Un programme réalise des fonctionnalités. Dans la phase d'analyse du cahier des charges, l'algorithme consiste à traduire chacune de ces fonctionnalités en une ou plusieurs fonctions. Une fonction principale appelé ***main()*** aura pour mission de gérer les appels à ces fonctions pour réaliser la totalité du programme.

On déclare la fonction entre **l'entête** de l'algorithme et le programme principal **main()**. On précise **le type** du paramètre qui sera retourné lors de l'appel de la fonction. On précise également les **arguments** (ou paramètres) et leurs **types** qui seront transmis à la fonction. Ces arguments seront disponibles (ou visibles) à l'intérieur de la fonction.

Exemple :

```
#include     …….....        // entêtes
#include     …….....

// Type de retour     nom fonction   type argument     argument

float        Doubler( float     z ) ;   // déclaration de la fonction  entre l'entête  et le programme principal
// ou float         Doubler( float       ) ;

int main()                  // programme principal
{
  float resultat ;
  resultat = Doubler(15) ;          // Appel de la fonction
  return 0;
}

// La fonction
// Type de retour   nom fonction    type argument     argument

float Doubler  (       float      x )          // ici l'argument est nommé x et est de type float
{
        return 2.0*x;                    // multiplication par 2 de x
}
```

L'écriture d'une fonction requiert donc une **déclaration** et **un corps** :

- **La déclaration** comprend : le type de retour, le nom de la fonction suivi d'une liste d'arguments entre parenthèses. Les arguments sont séparés par une virgule et sont notés : type arg1 nom arg1, type arg2 nom arg2, …
  S'il n'y a pas de type de retour ou d'argument, on utilise le mot **void** à la place du type de retour et de l'argument.

- **Le corps** est un bloc d'instructions

Une fonction possède trois aspects :

- le prototype : c'est la déclaration qui est nécessaire avant tout ;
- l'appel : c'est l'utilisation d'une fonction à l'intérieur d'une autre fonction (par exemple le programme principal main()) ;
- la définition : c'est l'écriture proprement dite de la fonction (**corps**).

Exemple avec un microcontrôleur pIC

```
#include <xc.h>              // entêtes
#include <stdio.h>


#pragma config FOSC = HS // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT enabled)

#define _XTAL_FREQ 4000000

/ * déclaration des protoypes des focntions

void init_ES(void);                 // pas d'argument (void) ni de type de retour (void)

void init_serie(void);              // pas d'argument (void) ni de type de retour (void)

void init_a2d(void) ;               // pas d'argument (void) ni de type de retour (void)

void clignote(unsigned char);    // un argument (unsigned char) et pas de type de retour (void)

unsigned int  read_a2d(unsigned char);  // un argument (unsigned char) et un de type de retour
                                        //  ( unsigned int )

void putch(char c);                     // un argument (char) et pas de type de retour (void)


char getch(void);                       // pas d'argument (void) et un type de retour (char)

char car_lu ;
unsigned int N=0 ;

void main()
{
        init_ES( );          // Appel de la fonction initialisation des entrées et sorties
       init_a2d( ) ;         // Appel de la fonction initialisation du C.A.N
       init_serie( );        // Appel de la fonction initialisation de la liaison série

       clignote(5) ;  //Appel de la fonction clignote avec l'argument 5

      __delay_ms(2000); //temporisation de 2 secondes

     // Boucle infinie
       while(1)
        {
           car_lu = getch() ;    // lecture d'un caractère envoyé par le Terminal
           // printf(……)         // affichage de car_lu
```

```
            N = read_a2d(0);     // lecture de la valeur renvoyée par le C.A.N
            //   printf(……)       // affichage de N
           __delay_ms(1000);

        }

}
// Fonction clignotement
void clignote(unsigned char n)
{
   int i;
   for(i=0;i<n;i++)
   {
    PORTD = 255;
    __delay_ms(100);
   PORTD = 0;
    __delay_ms(100);
   if(RB0==0) break;
   }
}

// Fonction initialisation des entrées sorties
void init_ES(void)
{
  TRISD=0x00; // Configuratio PORTD en sortie
   TRISA0=1;
   ANS0=1;
   TRISC6=0;
}

// Fonction initialisation du CAN
void init_a2d(void)
{
        ADCON0=0xC0;       // select Fosc/2
        ADCON1=0x80;       // select rightjustify result. A/D port configuration 0
        ADON=1;            // turn on the A2D conversion module
}

// Fonction lecture résultat de la conversion analogique numérique
unsigned int  read_a2d(unsigned char channel){
        channel&=0x0F;       // truncate channel to 4 bits
        ADCON0&=0xC3;    // clear current channel select
        ADCON0|=(channel<<2);     // apply the new channel select
   GO=1;       // initiate conversion on the selected channel
        while(GO)continue;
        return(256*ADRESH + ADRESL);   // return 8 MSB of the result
}

// lecture d'un caractère envoyé par le terminal vers le microcontrôleur
char getch()
{
   while(!RCIF);  // Attente arrivée caractère
   return RCREG;  // Renvoi du caractère reçu
}
```

```
// Ecriture d'un caractère vers le Terminal
void putch(char data)
{
   while(!TRMT);  // Attente buffer vide
   TXREG = data;  // envoi caractère
}
```