico. Le parentesi graffe includono il corpo di una funzione. Le funzioni possono essere non annidate in C.
Tutti i programmi C devono contenere una funzione chiamata "*main()*". La funzione "*main()*" è una funzione con la quale inizia il programma, le parentesi graffe che includono il corpo della "*main function*", rappresentano il punto iniziale e finale del nostro programma in C.

Esempio: Struttura generica di un programma in C

```
#include <stdio.h>          /* direttiva del preprocessore */
#define PI    3.142         // define un valore costante

float area;                 /* Variabile globale */
int square (int r);         /* Dechiarazione prototipo funzione */

int square(int r) {         /* Nome e tipo funzione*/
  int r_squared;            // Variabile locale

  r_squared = r * r;        //Calcolo quadrato del valore

  return(r_squared);        // return valore calcolato alla
                            // funzione chiamante
}
```

```
main()   {                  /* Inizio main function e programma */
   int radius_squared;      // Variabili locali
   int radius = 3;          // dichiarazione e init  variabile

/* Calcola il quadrato di "radius" - Pass valore  alla f unzione */
   radius_squared = square (radius);

/* Calcola il valore dell'area del cerchio */
   area = PI * radius_squared;
                            /* Invia valore*/
   printf("Area is %6.4f square units\n",area);
}                           /* Fine main function e programma*/
```

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

1. I Commenti

I commenti sono utilizzati per documentare il significato e l'operazione del codice sorgente. Tutti i commenti saranno ignorati dal compilatore. I commenti possono essere utilizzati in qualsiasi luogo nei programmi.

Ci sono due modi di includere commenti nei programmi:

a) utilizzando un carattere di doppia barra inclinata "// " seguito dall'opportuno commento :

// Questo è un commento ← Il compilatore non prenderà in considerazione tutto il testo che segue il // fino a fine riga

Char x; // **x è un char** ← Commento non considerato dal compilatore

b) Includendo il commento all'interno degli identificativi "/* "e "*/" ← Il compilatore non prenderà in considerazione il testo incluso tra questi due identificativi.

/* Questo è esempio di commento multilinea. Il commento serve per spiegare il programma e le varie operazioni che avvengono al suo interno. Questo commento termina su questa linea */

```
main()  {                          /* Inizio main function e programma */
   int radius_squared;             // Variabili locali
   int radius = 3;                 // dichiarazione e init  variabile

/* Calcola il quadrato di "radius" - Pass valore  alla f unzione */
   radius_squared = square (radius);

/* Calcola il valore dell'area del cerchio */
   area = PI * radius_squared;
                                   /* Invia valore*/
   printf("Area is %6.4f square units\n",area);
}                                  /* Fine main function e programma*/
```

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

2. Il Preprocessore

Il preprocessore è quella parte del compilatore C che agisce prima della fase di compilazione. I comandi (direttive) del preprocessore iniziano con il carattere "#".

Analizziamo alcuni dei comandi del preprocessore più utilizzati:

→ **Macro**

La parola chiave "**#define**" è utilizzata per definire una macro in cui un simbolo è sostituito con un altro simbolo o costante.

Ad esempio le seguenti dichiarazioni equiparano Ritardo al valore 100 e Orologio al valore 12:

```
#define  Ritardo  100
#define  Orologio 12
```

Quando questi simboli sono utilizzati in un programma, essi saranno sostituiti con i rispettivi valori.

Un esempio: X = Ritardo + 10 ; **//Alla variabile X è assegnato il valore 110**

```
#define  MAX(A,B)   (A > B) ? A : B
z = MAX(x,y);          //z conterrà il valore maggiore tra x e y
```

http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

2. Il Preprocessore

→ **Include**

Questa direttiva del preprocessore è utilizzata per includere un file per la compilazione. Nel seguente esempio;

#include < 16F877.h > //Ricerca il file da includere nella directory di installazione del compilatore

il contenuto del file "16F877.h" è incluso all'interno del file da compilare:

Allo stesso modo, il comando:

#include "16F877.h " //Ricerca il file da includere nella directory di progetto dell'utente

include in contenuto del file "16F877.h" all'interno del file da compilare.

