C++ 意外處理

```
try {
    ..... // 運算過程的程式碼
    throw 變數; // 一旦程式執行"有錯"
                丟 意外訊息
} catch (意外的型態1) {        程式設計師自訂
    ..... // 型態1之下的錯誤處理
} catch (意外型態2) {
    ..... // 型態2之下的意外錯誤處理
} catch (...) {
    ..... // 其它各式型態的意外處理
}
```

```
int main ()
{ int a, b;
  try {
    for (int i=0; i<3; i++)
      cin >> a >> b;
      if (b==0)
        throw -1;
      cout << a/b
           << endl;
    }
  } catch (int& err_code){
      型態  變數
    cout << err_code <<endl
    cout << "b==0 \n";
  } catch (char* err_string)
      字串型態  變數
    Cout << err_string <<endl
  } catch (...) {
    cout << "error ...\n";
  }
```

修改

throw "b==0 error";
      字串

```
int * p[10];
for (int i=0; i<10; i++) {
    try {
        p[i] = new int [1024000000];
    } catch (bad_alloc &err) {
        cout << i << ": Memory Error \n";
    }   cout << err.what() << endl;
}
```

// 配置記憶体

出錯

new的程式碼会丟出意外訊息

throw 丟拉;

型態

bad_alloc

C++自訂意外型態

一位參考的方式

型態 丟拉

#include <~~except~~>

<std except>