****

**C語言教材**

[1-1.什麼是C語言?](http://163.28.10.78/content/junior/computer/ks_mc/chapter/ch08/c08_03_02.htm#1-1)
[2-1.變數命名的原則?](http://163.28.10.78/content/junior/computer/ks_mc/chapter/ch08/c08_03_02.htm#2-1)
[3-1.自動變數(autovariable)](http://163.28.10.78/content/junior/computer/ks_mc/chapter/ch08/c08_03_02.htm#3-1)
[4-1.格式化輸出函數printf()](http://163.28.10.78/content/junior/computer/ks_mc/chapter/ch08/c08_03_02.htm#4-1)
[5-1.算術運算元(Arithmeticoperators)](http://163.28.10.78/content/junior/computer/ks_mc/chapter/ch08/c08_03_02.htm#5-1)
[6-1.控制結構(controlstructure)](http://163.28.10.78/content/junior/computer/ks_mc/chapter/ch08/c08_03_02.htm#6-1)

1-1.什麼是C語言?

C語言是1972年左右由DennisRitchie一個人在貝爾實驗室(簡稱AT&T)設計開發出來的電腦程式語言.他當時與KenThompson兩人共同負責UNIX作業系統的發展,而UNIX作業系統(是一個多人多工的作業系統)與大部份的公用程式及C本身的編譯器(Compiler)絕大部份是由C設計出來的,且C的可攜性(Portability)極佳,如果須跨平台執行,只需修改部份程式碼(甚至不需修改)即可執行.挾其強大的程式能力,相信C語言是值得學習的

1-2.在學C之前先來瞭解何謂編譯器(COMPILER)與直譯器(INTERPRETER)?

(1)編譯器就是一種能接受高階語言程式當作輸入,而產生等效的機械語言為輸出的軟體程式,例如TurboC,C++,BorlandC++,MicrosoftC,VisualC++etc.
(2)編譯器基本上由語言分析(LexicalAnalysis),語法分析(SyntaxAnalysis),編碼產生器(CodeGenerator),符號表(SymbolTable)所組成.

(4)連結器(LINKER)的用途是將相關的程式目的碼(由編譯器所產生之OBJ檔)與庫存函數連結起來.

(5)C語言程式需經由編譯器依下列步驟才能得到最後的可執行檔.

|  |
| --- |
| **原始程式-->經由編譯器-->變成目的檔-->經由連結器-->變成執行檔SOURCE.C-->COMPILER-->SOURCE.OBJ-->LINKER-->SOURCE.EXE** |

1-3.從程式寫作的角度來看C語言的特性.

(1)可以像組合語言一樣進行硬體之直接存取.
(2)為語法規則簡單,清楚,容易使用之結構化語言.
(3)必要時可以與組合語言連結.
(4)可攜性(Portability)極佳,跨平台的能力強.

1-4.標準C語言格式.

|  |
| --- |
| **#include<stdio.h>---->前置處理器main()---->主程式{inti=0;/\*變數宣告\*/i=i+1;/\*算數運算\*/---->程式主體printf("i=%d\n",i);/\*輸出至螢幕\*/}** |

備註:
(1){與}符號表示程式或敘述的開始與結束.
(2);符號表示變數宣告或指令敘述的段落.
(3)/\*與\*/符號表示程式內註解說明.
(4)主程式main為C語言程式開始執行時的進入點.

2-1.變數命名的原則?