N.B:➔ Quando il nome del file da "includere" è scritto tra i simboli < e > (<16F877.h>) allora il file viene cercato all'interno della directory di installazione del CCS Pic-C compiler (Es: **C:\programmi\Elettronica\ccs\picC\Devices**)

Quando il nome del file da "includere" è scritto tra i simboli " e " ("16F877.h") allora il file viene cercato all'interno della directory di progetto dell'utente e se il file non è presente viene ricercato nella directory di installazione del CCS Pic-C compiler

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

2. Il Preprocessore

→ **Compilazione condizionata**

Il C presenta alcune interessanti direttive del preprocessore che permettono di "Includere" o "Escludere" delle sezioni di codice da compilare in funzione di valori assegnati a certe variabili. Ad esempio:

```
#define   HW_Version   3              //Quando incontri HW_Version sostituiscilo con il valore 3
  #if  HW_Version > 2                 //HW_Version maggiore di 2?
            output_high(PIN_B1);      //Set PIN RB1 al valore H
  #else                               //Altrimenti
            output_high(PIN_B2);      //Set PIN RB2 al livello H
  #endif                              //Fine compilazione condizionata
```

Un altro esempio:

```
#define   Debug                       //Definisci Debug to TRUE
  #ifdef  Debug                       //Debug definito?
            printf("Ciao belli …");   //Invia il messaggio via RS232
  #endif                              //Fine compilazione condizionata
```

http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

2. Il Preprocessore

→ **Compatibilità hardware**

Il compilatore deve conoscere le definizione "hardware" e di "configurazione" del controllore usato in modo da produrre codice corretto, alcune direttive del preprocessore ci vengono in aiuto. Esempio:

```
#include  <18F452.h>                    //Device per specifiche HW come ad esempio i nomi dei PIN
#fuses    hs, nowdt                     //Config di funzionamento => oscillatore > 4 MHz, no watch dog timer
#use  delay(clock=20000000)             //Clock oscillatore 20 MHz => Utile per i delay
#use   rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)      //Uso della rs232 con specifiche
#use   i2c(master, scl=PIN_C4, sda=PIN_C4)            //Uso bus i2c con specifiche
```

Alcune direttive del preprocessore permettono di "creare" e "mappare" →variabili← per i riferimenti hardware di alcuni registri e flags. Esempio:

```
#bit   carry=3.0      //La variabile carry di tipo bit (int1) è il bit 0 del registro di stato (indirizzo 3)
#byte  portb=0x06     //La variabile portb di tipo byte (int8) è definito come il registro speciale all'indirizzo di memoria 6
#byte  intcon=11      //Analogamente intcon è il registro all'indirizzo 11 della mappatura di memoria
```

http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

2. Il Preprocessore

→ **#pragma**

La direttiva (comando) pragma istruisce il compilatore ad eseguire una certa azione in fase di compilazione; ad esempio specificando il tipo di Picmicro MCU device con questa direttiva:

```
#pragma  device  PIC16F887          //Formato file prodotto compatibile con questo device
```

Si dice al compilatore di generare un file che segua le direttive del device specificato

Ancora:

```
#pragma  device  ICD=TRUE          //Attiva la conidzione Debug HW nel  file generato
#pragma  device  ADC=8             //Impiego ADC con risoluzione ad 8 bit
```

Per il CCS Pic-C compiler il pragma è facoltativo cosicché verranno accettate anche le direttive:

```
#device   PIC16F887                //Formato file prodotto compatibile con questo device
#device  ICD=TRUE                  //Attiva la conidzione Debug HW nel  file generato
#device  ADC=8                     //Impiego ADC con risoluzione ad 8 bit
```

http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

3. Le costanti

Una costante è un valore fisso che non può essere cambiato dal programma. Ad esempio, 25 è una costante.

Sono costanti i valori: 11, 893, -25, 40, -100, 32.33, -45.3, 'A', '&', etc … . ← Si noti che le costanti di tipo char sono racchiuse tra apici.

