Cơ bản về lập trình C

Cơ bản về lập trình C_ Phần 1

Nguồn : biendt.biz

Để lập trình được vi điều khiển ta phải biết được 2 ngôn ngữ thông dụng nhất hiện nay là C và ASM với hai loại ngôn ngữ này ta có thể lập trình điều khiển được các dòng vi điều khiển. ở đây tôi chỉ giới thiệu với các pác về ngôn ngữ lập trình C thôi!

Ngôn ngữ lập trình C là một ngôn ngữ bậc cao nó hơn hẳn ASM nhưng nó được ra đời sau. Ngôn ngữ C thì cần phải hiểu bản chất của nó chứ không như ASM nên thế các pác phải học lý thuyết về ngôn ngữ C trước đã.

*** : Các từ khóa trong C**

+ Những từ khóa sau đây không được dùng làm tên biến hay tên hàm:

auto, break, case, char, continue, default, do, double, else, extern, float, for, goto, if, int, long, register, return, short, sizeof, static, struct, switch, typedef, union, unsigned, while.

+ Ngoài ra còn nhưng từ khóa đặc biệt khác như là : void, const, enum, volatige

*** : Các kiểu khai báo biến trong C**

Tên biến	Số bit	Số byte	Giá trị
Char	8	1	-128 đến -127
unsigned char	8	1	0 đến 255
short	16	2	-32769 đến 32767

unsigned short	16	2	0 đến 65535
int	16	2	-32768 đến 32767
unsigned int	16	2	0 đến 65535
long	32	4	
unsigned long	32	4	0 đến 4,294,697,295

ví dụ : Khai báo một biến là : unsigned char x; biến này là biến kí tự được nhận giá trị từ 0 đến 255

Mặt khác khi khai báo biến ta có thể gán luôn giá trị vào cho biến như unsigned char x=0; và cũng có thể khái bào biến cùng 1 lúc như : unsigned int x,y,x;

*** Lời giải thích:**

Tùy theo mặc định trong C không cho phép ta cho các lời giải thích lòng vào với nhau. Để lồng cái này thì tôi không biết (Các pác thông cảm)

+ Lời giải thích dài : Được đặt giữa dấu :/* và */

+ Lời giải thích ngắn : Được đặt sau dấu //

Cơ bản về lập trình C_ Phần 2

Nguồn : biendt.biz

Tiếp đến phần này ta tìm hiểu về : biểu thức đơn giản và câu lệnh gán, thứ tụ ưu tiên của các toán tử , 1 số tên hàm thường dùng và các kiểu khai báo thư viện.

*** Biểu thức đơn giản và các câu lệnh gán.**

Các biểu thức tính toán và các câu lệnh gắn trong C được quy định sắn chúng ta chỉ việc áp dụng vào thôi chứ không được sáng tạo ra đâu. Pác nào mà sáng tạo ra thì chương trình của pác không chạy được.

+ Số học và thao tác bít:

Các biểu thức	Chức năng
+	Phép cộng
-	Phép trừ
*	Phép nhân
/	Phép chia
%	Moduule toán học hay lấy phần trăm
&	Phép hội các bit
|	Phép tuyển các Bit
^	Phép tuyển có loại trừ (XOR)
~	Đảo toàn bộ các bit
>>	Dịch trái bit
<<	Dịch phải bít

Ví dụ : unsigned char x,y,z; x=y+z;

+ Các lệnh Logic

&&	Phép hội Logic (AND)
||	Phép tuyển Logic (OR)
!	Phép phủ định(Not)
<	Phép nhỏ hơn
>	Phép lớn hơn
<=	Phép nhỏ hơn hoặc băng
>=	Phép lớn hơn hoặc bẳng
==	Phép bằng
=	Phép gán giá trị
!=	Phép không bằng hay khác
++	Phép tăng giá trị lên 1 giá trị
--	Phép giảm giá trị đi 1 giá trị
+=	Phép tăng giá trị lên n lần
-=	Phép giảm giá trị đi n lần