(1)開頭必須是字母(letter)或底線(underscore).
例如：a,b,count,test\_var,\_date,.....etc.
(2)字元的大寫小寫所代表的意義不同.
例如：a,B,A,TEST\_VAR,test\_var,.....etc.
(3)不得使用關鍵字(keyword).
例如：asm,double,goto,static,auto,else,if,struct,sizeof,break,entry,int

2-2.變數的資料型態種類:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **類別** | **符號位元** | **位元長(bits)** | **表示法** | **數值範圍** |
| 整數 | 有 | 16 | int(short) | -32768->32767 |
| 32 | long | -2147483648->2147483647 |
| 無 | 16 | unsignedint | 0->65535 |
| 16 | unsignedshort | 0->65535 |
| 32 | unsignedlong | 0->4294967295 |
| 浮點數 | 有 | 32 | float | 10^-38->10^38 |
| 64 | double | 10^-308-->10^308 |
| 字元 | 無 | 8 | char | 0->255 |

備註:
(1)float的十進位數精確度最大為七位數.
(2)double的十進位數精確度最大為十五位數.

2-3.變數的宣告方法:

(1)整數部份:

{
inta;
intb,c=45;
intd=10L;/\*此字母L(大小寫均可)表示將d轉換成long來處理\*/
shortintname;
shorttest\_var;
longintvar\_name;
long\_variable;
unsignedintargu1=30;
.
.
}

(2)浮點數部份:

{
floata=12.3456;/\*單精準度\*/
floatb=0.12345e2;
doublescore=4.987654322e-7;/\*雙精準度\*/
.
.
}

(3)字元部份:

{
charc='c';
charbell=7;/\*7為ASCII之響聲字元\*/
.
.
}

備註:
(1)C語言中變數須在程式最上端宣告,而C++不在此限制.

3-1.自動變數(autovariable)

自動變數的宣告如下所示：

{
autointa;
autointb=12345;
autocharc;
autofloatd=123.45;
.
.
}

自動變數只在它所定義的區塊內有效。只要在變數所屬的區塊結構內執行，該變數的資料是有效而正確的。當程式執行離開了該區塊，所有於區塊內定義的自動變數就不存在了。自動變數宣告時所使用的關鍵字auto在實際的運用上可省略不寫。

main()
{
intx=1;
inner();
printf("%d\n",x);
}

inner()
{
intx=2;
printf("%d\n",x);
}

result:
2
1

3-2.暫存器變數(registervariable)

要定義暫存器變數必須使用關鍵字register。此類變數的使用與自動變數相似。
我們都知道，電腦內部的暫存器在資料存取或是運算的速度都比記憶體快很多，所以如果暫存器變數宣告成功，則有助於程式整體效率之提升。但是暫存器變數之宣告是一種請求而非命令，所以暫存器變數若宣告失敗，Ccompiler自動將此變數當作自動變數來使用。

main()
{
registerintx=1;
inner();
printf("%d\n",x);
}

inner()
{
registerintx=2;
printf("%d\n",x);
}

result:
2
1

3-3.靜態變數(staticvariable)

靜態變數的宣告如下所示：

{
staticinta;
staticintb=12345;
staticcharc;
staticfloatd=123.45;
.
.
}

靜態變數與自動變數一樣，是某特定函數內的區域性變數，但與自動變數不同的是靜態變數的值不會因函數的執行結束而消失，所以如果程式再回到函數內執行時，上一次執行結果還是存在變數內。

例一
main()
{
increment();
increment();
increment();
}

increment()
{
intx=0;
x=x+1;
printf("%d\n",x);
}

result:
1
1
1

例二
main()
{
increment();
increment();
increment();
}

increment()
{
staticintx=0;
x=x+1;
printf("%d\n",x);
}

result:
1
2
3

3-4.外部變數(externvariable)

外部變數和前面所提到的變數不同。外部變數的有效範圍不是區域性，而是整體性(global)，外部變數定義在任何函數的外面，表示可以被其他函數所共用。

intx=123;
main()
{
printf("%d\n",x);
}

result:
123

ps.在此補充一點；如果宣告自動變數、暫存器變數、靜態變數或外部變數時未設定初始值，則Ccompiler將會設定為0。

4-1.格式化輸出函數printf()

printf()是用來將資料格式化的輸出至螢幕，其語法如下：

printf(format\_string,arg1,arg2,...argn);