Quando il compilatore CCS Pic-C incontra una costante nel nostro programma, deve decidere quale tipo di costante è; Il compilatore C assegnerà per default, alla costante Il tipo di dati compatibile più piccolo che riterrà opportuno, così 15 è un int8 e 64000 è un unsigned int16 mentre 15.23 è un float etc … .

Una costante può essere dichiarata utilizzando la direttiva del preprocesssore #define.

Ad esempio:

```
#define   TRUE 1            //A TRUE viene assegnato il valore costante 1
#define   pigreco  3.14159265359      //Per pigreco viene assegnato il noto valore
```

Il Pic-C permette di specificare le costanti in formati esadecimali e ottali. Le costanti esadecimali devono essere preceduti dal prefisso '0x.' Ad esempio 0xA4 è una costante esadecimale valida.

```
#define   i2c_addr_ds1307   0xD0      //Un indirizzo i2c valido per ds1307 RTC
```

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

3. Le costanti

I valori costanti definiti con **#define** non sono memorizzati nella ROM del controllore, l'utilizzo del valore, in fase di compilazione, è contestuale durante la compilazione stessa.

Per memorizzare i valori delle costanti nella ROM del controllore, bisogna utilizzare la parola chiave **const** in fase di dichiarazione. Ad esempio:

char **const** [ ] = {"Nikolino"} **//Dichiara una stringa di carattere che occupa 9 locazioni di memoria**

Utilizza nove locazioni di memoria per immagazzinare la stringa perché per la stringa bisogna considerare il carattere NULL come terminatore [8 locazioni per "Nikolino" + 1 locazione come terminatore di stringa (carattere ascii NULL ← /0)].

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

4. Le variabili

Tutte le variabili usate in un programma devono essere dichiarate prima del loro utilizzo, possono essere definite all'interno di una funzione (variabili locali, visibilità locale) e il loro valore viene mantenuto solo all'interno della funzione dove sono state dichiarate; le variabili globali vengono dichiarate all'esterno di tutte le funzioni, la loro visibilità e il loro valore permane all'interno di tutto il programma e possono essere utilizzate in qualsiasi punto del programma; naturalmente le variabili globali occupano risorme (memoria, registri) per tutta la durata del programma mentre le variabili locali occupano risorse solo durante la esecuzione della funzioni cui appartengono.

Il CCS Pic-C compiler supporta le seguenti variabili:

Tipo	Dimensione	Unsigned	Signed	C Standard type
Int1 - bit	1 bit	0 to 1	nd	Short
Int8 –byte - char	8 bit	0 to 255	-128 to 127	Char - Int
Int16 – long int	16 bit	0 to 65535	-32768 to 32767	Int - Long
Int32	32 bit	0 to 4294967295	-2147483648 to 2147483647	Long Long
Float	32 bit	-1.5×10^{45} to 3.4×10^{38}		Float

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

5. Array e stringhe

Un array è semplicemente una lista di variabili dello stesso tipo di dati

In CCS C compiler un array viene dichiarato come dichiarato in C Standard. Per esempio:

```
int8  Nikolino[10];          // Dichiara un array chiamato Nikolino contenente 10 variabili di tipo int8
```

Il primo elemento dell'array è Nikolino[0] mentre l'ultimo elemento e Nikolino[9].

Il valore tra parentesi quadre è l'**indice** dell'array ed è compreso, nel nostro esempio, tra 0 e 9. Esempio:

```
a = Nikolino[5]              // a dichiarato in precedenza di tipo int8
```

Alla variabile "a" viene assegnato il **sesto valore** dell'**array Nikolino**.

Un modo per inizializzare un array:

```
int8 i[5] = {1, 22, 3, 34, 15}          //Array inizializzato con i valori 1, 22, 3, 34, 15
```

L'elemento i[0] = 1, …, l'elemento i[2] = 3, …, l'elemento i[4] = 15

http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

5. Array e stringhe

Una stringa in C è un array di char (Caratteri) che è 'terminato' con il carattere Null (/0 – codice ascii 0)

Le stringhe possono essere inizializzate includendo i componenti dell'array tra virgolette. Nel seguente esempio, la stringa nome è inizializzato a "Pippo" :

char nome[ ] = "Pippo" ; **// La stringa 'nome' è inizializzato a "Pippo"**

L'array 'nome' contiene 6 componenti di tipo char infatti si hanno 5 char ("Pippo") + carattere Null (/0) come terminatore di stringa**.**

Un altro modo per inizializzare un array di char:

char str[] ('a', 'p', 'C', 'Q') **//Stringa inizializzata con 4 caratteri**

Il compilatore può calcolare la dimensione dell'array dal numero di componenti racchiuso tra virgolette.

