

編譯程式設計

期末專題說明 V1.1

May 2004

專題目標

- 實作一個可用的編譯器
- 編譯指定的程式語言並產生C++
- 編譯器與產生的執行檔必須可以在 Linux 或 FreeBSD 作業系統下執行

程式語言

- 命令式語言
- 使用保留字(reserve word)
- 由最簡單到最難共分為三個階段
 - 基本功能, 進階功能, 完整功能
 - 依完成度作為評分依據
- 做不夠嗎?
 - 額外功能

基本語法

`program`
主程式區塊

- 序述以換行符號作為結尾
- 標記(token)間可以有一個或多個空白
 - 空白含, `space`, `\t`, `\r`
 - 換行符號`\n`只能出現在序述的最後面, 但序述和序述間或檔頭檔尾可以有一個或以上的換行符號
- 符號'#'開始到行尾為註解
- 程式由 `program` 宣告開始, 後接一主程式區塊
- 大小寫視為不同 (case sensitive)

程式區塊

- 程式區塊
 - 一行序述; 或
 - 由begin和end所包圍的一行或以上的序述
 - begin和end獨自各佔一行
 - 允許巢狀結構
- 空序述
 - 使用分號 ‘;’ 做為空序述
 - 空序述為一不做任何動作的序述

基本功能 (1/5)

```
var  
    int 變數1, 變數2, ...
```

● 變數宣告

- 接在 `program` 宣告之後，程式區塊之前，可有可無
- 變數宣告區段以 `var` 開頭，其後可以有一行或多行
- 每一行開頭為變數型態，基本功能只需支援 `int`
 - `int` 是32位元有號整數
- 可宣告一個或多個變數，以逗號分開
- 變數名稱最長不會超過 255 個字元
- 變數名稱可以是英文字母大小寫，數字或是底線，但是開頭不可以是數字

基本功能 (2/5)

```
x := 1  
y := 2  
z := (x + y) * (x - y) # z 的值为 -3
```

- 變數值設定
 - 運算元:= 為指定變數
 - 左邊為變數, 右邊為運算式
- 運算式
 - 基本整數四則運算
 - 提供 +, -, *, /, mod, (), -(負號)

基本功能 (3/5)

```
x := read()  
writestr("the answer is ")  
write( 運算式 )  
writeln()
```

- 整數的輸入與輸出
 - 呼叫 **read()** 可以由鍵盤輸入一整數並傳回
 - 呼叫 **writestr(str)** 輸出一字串
 - 字串為一用兩個雙引號所包函的字元，其中不可有換行符號或雙引號
 - 呼叫 **write(x)** 將 **x** 的內容以十進位方式輸出到螢幕上，並在最後附加一空白字元
 - 呼叫 **writeln()** 輸出換行符號到螢幕上

基本功能 (4/5)

`(x >= y) and not (x <= z)`

- 比較運算
 - 提供 `>`, `>=`, `<`, `<=`, `=`
- 布林運算
 - 提供 `and`, `or`, `not`
 - 如果 `and` 左邊運算元為 `false`, 則右邊不去計算
 - 如果 `or` 左邊運算元為 `true`, 則右邊不去計算
- 運算式零值為假，非零值為真

基本功能 (5/5)

```
while ( 運算式 )
    程式區塊

if ( 運算式 ) then
    程式區塊
else
    程式區塊          # 這部份可有可無
                    #
fi
```

- if-then-else
 - else的部份可有可無
- while 迴圈控制
- 兩者皆允許巢狀結構

基本功能的程式範例

```
#  
# 基本功能範例  
#  
program  
var  
    int i, result, x  
begin  
    x := read()  
    result := 0  
    i := 1  
    while(i <= x)  
    begin  
        result := result + i  
        i := i + 1  
    end  
    write(result)  
    writeln()  
    while(1)          # 無窮迴圈  
        ;            #  
end
```

進階功能 (1/2)

```
var  
int 變數[長度]
```

```
length(陣列變數)
```

- 整數陣列

- 宣告長度為一編譯時即可決定之常數運算式, 例如1+1
- 索引值從0開始
- 用方括號定址
- 編譯時與程式執行時都必須能夠偵測陣列索引值是否超過範圍, 並印出錯誤訊息(並停止程式執行)
- 允許陣列拷貝, 複製整個陣列的內容, 如果來源陣列比目的陣列大的話則印出錯誤訊息
- 提供 length() 函式取得陣列的元素個數