由下面的例子中%d這種由符號%帶頭的，我們稱之為"轉換規格"(conversionspecification)，字元d稱為"轉換字元"(conversioncharacter)；請參考表一。另外一提的是C語言定義的特殊控制碼，如表二。

main()
{
intx=1,y=2;
printf("Example4-1-1:\n");
printf("x=%d\n",x);
printf("x=%dy=%d\n",x,y);
}

result:
x=1
x=1y=2

|  |
| --- |
| 表一.轉換字元的種類 |
| **種類** | **表示法** | **功能敘述** |
| 整數(integer) | d | 以十進位方式印出。 |
| o | 以八進位方式印出。 |
| x | 以十六進位方式印出。 |
| u | 以不帶符號的十進位方式印出。 |
| l | 以長整數(long)方式印出。 |
| 浮點數(float) | f | 以xxx.xxx方式印出。 |
| e | 以指數的方式印出。 |
| 字元(char) | c | 以字元方式印出。 |
| s | 以字串方式印出。 |
| 其他(other) | - | 向左邊靠齊印出。 |
| dd | 指定欄位寬。 |
| . | 分隔欄寬。 |
| \*.\* | 指定浮點數之精確度。 |

main()
{
intx=42;
floaty=12.345;
charc='A',c1[4]="ABC";
printf("%d%o%x%u\n",x,x,x,x);
printf("%15d\n%-15d\n",x,x);
printf("%f%e\n",y,y);
printf("%08.4f\n",y);
printf("%0\*.\*f\n",8,4,y);
printf("%c%s\n",c,c1);
printf("%-10c\n%15.2s\n",c,c1);
}

result：
42522a42
42
42
12.3450001.23450e+01
012.3450
012.3450
AABC
A
AB

|  |
| --- |
| 表二.特殊控制碼 |
|  |
| **種類** | **功能敘述** |
| \n | newline新行。 |
| \r | carriagereturn回歸鍵。 |
| \t | tab跳格。 |
| \b | backspace退位。 |
| \f | formfeed跳頁。 |
| \\ | backslash反斜線。 |
| \' | singlequote單引號。 |
| \" | doublequote雙引號。 |

4-2.格式化輸入函數scanf()

scanf()是與printf()相對應的函數，提供了許多相似但是方向相反的轉換功能，其語法如下：

scanf(format\_string,argptr1,argptr2,...argptrn);

如同printf()，scanf()的引數數目亦不固定，但二者有一很大的不同，scanf()的引數型態為指標，這是必要的，因為scanf()從標準輸入依轉換規格取得輸入值，然後存放到各對應的引數所指向的位址中。請參考。此處所提到的轉換規格同printf()函數。

main()
{
intx;
floaty;
charz;
scanf("%d%f%c",&x,&y,&z);
printf("%d%f%c\n",x,y,z);
scanf("%3d%4f",&x,&y);
printf("%d%f\n",x,y);
}

result：
2412.45G------>此處為鍵盤輸入之資料
2412.450001G
1234567890------>此處為鍵盤輸入之資料
1234567.000000

4-3.getchar()與putchar()函數

getchar()是一個基本函數，它不需要引數，但是會傳回給程式一個int型態的值。這個返回值就是一個ASCII字元(由鍵盤輸入的)。所以我們知道getchar()的作用是至標準輸入(keyboard)取得一字元資料。相反的putchar()是將括號內所指定的變數或字元送至標準輸出(monitor)。

|  |
| --- |
| getchar()與getch()的比較 |
| **函數名** | **型態** | **功能** |
| getchar() | bufferedinput | 待一字元輸入後需加一Enterkey才算完成。 |
| getch() | nobufferedinput | 待任何字元輸入後即完成。 |