Nella inizializzazione di stringhe non vi sono le parentesi graffe questo perché le stringhe in C terminano con il carattere Null (/0 – codice ascii 0); il compilatore automaticamente pone il carattere null come terminatore di stringa!

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

5. Array e stringhe

Gli array possono avere più di una dimensione come mostrato nel seguente esempio:

```
char temp [3] [2]          //Viene dichiarato un array bidimensionale con 3 righe e 2 colonne
```

Qui, temp è un array di 3 righe e 2 colonne.

Il primo elemento dell'array è temp[0][0], il secondo elemento è temp[0] [1], il terzo elemento è temp[1][0] il quarto elemento è temp[1] [1], il quinto elemento è temp[2][0] e l'ultimo elemento è temp[2] [1].

Un array bidimensionale può essere inizializzato specificando gli elementi righe e colonne tra parentesi graffe.

In questo esempio viene inizializzato l'array 'Key' (4 righe, 4 colonne):

```
char Key [4] [4] = { '1', '2', '3', 'A',          //Inizializzazione di un array bidimensionale
                     '4', '5', '6', 'B',
                     '7', '8', '9', 'C',
                     '*', '0', '#', 'D' };
```

Un array di char con 10 nomi ognuno lungo 40 caratteri (compreso il carattere null):

```
char  nome [10] [40];          //Array di 10 nomi ognuno di 39 caratteri
```

Per accedere al singolo nome basta indicare solo l'indice di riga => es: nome[3]

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

6. Operatori

Il CCS Pic-C compiler supporta tutti gli operatori definiti in standard C

( ) [ ]	Parentesi	**sizeof**	Dimensione variabile
→ .	Operatori membro di struttura	**== !=**	uguale diverso
!	NOT logico	**||**	OR logico - Relazionale
~	Complemento a uno	**&&**	AND logico - Relazionale
+ - * /	Operatori aritmetici	**|**	OR logico (tra bit)
%	Modulo - Resto della divisione	**&**	AND logico (tra bit)
++	Incremento	**^**	OR Esclusivo (XOR tra bit)
--	Decremento	**?:**	Operatore ternario
&	Operatore su puntatore	**+= -= /= *=**	Operatori abbreviati
*****	Operatore su puntatore	**|= %= ~=**	Operatori abbreviati
<< >>	Operatori di scorrimento	**&= <<=**	Operatori abbreviati
>= >	Operatori logici - Relazionale	**>>=**	Operatori abbreviati
<= <	Operatori logici - Relazionale		

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

7. Le funzioni

Una funzione è un blocco indipendente di codice che può essere chiamato dal programma principale.

Le funzioni sono utilizzate di solito quando è richiesto di ripetere la stessa operazione in diverse parti del codice principale. Una funzione **può** o **non può** ritornare un valore alla funzione chiamante (dipende dalla funzione chiamante). **Tutte le funzioni devono essere dichiarate prima del loro utilizzo!**

Nel seguente esempio abbiamo 2 dichiarazioni di funzioni; la funzione Resto ritorna un int8 alla funzione chiamante mentre la funzione Led non ritorna alcun valore:

```
int8 Resto();        void Led();
```

Il formato generale di una funzione è:

specificatore_di_tipo nome_funzione(argomenti_funzione)

dichiarazione argomenti // specifica il tipo degli argomenti che compaiono nella lista argomenti_funzione