進階功能 (2/2)

```
for i = m to n  
  程式區塊
```

```
foreach i in a  
  程式區塊
```

- for迴圈
 - i 是一個宣告過的變數
 - m 和 n 是兩個運算式, n 可以比 m 小, 大, 或相等
- foreach迴圈
 - i 是一個宣告過的變數
 - a 是一個陣列變數
 - 在程式區塊中, i 的值是 a[0], a[1], a[2], ...
 - 改變 i 不會影響到原來陣列中的值

進階功能的程式範例

```
#  
# 進階功能  
#  
program  
var  
    int a[2], i  
begin  
    a[0] := read()  
    a[1] := read()  
    for i = 0 to length(a) - 1  
    begin  
        writestr("a[ ")  
        write(i)  
        writestr("] is ")  
        write(a[i])  
        writeln()  
    end  
end
```

完整功能

```
sub 函式名稱(int 參數1, int 參數2, ...)  
var  
    int 區域變數1, 區域變數2, ...  
程式區塊
```

- 函式宣告

- 宣告在全域變數之後，主程式區塊之前
- 函式名稱和變數名稱可重複
- 不需支援forward reference
- 允許遞迴呼叫
- Call by Value
- 參數只需支援整數型態，不需支援陣列參數

完整功能的程式範例

```
#  
# 完整功能  
#  
program  
  
# 全域變數  
var  
    int x  
  
# 函式宣告  
sub hello()  
begin  
    write(x)  
    writeln()  
end  
  
# 底下是主程式  
begin  
    x := 1234  
    hello()  
end
```

額外功能1

```
var  
  int 變數[]...  
  
setlength(變數, 長度)
```

- 動態陣列

- 宣告時不指定陣列長度

- 預設長度為零

- 若在**foreach**中改變陣列長度，則結果未定義

- 程式中可以用`setlength(x, len)`來指定陣列 `x` 的長度為 `len`

額外功能2

```
var  
int 變數[第一維長度][第二維長度]...
```

- 多維陣列
 - 不限維數的多維陣列
 - 不需支援foreach, length()等功能

額外功能3

```
var  
  int 變數[][] ...  
  
setlength(變數, 第一維長度, 第二維長度, ...)
```

- 動態多維陣列
 - 不限維數的多維陣列
 - 函式 `setlength` 的參數個數視該陣列維數而定

額外功能4

```
var  
  short 變數 ...  
  byte 變數 ...
```

- 其他長度整數資料型態
 - 提供short：16位元有號整數
 - 提供byte：8位元有號整數
 - 整數型態間可以互相轉換，但是會受到表示範圍的影響
 - 當表示位元不足時，取低位元部份

額外功能5

```
sub 函式名稱(int &參數1, ...)
```

- Call by Reference

- 在宣告函式的參數名稱前加上&表示該參數是reference
- Reference的值會被函式所更動

額外功能6

```
type 自定型態名
  int a
  int b
  ...
epyt

var.a := 1
```

- 自訂結構

- 宣告在program之後，全域變數之前
- 型態名可以和變數及函式名重複
- 用 . 來指定結構的成員

額外功能7

```
sub function(int 變數1)  
sub function(int 變數1, int 變數2)
```

- 多型
 - 同名的函式可以有不同的參數
 - 按照呼叫時的參數決定所呼叫的函式

額外功能8

```
try
    程式區塊1
catch(i)
    程式區塊2

throw(-1)
```

- 例外處理

- 使用 `throw` 函式丟出一個例外，例外的資料型態為 `int`
- `i` 是一宣告過的 `int` 變數，存放例外的值
- 丟出的例外會隨著函式呼叫的堆疊往上層跑，直到被 `try catch` 抓到
- 如果在程式區塊1中發生例外，則程式流程跳到區塊2，區塊1中剩下的程式碼不被執行
- 如果區塊1中外發生任何例外，則區塊2不會被執行
- 存取陣列的索引超出範圍時也會產生例外，其例外值為-1
- 除零錯誤也會產生例外，其例外值為-2

附錄 - 保留字

- 保留字不可以用作變數或函式名稱
- `and, begin, byte, catch, char, do, else, end, epyt, fi, for, foreach, if, in, int, mod, not, or, program, short, sub, then, to, try, type, var, while`

附錄 - 運算元結合優先順序

- 優先順序由上到下遞減，結合方向由左至右
 - () []
 - not, -(負號)
 - * / mod
 - + -
 - < <= > >=
 - =
 - and
 - or

感謝

- 感謝蕭皖文同學協助更新 v1.1