main()
{
intx;
x=getchar();
printf("%d%c\n",x,x);
}

result:
A------>此處為鍵盤輸入之資料
65A

main()
{
intx;
x=getchar();
putchar(x);
putchar('\n');
putchar('A');
}

result：
A------>此處為鍵盤輸入之資料
A
A

4-4.gets()與puts()函數

基本上gets()，puts()與getchar()，putchar()的功能類似，差異僅在於前者以字串當作輸入/輸出，而後者是以字元當作輸入/輪出。

main()
{
charstring[10];
puts("Pleaseinputastring:");
gets(string);
puts(string);
}

result:
Pleaseinputastring:
mybook------>此處為鍵盤輸入之資料
mybook

5-1.算術運算元(Arithmeticoperators)
(1)指定運算(assignmentoperator)
將整數、浮點數或字元指定給某一不特定的變數之動作稱之。
main()
{
intx,x1,x2,x3;
floaty,y1;
charz;
x=1;
x1=x2=x3=2;/\*多重指定運算\*/
y=59.5;
y1=2.5e12;
z='c';
printf("%d\n",x);
printf("%d%d%d\n",x1,x2,x3);
printf("%f%e%c\n",y,y1,z);
}

result：
1
222
59.5000002.50000e+12c

(2)加、減、乘、除、負數及模數運算
加(+)、減(-)、乘(\*)、除(/)為一般的四則運算，在此不多述。而模數運算(%)是經由兩常數相除所得的餘數稱之。負數是在某個常數前面加個減號所成的組合稱之。
main()
{
intc1,c2,c3,c4,c5,c6;
c1=c2=c3=c4=c5=20;
c1=c1+5;
c2=c2-10;
c3=c3\*5;
c4=c4/10;
c5=c5%6;
c6=-5;
printf("%d%d%d%d\n",c1,c2,c3,c4);
printf("%d\n",c5);
printf("%d\n",c6);
}

result：
25101002
2
-5

(3)運算的優先權
\*、/、%、+、-以上所標示的運算元，其運算的優先順序由左至右。
(4)遞增與遞減
遞增與遞減又分為前置模式(prefix)與後置模式(postfix)；前置模式是將++或--置於變數之前。而後置模式是將++或--置於變數之後。
main()
{
inta=1,b=1,c=1,d=1;
intaplus,bplus;
intcminus,dminus;
aplus=++a;
bplus=b++;
cminus=--c;
dminus=d--;
printf("%d%d\n",aplus,bplus);
printf("%d%d\n",a,b);
printf("%d%d\n",cminus,dminus);
printf("%d%d\n",c,d);
}

result：
21
22
01
00

5-2.邏輯運算元(Logicaloperators)
(1)邏輯運算的種類：

|  |
| --- |
| 表一.邏輯運算元 |
| **運算元** | **功能敘述** | **運算元** | **功能敘述** |
| & | AND(及) | ~ | 1'complement(一的補數) |
| | | OR(或) | << | Leftshift(左移) |
| ! | NOT(反向) | >> | Rightshift(右移) |
| ^ | ExclusiveOR(互斥或) |  |  |

main()
{
intland=0x0003,lor=0x0003;
intlxor=0x000c,lcomp=0x00f0;
intlleft=0x0003,lright=0x0080;
land=land&0x000c;
lor=lor|0x000c;
lxor=lxor^0x000c;
lcomp=~lcomp;
lleft=lleft<<5;
lright=lright>>2;
printf("%x%x\n",land,lor);
printf("%x%x\n",lxor,lcomp);
printf("%x%x\n",lleft,lright);
}

result：
0f
0ff0f
6020

!(NOT)、~(1'complement)、<<(Leftshift)、>>(Righshift)、&(AND)、^(ExclusiveOR)、|(OR)，以上所列之邏輯運算元其運算的優先順序依排列順序由左至右為之。另外一提的是算術運算元其運算的優先權高於邏輯運算元。