{ corpo della funzione

}

specificatore_di_tipo => Dichiara il tipo di appartenenza del valore che la funzione restituisce attraverso return

argomenti_funzione => E' una lista di nomi di variabili, separati da virgola, che ricevono i nomi (valori) degli argomenti quando la funzione viene chiamata. Spesso argomenti_funzione e dichiarazione argomenti coincidono con una unica dichiarazione. Gli argomenti_funzione possono anche non esserci

http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

7. Le funzioni

Le funzioni che accettano argomenti devono predisporre un insieme di variabili che hanno lo scopo di ricevere i valori. Tali variabili sono detti parametri formali della funzione e si comportano come qualsiasi altra variabile locale. La dichiarazione dei parametri formali è posta tra il nome della funzione e il corpo della funzione. Un esempio:

```
Int16 quad(x) //La funzione quad è di tipo int16 (ritorna un valore a 16 bit
  int8 x      //dichiarazione del parametro formale
{             //Inizio corpo della funzione
  x= x*x;     //Calcolo quadrato – Il valore di x viene modificato
  return(x);  //Ritorna il valore calcolato
}            //Fine corpo della funzione
/*  La  funzione  main  non  ritorna  nessun  valore  (tipo  Void)  e  non  presenta  alcun
argomento_funzione (void per argomenti_funzione) */
Void main(void) {    //Dichiarazione funzione main
  int8 t=10;         //Variabile locale 8 bit
  int16 y;           //Variabile locale 16 bit
  y = quad(t)  //A y viene assegnato il valore che ritorna la funzione quad
}
/*Si noti che alla funzione quad viene passato il parametro attuale t=10 e una copia
del valore passato viene assegnato al parametro formale x in quad */
```


http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

7. Le funzioni

Il passaggio di valori dai parametri attuali ai parametri formali delle funzioni può avvenire come passaggio per valore. Il valore del parametro attuale viene copiato nella variabile che costituisce il parametro formale**.** Modifiche eseguite sul parametro formale non hanno alcun effetto sulla variabile usata all'atto della chiamata**.**

Il CCS Pic-C compiler accetta la seguente sintassi per il passaggio di valori tra funzioni:

```
// Il tipo di parametro formale viene dichiarato insieme al parametro
Int16 quad(int8 x)    //La funzione quad è di tipo int16 (ritorna un valore a 16 bit)
{             //Inizio corpo della funzione
  x= x*x;     //Calcolo quadrato – Il parametro formale viene modificato
  return(x);  //Ritorna il valore calcolato
}            //Fine corpo della funzione
/*  La  funzione  main  non  ritorna  nessun  valore  (tipo  Void)  e  non  presenta  alcun
argomento_funzione (void per argomenti_funzione) */
Void main(void) { //Dichiarazione funzione main
  int8 t=10;            //Variabile locale 8 bit
  int16 y;              //Variabile locale 16 bit
  y = quad(t)  //A y viene assegnato il valore che ritorna la funzione quad
}
/*Si noti che alla funzione quad viene passato il parametro attuale t=10 e una copia
del valore passato viene assegnato al parametro formale x in quad */
```


http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

7. Le funzioni

Abbiamo accennato in precedenza: **Tutte le funzioni devono essere dichiarate prima del loro utilizzo!**

Negli esempi precedenti tutte le funzioni con i rispettivi corpi sono state implementate prima del loro utilizzo da parte della funzione chiamante, infatti "int16 quad(int8 x)" è stata implementata prima della funzione "Void main(void)". Vi è un'altra possibilita per dichiarare una funzione prima del suo utilizzo anche se la funzione viene implementata dopo la funzione chiamante, la dichiarazione del prototipo di funzione prima della funzione chiamante. Vediamo un esempio con sintassi valida per CCS Pic-C compiler:

```
Int16 volume(int8 s1, int8 s2, int8 s3);      //Dichiarazione prototipo funzione prima della funzione chiamante
Void main(void)  {            //Funzione main di tipo Void senza parametri attuali (void)
   int16 vol;                 //Variabile locale
   vol = volume(5, 7, 16);    //A vol viene assegnato il valore che ritorna la fiunzione – Vengono passati 3 parametri attuali
}
/* Funzione che calcola il volume e ritorna il rispettivo valore – Riceve 3 parametri formali di tipo int8 – La funzione viene implementata dopo la funzione chiamante, questo è possibile perché il prototipo di funzione è stato dichiarato prima dell'utilizzo di questa funzione in main */
Int16 volume(int8 s1, int8 s2, int8 s3)  {
   return(s1 * s2 * s3);
}
```