Ví dụ : if(x!=y) z=0; hay x++, x+=y

*** Thứ tự ưu tiên của các toán tử**

Các toán tử khác nhau không cùng 1 mức ưu tiên tức là một số phép tính sẽ

được thực hiện trước. Các toán tử ở dòng 1 có mức ưu tiên hơn dòng 2 và cũng như vậy như các dòng tiếp theo

Ví dụ : < <== thứ tự của chúng được thực hiện từ trái sang phải tức là từ < ....<==

Chú ý : Có 14 quy tắc ưu tiên trong C chảng hạn toán tử && được thực hiện trước toán tử || nhưng sau toán tử <<,..., Chủ yếu nó được thực hiện từ trái sang phải. Nói cái này khó hiểu quá . Tôi cũng chả hiểu được phần này đọc mãi mà vẫn thấy không hiểu được (Các pác thông cảm)

*** Một số tên hàm thường dùng trong :**

Tên hàm	Nội dung
sqpt (x)	Tính căn bậc 2 của giá trị x
exp(x)	Tính e mũ của giá trị x
log(x)	Tính logarit cơ số tự nhiên của giá trị x
log10(x)	Tính logarit của cơ số 10 của giá trị x
fabs(x)	Trị tuyệt đối của x
floor (x)	Làm tròn giá trị x
fmod(x,y)	Phần dư của phép chia cho x
sin(x)	Tính sin(x)
cos(x)	Tính cos(x)
x%y	Lấy phần dư của x chia y

Các giá trị của x, y là số thực
Ví dụ : 16%7 giá trị này là lấy phần dư của 16/7

*** : Kiểu khai báo thư viện**

Chỉ thị #include chỉ cho phép vi xử lý nhận nội dung của tệp khác và nhận chúng vào chương trình.
Các dạng chung của một chỉ thị bao hàm:

#include / / Tệp bao hàm hệ thống chuẩn

#include"file.h" / / Tệp bao hàm cục bộ

Nếu tệp tiêu đề được đặt trong nháy kép ("") thì tệp là cục bộ và C tìm thấy tệp này trong các thư mục hiện tại. Nếu tệp được đặt trong () thì nó là tệp hệ thống chuẩn và nó được tìm thấy trong các thư mục include.

Cơ bản về lập trình C _ Phần 3

Nguồn : biendt.biz

Ở trong phần này chúng ta cùng tìm hiểu về cấu trúc hàm trong C, Mảng trong C và bộ tiền xử lý trong C

* : Cấu trúc hàm trong C

Hàm dùng để chứa các chỉ thị có thể thực hiện được vào chương trình ngắn gọn.Trong C có hai kiểu cấu trúc hàm :

+ Hàm trả lại giá trị

Cấu trúc : kieugiatritralaiham tên hàm(Biên truyền vào hàm)

{

Các câu lện xử lý ở đây;

}

Trong hàm này thường được sử dụng lệnh return để truyền 1 giá trị ra ngoài hàm. Tất cả các hàm trừ void đều được xử dụng bằng lệnh return

ví dụ :

```
unsigned char biendt(unsigned char x, unsigned char y)

{

x=x+y;

reture x;

}
```

+ Hàm không trả lại giá trị

Cấu trúc : void ten_ham()

{

Các câu lệnh thực hiện ở đây;

}

kiểu giá trị hàm này cũng dễ hiểu hơn cái này là đặc trưng cho C.

Ví dụ:

void biendt() // Khai bao ham mang ten biendt

{

unsigned char x,y;

x+=y;

}

Ngoài ra trong C nó còn có 1 hàm ngắt phần này tương đối khó hiểu nên các pác tìm hiểu thêm!

*** Mảng trong C**

Mảng là được dùng để lưu các nhóm dữ liệu giống nhau. Khuôn dạng mảng 1 chiều:

cấu trúc : kiểu tên[số phần tử];

Trong đó : Kiểu là kiểu dữ liệu trong C như int, char
tên là tên biến

số phần tử là một số nguyên chỉ giá trị lớn nhất của mảng.