5-3.關係運算元(Relationaloperators)
關係運算元一般用於控制流程或有條件的敘述當中，並非本章的討論重點，所以僅列於表二，待以後相關的章節中再詳細討論之。至於其運算的優先權則是依表列由上而下、由左至右行之。

|  |
| --- |
| 表二.關係運算元 |
| **運算元** | **功能敘述** | **運算元** | **功能敘述** |
| < | 小於 | >= | 大於等於 |
| <= | 小於等於 | == | 等於 |
| > | 大於 | != | 不等於 |

5-4.複合型指定運算
在例一中為C語言的一般運算式，而例二為C語言的複合型指定運算；此處的exp1為變數，而exp2為變數或運算式，op為運算元。我們在第五章所提到的算術運算元與邏輯運算元均適用於複合型指定運算。

例一
exp1=exp1opexp2;
x=x+1;
x=x-y;
x1=x1\*y1;
abc=abc&xyz;
r=r>>n;
x=x/(y+1);
.
.
.

例二
exp1op=exp2;
x+=1;
x-=y;
x1\*=y1;
abc&=xyz;
r>>=n;
x/=y+1;
.
.
.
5-5.條件運算子
其語法如下所述，exp1(變數)=exp2(關係運算)?exp3(值或運算):exp4(同exp3)

main()
{
intx=0,y=1;
x=(y>=1)?x+1:x;
printf("%d\n",x);
}

result:
1

6-1.控制結構(controlstructure)
結構化程式設計的基礎建立在下列三個結構：
(1)循序(sequential):以一個接著一個方式進行作業。
(2)條件執行(conditional):如果....則....否則(if\_then\_else)。如果條件符合，就進行"則"(then)之後的敘述。反之，若條件不符合，就進行"否則"(else)之後的敘述。
(3)迴路(looping):重覆執行某一工作，直到滿足某預定條件為止。

6-2.條件執行if的運用
以下是if\_else流程控制可能出現的四種語法並請參考example6-2-1、2。

syntax1：

if(關係運算元)

statement;

syntax2：

if(關係運算元)
{
statement1;
statement2;
.
.
statementn;
}

ps.如果statement超過一個以上，則須以{}將其區塊分隔開來。

syntax3：

if(關係運算元)
statement;
else
statement;

syntax4：

if(關係運算元)
{
statement1;
statement2;
.
statementn;
}
else
{
statement1;
statement2;
.
statementn;
}

example6-2-1：

main()
{
intx;
scanf("%d",&x);
if(x>=10)
printf("%d>=10\n",x);
else
printf("%d<10\n",x);
}

result：
10----->此處為鍵盤輸入值
10>=10

example6-2-2：

main()
{
charc;
scanf("%c",&c);
if(c=='A')
{
printf("c=%c\n",c);
printf("Thisisture.\n");
}
else
{
printf("c!=A\n");
printf("Thatisn'tture.\n");
}

result：
A----->此處為鍵盤輸入值
c=A
Thisisture.

6-2-1.邏輯運算元&&與||
在此補充兩個邏輯運算元&&與||。此兩者是用於連接多個關係運算元，其用法我們以實際的範例加以說明，請參考example6-2-3。

example6-2-3：

main()
{
intx=1,y=2;
if(x==1&&y==3)
printf("1thstep.\n");
else
if(x==1||y==3)
printf("2thstep.\n");
}

result：
2thstep.

6-3.迴路while的運用
在while的敘述中，當關係運算元之條件為真時，會不斷地重覆執行位於while後所列的敘述，直到條件變為否定才停止。while所控制的敘述，不止限於一行，也可以是數行，但要用大括號將欲執行的敘述括起來。請參考example6-3-1，2，3。

example6-3-1：

syntax1：

while(關係運算元)
statement;

syntax2：

while(關係運算元)
{
statement1;
.
.
statementn;
}

example6-3-2：

main()
{
inti=0;
while(i<5)
printf("%d\n",i++);
printf("We'reoutoftheloop.\n");
}

result：
0
1
2
3
4
We'reoutoftheloop.

example6-3-3：

main()
{
inti=0,c=0;
while(i<5)
{
printf("%d",i++);
printf("%d\n",++c);
}
printf("We'reoutoftheloop.\n");
}

result：
01
12
23
34
45
We'reoutoftheloop.