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

8. Strutture di controllo

Il CCS Pic-C compiler dispone delle seguenti strutture di controllo:

- **if - else**
- **for**
- **while**
- **do -while**
- **cicli annidati**
- **continue**
- **break**
- **switch - case**
- **loop infiniti**
- **loop privi di corpo**
- **return**

I costrutti di controllo sono elementi importanti in qualsiasi linguaggio e lo è anche per il C; li analizzeremo con maggior dettaglio.

http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

8. Strutture di controllo

→If – Else

Queste istruzioni sono utilizzate per operazioni condizionali. Il formato generale è:

```
if(condizione)          //Se condizione è > vera
    istruzione1;        //esegui questa struzione1
else                    //Altrimenti
    istruzione2;        //esegui questa struzione2
```

```
if(x >= 5)
    a = b-12;
else
    a--;
```

Oppure

```
if(condizione)  {       //Se condizione è vera
    istruzione;         //esegui istruzioni tra
    ……..                //le parentesi graffe
}
else    {               //Altrimenti
    istruzioni;         //Esegui istruzioni
    ……..                //tra queste
}                       //parentesi graffe
```

```
if(x==0) {
    b=b+1;
    a=b-2;
}
else {
    b=3;
    a++;
}
```

La condizione else può essere facoltativa

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

8. Strutture di controllo

→**for**

Il costrutto for permette di eseguire cicli iterativi (loop) sino a quando non si verifica un certa condizione.

Il formato generale è:

```
for(init; condizione; incremento)     //init  Valore iniziale della variabile di controllo
    istruzione;          //condizione  valutazione relazionale per determinare  uscita loop
                         //incremento    incremento variabile di controllo
```

Oppure

```
for(init; condizione; incremento) {
istruzione;              //Istruzioni tra
    ……..                 //le parentesi graffe

}
```

```
for(i=0; i<11; i++) {
   for(j=0; j<(a+3); j=j+2) {
      k++;
      m=k;
    }
}
```

Nell'esempio riportato abbiamo un ciclo annidato il primo ciclo è controllato dalla variabile i mentre il secondo ciclo viene controllato con la variabile j.

Per i che và da zero e fino a quando i<11, con incremento di i di 1 per ogni loop, esegui il loop controllato dalla variabile j (per ogni loop j si incrementa di 2).

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

8. Strutture di controllo

→**while**

Il costrutto while permette di eseguire cicli iterativi (loop) sino a quando non si verifica un certa condizione.

Il formato generale è:

```
while(condizione)     //condizione  valutazione relazionale per determinare  uscita loop
    istruzione;
```

Oppure

```
while(condizione) {   //Sino a quando condizione è vera
    istruzione;       //esegui le istruzioni tra
    ……….              //le parentesi graffe

}
```

```
i=0;
while(i<10) {
   i++;
   printf("%d ",i);
}
```

Nell'esempio riportato abbiamo un ciclo ripetuto per 10 volte; per ogni ciclo i viene incrementato e viene inviato via rs232 il valore di i. Il ciclo termina quando i=10.

```
While(1)              Questa è una condizione di ciclo infinito in quanto la condizione
    printf("ciao")    1 > 0 è sempre verificata!!!!
```

http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

8. Strutture di controllo

→**do-while Loop**

Il costrutto do-while è simile al costrutto while ma a differenza del while nel do-while la verifica viene effettuata alla fine del ciclo, questo fa sì che il loop viene eseguito almeno una sola volta.