Ví dụ : int biendt[3]={0,1,2}; // mảng này gồm 3 phần tử có độ dài là 3

Ngoài ra nó còn mảng nhiều chiều thường được sử dụng mảng 2 chiều và 3 chiều

cấu trúc : kiểu tên [số pt1][số pt2]...[];

ví dụ: unsigned char biendt[20][30];

*** Bộ tiền xử lý.**

Ngoài kiểu khai báo thư viện trong C thì nó còn câu lệnh #define. Câu lệnh này cho phép người lập trình định nghĩa trực tiếp các biến hơn thế nữa nó còn cho phép định nghĩa một maco hay thay thế đơn giản.

Ví dụ:

#define biendt 100

ở trên ta nhận thấy rằng biến biendt được gắn bằng 1 giá trị là 100. Kết thúc phần này !

Cơ bản về lập trình C _ Phần 4

Nguồn : biendt.biz

Trong phần này mình sẽ đi tìm hiểu các cấu trúc lập trình cơ bản trong C như : if, while, for...

* Cấu trúc có điều kiện IF

Nếu giá trị biểu thức khác không thì câu lệnh sẽ được thực hiện

+Cấu trúc : if (biểu thức) câu lệnh;

hay if(biểu thức)
{

câu lện 1;
câu lệnh 2;
.....
Câu lện n;
Còn nếu điều kiện sai thì các câu lệnh dưới if sẽ không được thực hiện

Ví dụ :

```
unsigned int i,j;
if(++i>100) j++;
```

+ Ngoài ra chúng ta còn sử dụng cấu trúc if - else. Nếu biểu thức trong if không đúng thì nó thực hiện câu lệnh dưới esle

```
if(biểu thức)
{
câu lệnh 1;
câu lệnh 2;
.....
câu lệnh n;
}
else
{
câu lệnh 1;
câu lệnh 2;
.....;
câu lệnh n;

}
```

Chú ý:

-Trong C các biểu thức điều kiện ngoài biểu thức quan hệ có thể là một biểu thức số , nếu giá trị biểu này bằng 0 sẽ nhận giá trị sai, nếu khác không nhận giá trị đúng

- Trước ELSE mà chỉ có 1 câu lệnh thì kết thúc lệnh phải có dấu ; của lệnh if
- Biểu thức điều kiện phải đặt giữa ()

+ Các điều kiện lồng nhau:

```
if(biểu thức 1) lệnh 1;
else(biểu thức 2) lệnh 2;
```

else(biểu thức 3) lệnh 3;

.....

else lệnh n;

* Cấu trúc vòng While

Dạng của nó như sau:

while (điều kiện) statement

while(1) {};

Tạo vòng lặp mãi mãi , rất hay đùng trong lập trình VXL .Chương trình chính sẽ được viết trong dấu

ngoặc.

Vòng lặp do-while

Dạng thức:

do statement while (điều kiện);

do

{

x++; // cho nay cac ban co the viet nhieu cau lenh ,

}

while(x>20)

tăng giá trị của x cho đến khi x > 10

Chức năng của nó là hoàn toàn giống vòng lặp while chỉ trừ có một điều là điều kiện điều khiển vòng

lặp được tính toán sau khi statement được thực hiện, vì vậy statement sẽ được thực hiện ít nhất

một lần ngay cả khi condition không bao giờ được thoả mãn .Như vd trên kể cả x >20 thì nơ vẫn
tăng giá trị 1 lần trước khi thoát
nếu x=100 thì tăng x thêm 1 còn không thì giảm x. Nói chung câu lệnh while(1) thường được sử dụng sau void main()

*** Cấu trúc for**

Vòng lặp for .
Dạng thức:
for (điều kiện 1; điều kiện 2; điều kiện 3) câu lệnh;
và chức năng chính của nó là lặp lại câu lệnh chừng nào condition còn mang giá trị đúng, như
trong vòng lặp while. Nhưng thêm vào đó, for cung cấp chỗ dành cho lệnh khởi tạo và lệnh
tăng. Vì vậy vòng lặp này được thiết kế đặc biệt lặp lại một hành động với một số lần xác định.
Cách thức hoạt động của nó như sau:
1, điều kiện 1 được thực hiện. Nói chung nó đặt một giá khí ban đầu cho biến điều khiển. Lệnh này được thực hiện chỉ một lần.
2, condition được kiểm tra, nếu nó là đúng vòng lặp tiếp tục còn nếu không vòng lặp kết thúc và lệnh được bỏ qua.
3, câu lệnh được thực hiện. Nó có thể là một lệnh đơn hoặc là một khối lệnh được bao trong
một cặp ngoặc nhọn.
4, Cuối cùng, điều kiện 3 được thực hiện để tăng biến điều khiển và vòng lặp quay trở lại bước 2.

