

# 報告

## 委託

	123 stock	123 price	123 qty	123 bscore
1	50	10	1	1
2	6,016	20	5	1
3	2,330	500	3	1
4	1,416	50	5	1
5	1,234	58	3	1
6	939	15	200	2
7	2,435	1,000	10	2
8	1,056	11	50	2
9	7,058	55	10	2
10	2,589	22	66	2

## 股票庫存

	123 stock	123 qty
1	50	0
2	6,016	10
3	2,330	30
4	1,416	4
5	1,234	6
6	939	1,000
7	2,435	1
8	1,056	3
9	7,058	40
10	2,589	77

## HLS

```
#include <ap_int.h>
#include <hls_stream.h>
#include <iostream>
#include <stdio.h>
using namespace hls;

#define stock_len 14
#define qty_len 14
#define price_len 20
#define bs_len 2
#define order_len stock_len + qty_len + price_len + bs_len

typedef ap_uint<order_len> order_t;
```

```

typedef ap_uint<stock_len> stock_t;
typedef ap_uint<qty_len> qty_t;
typedef ap_uint<price_len> price_t;
typedef ap_uint<bs_len> bs_t;

// 預設最大處理數量
const int customers_max = 50;

// 讀取客戶資料
static void read_input(order_t *customer_data, hls::stream<order_t>
&inStream, int number_of_customer);

// 判斷資料
static void compare(int *initdata, hls::stream<order_t> &inStream,
hls::stream<int> &outStream, int number_of_customer);

// 回傳對應的結果
static void write_result(int *customer_ans, hls::stream<int> &outStream,
int number_of_customer);

extern "C"
{
    void riskcontrol(order_t *customer_data, int *customer_ans, int
customers_number);
}

```

```

#include "hls.hpp"
#include <ap_int.h>
#include <hls_stream.h>
using namespace hls;

// 讀取客戶資料
static void read_input(order_t *customer_data, hls::stream<order_t>
&inStream, int number_of_customer)
{
    mem_rd:
        for (int i = 0; i < number_of_customer; i++)
        {
            #pragma HLS LOOP_TRIPCOUNT min = customers_max max = customers_max
            inStream << customer_data[i];
        }
}

// 判斷
static void compare(int *initdata, hls::stream<order_t> &inStream,
hls::stream<int> &outStream, int number_of_customer)
{
    execute:
        int money = 0;
        for (int i = 0; i < number_of_customer; i++)
        {