Il formato generale è:

```
do  {                    //Esegui
    istruzione;          //queste istruzioni tra
    ……..                 //le parentesi graffe
}while(condizione);      //sino a quando condizione è vera
```

Si noti che la prima verifica di condizione avviene dopo che il ciclo è stato eseguito per una volta

```
#include  <16F877.h>
#use RS232 (Baud=9600, xmit=PIN_C6, RCV=PIN_C7)
Void main(void) {        //Inizio programma
    do  {                //Ripeti ciclo
       ch=getch();
    }while(ch != 'q');   //Sino a che ch != 'q'
    printf("OK! Tutto bene\n");
}                        //Fine programma
```

Il messaggio viene inviato via rs232 dopo che viene letto 'q'

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

8. Strutture di controllo

→Cicli annidati

Quando il corpo di un ciclo contiene un altro ciclo, si dice che il secondo ciclo è annidato dentro il primo ciclo. Lo standard ANSI-C specifica che un compilatore deve permettere almeno 15 cicli annidati.

Un esempio di ciclo annidato:

```
i = 0;
While(i<10)   {                          //Sino a quando i<10 ripeti il seguente ciclo
      for(j=0; j<10; j++)                //Ciclo annidato di scrittura di 10 numeri
           printf("%d  ", i * 10+j);
      i++;
}                                        //Fine del ciclo esterno
```

L'esempio riportato invierà via rs232 tutti i valori compresi tra 00 e 99

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

8. Strutture di controllo

→ Continue

Continue è una istruzione che permette di uscire da un loop (ciclo). Se il ciclo corrente è un ciclo while, dopo l'istruzione continue viene saltato tutto il codice che segue sino alla fine del ciclo e il programma torna a verificare la condizione che permette di uscire dal ciclo. Se invece siamo dentro un ciclo for dopo l'istruzione continue il programma salta alla fine del ciclo e quindi torna a verificare la nuova condizione di abbandono del ciclo stesso.

Un esempio:

```
#include  <16F887.h>                    //Definizioni per Pic16F887
Void main(void)  {                      //Inizio programma
  int i;                                //Variabile locale
  for(i=0; i<100; i=i+2)  {             //Esegui questo ciclo per 50 volte
    continue;                           //Salta a fine ciclo
    printf("%d ", i);                   //Non sarà mai eseguita!!!!
  }                                     //Fine ciclo – Incrementa i di 2 e verifica
}                                       //Fine programma
```

L'istruzione printf non verrà mai eseguita! Con continue si salterà sempre a fine ciclo

http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

8. Strutture di controllo

→ Break

Questa istruzione provocherà un abbandono del ciclo in esecuzione e un salto alla istruzione immediatamente dopo la fine del ciclo..

Un esempio:

```
Void main(void)  {                    //Inizio programma
  int i;                              //Variabile locale
  for(i=0; i<50; i++)  {              //Esegui questo ciclo per 50 volte
    printf("%d ", i);                 //invia valore di i via rs232
    if(i==15)                         //Se i è uguale a 15
       break;                         //Abbandona il ciclo e invia "Bravo" via rs232
  }                                   //Fine ciclo – Incrementa i di 2 e verifica
  printf("Bravo");
}                                     //Fine programma
```

Questo esempio invierà via rs232 i valori da 0 a 15 e poi invierà "Bravo".

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

8. Strutture di controllo

→ Switch - Case

Switch-Case è una istruzione di test multiplo, il valore di una variabile viene confrontato con una lista di costanti che possono essere di tipo intero o caratteri, quando si verifica una eguaglianza si esegue il blocco di istruzioni corrispondente; se nessuna eguaglianza si verifica viene eseguito il blocco di istruzioni di `default` (se presenti).

Il seguente esempio legge un numero tra 0 e 6. Se il numero letto non è compreso nell'intervallo previsto si invierà "valore errato" altrimenti si invierà il nome del giorno della settimana corrispondente al valore letto; con il carattere 'x' termina il ciclo di lettura:

```
Void main(void)  {      //Inizio programma
  char ch;              //Variabile locale
  for(;;)  {            //Ciclo infinito
    ch=getch();         //Leggi valore
    if(ch==“x”)         //Letto ‘x’ ?
       return 0; //Esci ciclo for-Return valore 0
    switch(ch)  {
       case ‘0’: printf(“Dom\n”);  //Invia Dom
                 break;         //Esci ciclo for
       case ‘1’: printf(“Lun\n”); //Invia Lun
                 break;        //Esci ciclo for
       case ‘2’: printf(“Mar\n”);
                 break;
       case ‘3’: printf(“Mer\n”);
                 break;
       case ‘4’: printf(“Gio\n”);
                 break;
       case ‘5’: printf(“Venm\n”);
                 break;
       case ‘6’: printf(“Sab\n”);
                 break;
       default: printf(“Valore errato\n”);
     }    //Fine Switch-Case
   }      //Chiusura ciclo infinito for
}         //Fine programma
 Per terminare il ciclo premere la lettera ‘x’
```