6-3-1.迴路do-while的運用
我們已經知道while迴路是先檢查關係運算元條件是否成立，再決定要不要執行後面的敘述；而do-while迴路，由於測試條件在迴路的後面，所以迴路中的敘述至少會被執行一次，如果條件成立，則敘述會再被執行。反之，則會跳離迴路。請參考example6-3-4，5。

example6-3-4：

syntax1：

do
statement;
while(關係運算元);

syntax2：

do
{
statement1;
statement2;
.
.
statementn;
}while(關係運算元);

example6-3-5：

main()
{
inti=0,c=0;
do
{
printf("%d",i++);
printf("%d\n",++c);
}while(i<5);
printf("We'reoutoftheloop.\n");
}

result：
01
12
23
45
We'reoutoftheloop.

6-4.迴路for的運用
for允許使用者在同一行敘述中就設定了迴路的三個部份。(1)計數器的初值，(2)關係運算元，(3)增減迴路計數器的值。其語法如下所示，並請參考example6-4-1，2。
syntax1
for(ii=1; i<=10; i++)

for(ii=1; i<=10; i=i+1)

for(計數器初值，關係運算元，計數器值更新)
statement;

syntax2

for(計數器初值，關係運算元，計數器值更新)
{
statement1;
.
.
statementn;
}

example6-4-1：

main()
{
inti;
for(i=0;i<=5;++i)
printf("%d\n",i);
}

result：
0
1
2
3
4
5

example6-4-2：

main()
{
inti,c;
for(i=0,c=1;i<5;++i,++c)
{
printf("%d",i);
printf("%d\n",c);
}
printf("We'reoutoftheloop.\n");
}

result：
01
12
23
34
45
We'reoutoftheloop.

6-5.條件執行switch的運用
我們在實際的程式寫作時常會遇到多種選擇情況，而使用一連串if-else來表示是常發生的，所以C提供了一項特殊的控制結構，讓我們能夠有效且精簡處理程式。請參考example6-5-1，2，3。

example6-5-1：

switch(變數)
{
case常數1:
statement1;
case常數2:
statement2;
.
.
case常數n:
statementn;
default:
statement;
}

ps.此處常數的資料型態為整數與字元兩種。

example6-5-3：

main()
{
charvar='D';
switch(var)
{
case'A':
printf("var=A\n");
break;
case'B':
printf("var=B\n");
break;
case'C':
printf("var=C\n");
break;
default:
printf("I'mindefault.");
}
}

result：
I'mindefault.

example6-5-2：

main()
{
intvar=3;
switch(var)
{
case1:
printf("var=1\n");
break;
case2:
printf("var=2\n");
break;
case3:
printf("var=3\n");
break;
default:
printf("I'mindefault");
}
}

result：
var=3
6-6.goto的運用
在一個程式中，使用goto敘述可以強制改變程式執行的步驟，但也會因此使程式的結構混亂，所以此敘述應儘量不用。其語法如下所述。請參考example6-6-1。

syntax:
gotolabel;
.
.
.
label:
.
.
.

example6-6-1：

main()
{
inti=1111;
repeat:
printf("%d\n",i);
gotorepeat;
}

result：
1111
1111
1111
.
.
.
6-7.break與continue的運用

example6-7-1：

main()
{
inti=0;
while(i<=5)
{
++i;
if(i==3)
break;
printf("%d\n",i);
}
}

result：
i=1
i=2

example6-7-2：

main()
{
inti=0;
while(i<=5)
{
++i;
if(i==3)
continue;
printf("i=%d\n",i);
}
}

result：
i=1
i=2
i=4
i=5
i=6

ps1.break與continue均適用於while、do-while、for迴路中，用法一致。
ps2.break的敘述只會跳出包含break部份的迴圈，並不會一次就跳到最外層。