Phần khởi tạo và lệnh tăng không bắt buộc phải có. Chúng có thể được bỏ qua nhưng vẫn phải
có dấu chấm phẩy ngăn cách giữa các phần. Vì vậy, chúng ta có thể viết for (;n<10;) hoặc for
(;n<10;n++).
Bằng cách sử dụng dấu phẩy, chúng ta có thể dùng nhiều lệnh trong bất kì trường nào trong
vòng for, như là trong phần khởi tạo. Ví dụ chúng ta có thể khởi tạo một lúc nhiều biến trong
vòng lặp:

```
for ( n=0, m=200 ; n!=m ; n++, m-- )
{
câu lệnh
}
```

Ví dụ dùng trong vi điều khiển:

```
void delay(unsigned int t)

{
unsigned int i,j;
for(i=0;i<100;i++)
for(j=0;j
}
```

*** Câu lệnh rẽ nhánh và nhảy**

+ Lệnh break.

Sử dụng break chúng ta có thể thoát khỏi vòng lặp ngay cả khi điều kiện để nó kết thúc chưa được thoả mãn. Lệnh này có thể được dùng để kết thúc một vòng lặp không xác định hay buộc nó phải
kết thúc giữa chừng thay vì kết thúc một cách bình thường. Ví dụ, chúng ta sẽ dừng việc đếm
ngược trước khi nó kết thúc:

+ Lệnh continue.

Lệnh continue làm cho chương trình bỏ qua phần còn lại của vòng lặp và nhảy sang lần lặp tiếp
theo. Ví dụ chúng ta sẽ bỏ qua số 5 trong phần đếm ngược:

+ Lệnh goto.

Lệnh này cho phép nhảy vô điều kiện tới bất kì điểm nào trong chương trình. Nói chung bạn nên
tránh dùng nó trong chương trình C++. Tuy nhiên chúng ta vẫn có một ví dụ dùng lệnh goto để đếm
ngược:
+ Hàm exit.

Mục đích của exit là kết thúc chương trình và trả về một mã xác định. Dạng thức của nó như sau

void exit (int exit code);
exit code được dùng bởi một số hệ điều hành hoặc có thể được dùng bởi các chương trình gọi.
Theo quy ước, mã trả về 0 có nghĩa là chương trình kết thúc bình thường còn các giá trị khác 0 có
nghĩa là có lỗi.
các lệnh trên mình chủ yếu chỉ dùng lệnh break để thoát khỏi vòng lặp . Các lệnh khác thường rất ít
dược sử dụng

***Cấu trúc lựa chọn: switch.**

Cú pháp của lệnh switch hơi đặc biệt một chút. Mục đích của nó là kiểm tra một vài giá trị hằng
cho một biểu thức, tương tự với những gì chúng ta làm ở đầu bài này khi liên kết một vài lệnh if
và else if với nhau. Dạng thức của nó như sau:
Code:

```
switch (bieu thuc)

{

case 0:
câu lệnh;
break;
case 1:
câu lệnh ;
```

```
break;
.
.
.
default:
case n: câu lệnh ;break;
```

Ví dụ của câu lệnh này:

```
void hienthi(unsigned char x)
{
switch(x)
{
case0:PRT1DR=0x00;break;
case1:PRT1DR=PRT1DR&0xfe;break;
case2:PRT1DR=PRT1DR&0xfd;break;
}

void main()
{
unsigned char n;
for(n=0;n<3;n++)
{
hienthi(n);
delay();
}}
```

Tôi đã nói về những vấn đề cơ bản về C ứng dụng cho vi điều khiển. Các pác chỉ cần nắm vững nhứng điều trên thì có lập trình cho vi điều khiển dùng C là OK! Chúc các pác học tốt!