http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

8. Strutture di controllo

→ Loop senza fine

Le istruzioni for e while possono facilmente essere impiegate per la ripetizione di un loop infinite volte; è sufficiente lasciare vuota la parte relativa alla condizione di uscita affinchè questa sia sempre verificata come valore logico vero:

```
For(;;)      //Nei loop senza fine vengono tralasciati anche la inizializzazione e l'incremento
    printf("Questo loop non finisce mai \n");
```

Per creare un loop senza fine molti preferiscono usare la struttura while(1).

L'istruzione break spesso permette la uscita istantanea da un loop infinito:

```
For(;;)  {      //Ciclo infinito
  car=getch(); //Attendi pressione tasto
  if(car=='A') //Premuto 'A' ?
    break;     //Si!! Esci dal loop
}
Printf("Hai inviato 'A' \n");
```

```
While(1) {      //Ciclo infinito
  car=getch(); //Attendi pressione tasto
  if(car=='A') //Premuto 'A' ?
    break;     //Si!! Esci dal loop
}
Printf("Hai inviato 'A' \n");
```

http://www.mrscuole.net/anxapic/

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

8. Strutture di controllo

→ Loop privi di corpo

La sintassi del C permette di avere istruzioni vuote, il corpo di un loop può essere vuoto; ciò può essere sfruttato per migliorare l'efficienza di certi programmi e anche per generare cicli di ritardo.

/* Questo esempio genera un ritardo per inizializzare, incrementare e verificare la variabile i per 1000 volte */

```
For(t=0; t<1000; t++) ; //Notare il ; alla fine del costrutto for – Ciclo di ritardo per i che và da 0 a 1000
```

/* Questo esempio genera un loop di attesa infinito; la condizione è sempre vera e poi non "succede" nulla in quanto il corpo del ciclo while è vuoto*/

```
while(1);  //Notare il ; alla fine del costrutto while – Ciclo di attesa infinito
```

La istruzione "vuota" (null) è molto simile al comando NOP (NO oPeration) in assembler.

Programmare i microcontrollori in linguaggio C

Il CCS Pic-C Compiler --- Componenti di un programma in C

8. Strutture di controllo

→ Return

L'istruzione return termina l'esecuzione di una funzione e ritorna il controllo alla funzione chiamante. Un valore può essere restituito alla funzione chiamante se richiesto dalla stessa. Una funzione che non deve restituire nessun valore alla funzione chiamante può essere dichiarata di tipo **void** (Ricordiamo che **void** è una estensione proposta dall'ANSI-C e non necessariamente disponibile in tutti i compilatori; il CCS Pic-C compiler dispone di questa estensione).

Tutte le funzioni, ad eccezione di quelle dichiarate di tipo **void**, restituiscono un valore; tale valore può essere esplicitato con l'istruzione **return** oppure restituiscono il valore 0 (zero) se non viene esplicitamente specificato in **return.**

```
Int8 GetValue() {
  int8 c;       //var locale
  c=12;
  return c;    //return val c
}
```

Questa funzione ritorna alla funzione main il valore di c, notare che c è una variabile locale. La funzione è stata dichiarata di tipo int8 perché ritorna un valore di questo tipo

```
Void GetNothing()  {
  int8 c;  //var locale
  c=22;
  return;
}
```

Questa funzione non ritorna nessun valore infatti è stata dichiarata di tipo void

```
Void Main()  {
  int8 x;
  x=GetValue(); //X=12
  GetNothing();
}
```

Questa programma memorizza in x il valore di c che ritorna la funzione chiamata GetValue() Main() è una funzione di tipo void perché non ritorna nessun valore

http://www.mrscuole.net/anxapic/