```

```

#pragma HLS LOOP_TRIPCOUNT min = customers_max max = customers_max
    order_t order = inStream.read();
    stock_t tempstock = order(stock_len + price_len + qty_len + bs_len
- 1, price_len + qty_len + bs_len);
    price_t tempprice = order(price_len + qty_len + bs_len - 1,
price_len);
    qty_t tempqty = order(qty_len + bs_len - 1, qty_len);
    bs_t tempbs = order(bs_len - 1, 0);
    money = tempqty * tempprice;
    if (i == 0)
        printf("dataassign i=%d 資料: [%d] stock: [%d] price: [%d] qty:
[%d] bs: [%d]\n", i, order, tempstock, tempprice, tempqty, tempbs);

    if (tempbs == 1)
    { // 買單檢查額度
        if (5000000 < money)
            outStream.write(1);
        else
            outStream.write(0);
    }
    else
    { // 賣單檢查庫存
        if (initdata[i] < tempqty)
            outStream.write(1);
        else
            outStream.write(0);
    }
}

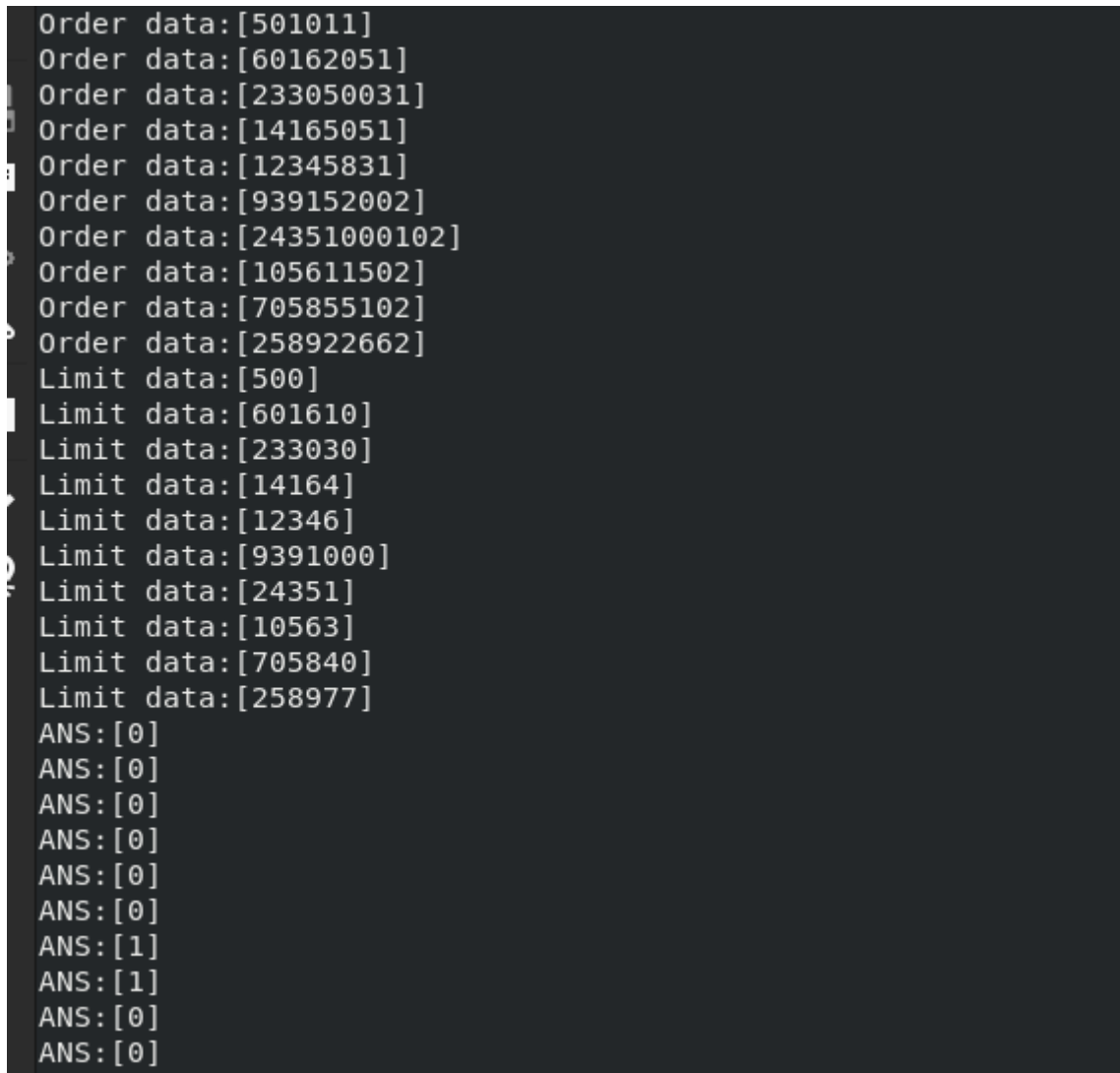
// 回傳對應的結果
static void write_result(int *customer_ans, hls::stream<int> &outStream,
int number_of_customer)
{
    mem_wr:
        for (int i = 0; i < number_of_customer; i++)
        {
#pragma HLS LOOP_TRIPCOUNT min = customers_max max = customers_max
            customer_ans[i] = outStream.read();
        }
}

extern "C"
{
    void riskcontrol(order_t *customer_data, int *customer_ans, int
customers_number)
    {
#pragma HLS INTERFACE m_axi port = customer_data bundle = gmem0 depth = 32
#pragma HLS INTERFACE m_axi port = customer_ans bundle = gmem0 depth = 32
        static hls::stream<order_t> customers_in("input_stream");
        static hls::stream<int> customers_out("output_stream");
        int initdata[] = {0, 10, 30, 4, 6, 1000, 1, 3, 40, 77};
#pragma HLS STREAM variable = customers_in depth = 32
#pragma HLS STREAM variable = customers_out depth = 32

```

```
#pragma HLS dataflow
    read_input(customer_data, customers_in, customers_number);
    compare(initdata, customers_in, customers_out, customers_number);
    write_result(customer_ans, customers_out, customers_number);
}
```

## 圖片



```
Order data:[501011]
Order data:[60162051]
Order data:[233050031]
Order data:[14165051]
Order data:[12345831]
Order data:[939152002]
Order data:[24351000102]
Order data:[105611502]
Order data:[705855102]
Order data:[258922662]
Limit data:[500]
Limit data:[601610]
Limit data:[233030]
Limit data:[14164]
Limit data:[12346]
Limit data:[9391000]
Limit data:[24351]
Limit data:[10563]
Limit data:[705840]
Limit data:[258977]
ANS:[0]
ANS:[0]
ANS:[0]
ANS:[0]
ANS:[0]
ANS:[0]
ANS:[0]
ANS:[1]
ANS:[1]
ANS:[0]
ANS:[0]
```