

變數生命週期

sprout 2019

聊天室:tlk.io/sprout-clang

看過這樣的code

有什麼問題?

這就是為什麼你需要知道
變數的生命週期!!!

```
#include<iostream>
int main(){
    //.....
    for(int a=0;a<N;a++){
        //.....
    }
    for(int b=0;b<N;b++){
        //.....
    }
    //.....
    for(int y=0;y<N;y++){
        //.....
    }
    for(int z=0;z<N;z++){
        //.....
    }
    //.....
    return 0;
}
```

變數生命週期

- 變數存活(作用)的範圍
- 全域變數
- 區域變數

全域變數(global variable)

- 在所有函式外面(包括main)
- 整個程式都存在
- 容易出bugs，盡量避免使用

區域變數(local variable)

- 在函式內宣告
- 只存在特定的區域
 - **ex : function 、 block**
- 每個函數的區域變數、參數是獨立的，即使有相同的名字也是兩個不同的東西

function

```
#include<iostream>
void printPi(void);
int main(){
    printPi();
    std::cout<<pi<<"\n"; //error
    return 0;
}
void printPi(void){
    double pi=3.14159;
    std::cout<<pi<<"\n";
    return;
}
```

pi 是區域變數，只存在函式printPi中

block

```
#include<iostream>
int main(){
    for(int i=0;i<10;i++){
        std::cout<<i<<' ';
    }
    std::cout<<i+1<<"\n"; //error
    return 0;
}
```

i 是區域變數，只存在for迴圈所形成的block中

例子

分別是哪種變數?作用範圍為何?

- **a** : 全域 ; 整個程式
- **e** : 全域 ; 整個程式
- **z** : 區域 ; foo function
- **n** : 區域 ; main function
- **i** : 區域 ; main function裡的for迴圈
- **q** : 區域 ; foo function

```
#include<iostream>
int a=10;
double e=2.718281;
void foo(int z);
int main(){
    int n;
    std::cin>>n;
    foo(n);
    for(int i=0;i<a;i++){
        std::cout<<e*n<<"\n";
    }
    return 0;
}
void foo(int z){
    int q=1024;
    z=z+a-q;
    std::cout<<"z = "<<z<<"\n";
    return;
}
```


遮蔽效應

- 在不同大小的scope重複命名，且有重疊
- 優先使用最後命名的那個

輸出:

0
1
2

```
#include<iostream>
int n=10;
int main(){
    int n=3;
    for(int i=0;i<n;i++){
        std::cout<<i<<"\n";
    }
    return 0;
}
```

常見錯誤

```
#include<iostream>
int main(){
    for(int i=0;i<10;i++){
        for(int i=0;i<10;i++){
            std::cout<<i<<"*"<<i<<"="<<i*i<<'  ';
        }
        std::cout<<endl;
    }
    return 0;
}
```

```
0*0=0 1*1=1 2*2=4 3*3=9 4*4=16 5*5=25 6*6=36 7*7=49 8*8=64 9*9=81
0*0=0 1*1=1 2*2=4 3*3=9 4*4=16 5*5=25 6*6=36 7*7=49 8*8=64 9*9=81
0*0=0 1*1=1 2*2=4 3*3=9 4*4=16 5*5=25 6*6=36 7*7=49 8*8=64 9*9=81
0*0=0 1*1=1 2*2=4 3*3=9 4*4=16 5*5=25 6*6=36 7*7=49 8*8=64 9*9=81
0*0=0 1*1=1 2*2=4 3*3=9 4*4=16 5*5=25 6*6=36 7*7=49 8*8=64 9*9=81
0*0=0 1*1=1 2*2=4 3*3=9 4*4=16 5*5=25 6*6=36 7*7=49 8*8=64 9*9=81
0*0=0 1*1=1 2*2=4 3*3=9 4*4=16 5*5=25 6*6=36 7*7=49 8*8=64 9*9=81
0*0=0 1*1=1 2*2=4 3*3=9 4*4=16 5*5=25 6*6=36 7*7=49 8*8=64 9*9=81
0*0=0 1*1=1 2*2=4 3*3=9 4*4=16 5*5=25 6*6=36 7*7=49 8*8=64 9*9=81
0*0=0 1*1=1 2*2=4 3*3=9 4*4=16 5*5=25 6*6=36 7*7=49 8*8=64 9*9=81
```

```
-----
Process exited after 0.5197 seconds with return value 0
請按任意鍵繼續 . . .
```

命名空間(namespace)

- 不同的命名空間可能有相同的變數名稱
- [命名空間名稱]::[命名空間成員]
 - 可以存取空間裡面的成員
- using namespace [命名空間名稱]
 - 可以直接存取空間裡面的任何成員
- using [命名空間名稱]::[命名空間成員]
 - 可以直接存取空間裡面的指定成員

```
#include<iostream>
int main(){
    int a;
    std::cin>>a;
    std::cout<<"a = "<<a<<"\n";
    return 0;
}
```

```
#include<iostream>
using std::cin;
int main(){
    int a;
    cin>>a;
    std::cout<<"a = "<<a<<"\n";
    return 0;
}
```

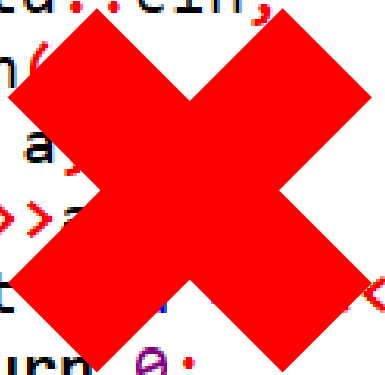
```
#include<iostream>
using std::cin;
int main(){
    int a;
    cin>>a;
    cout<<"a = "<<a<<"\n";
    return 0;
}
```

```
#include<iostream>
using namespace std;
int main(){
    int a;
    cin>>a;
    cout<<"a = "<<a<<"\n";
    return 0;
}
```

```
#include<iostream>
int main(){
    int a;
    std::cin>>a;
    std::cout<<"a = "<<a<<"\n";
    return 0;
}
```

```
#include<iostream>
using std::cin;
int main(){
    int a;
    cin>>a;
    std::cout<<"a = "<<a<<"\n";
    return 0;
}
```

```
#include<iostream>
using std::cin;
int main(){
    int a;
    cin>>a;
    cout<<"a = "<<a<<"\n";
    return 0;
}
```



```
#include<iostream>
using namespace std;
int main(){
    int a;
    cin>>a;
    cout<<"a = "<<a<<"\n";
    return 0;
}
```