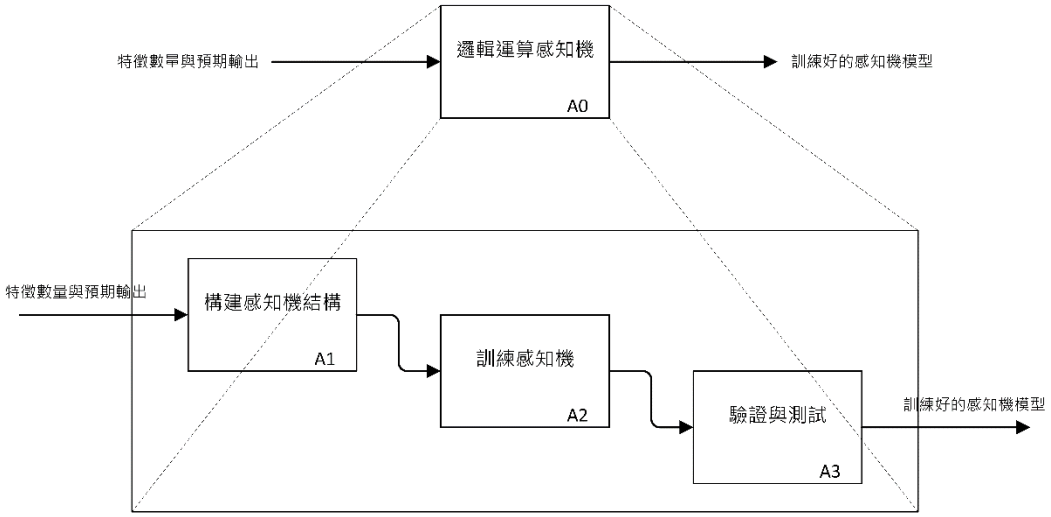
嵌入式系統設計

期中作業報告

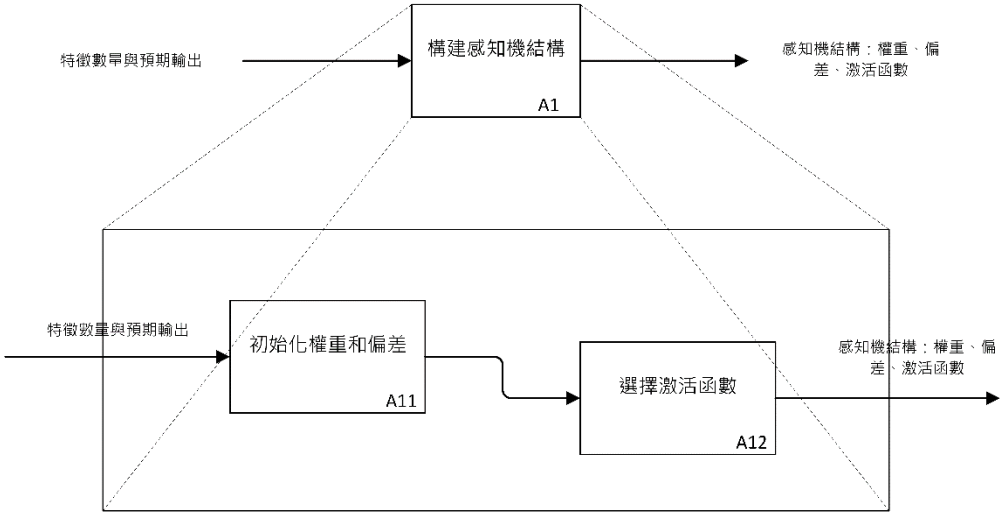
112522083 資工系 鄧祺文

IDEF0 設計感知機系統的階層式模組化架構

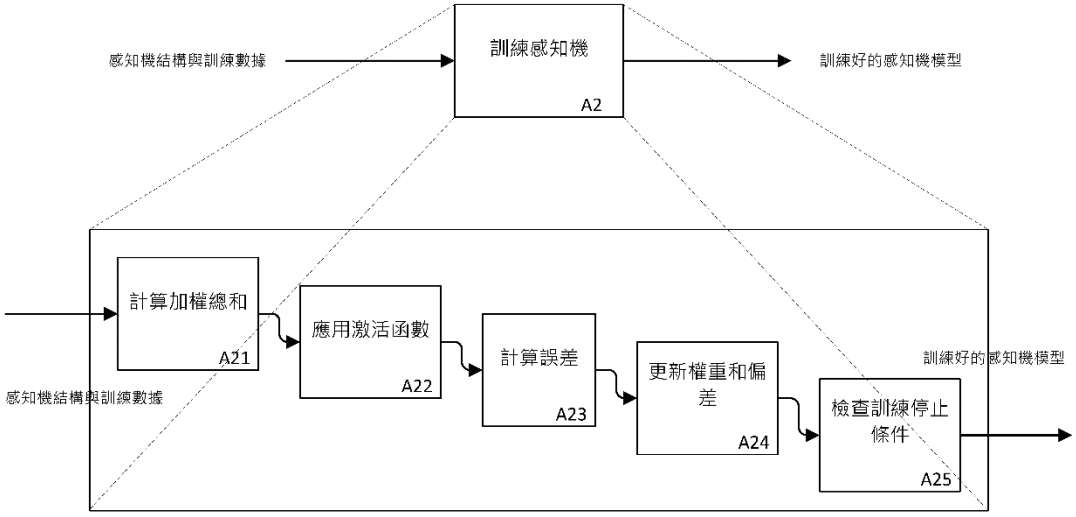
基本設計：一共包含三大部分，分別是「構建感知機結構」、「訓練感知機」和「驗證與測試」（輸入：特徵數量與預期輸出；輸出：訓練好的感知機模型）



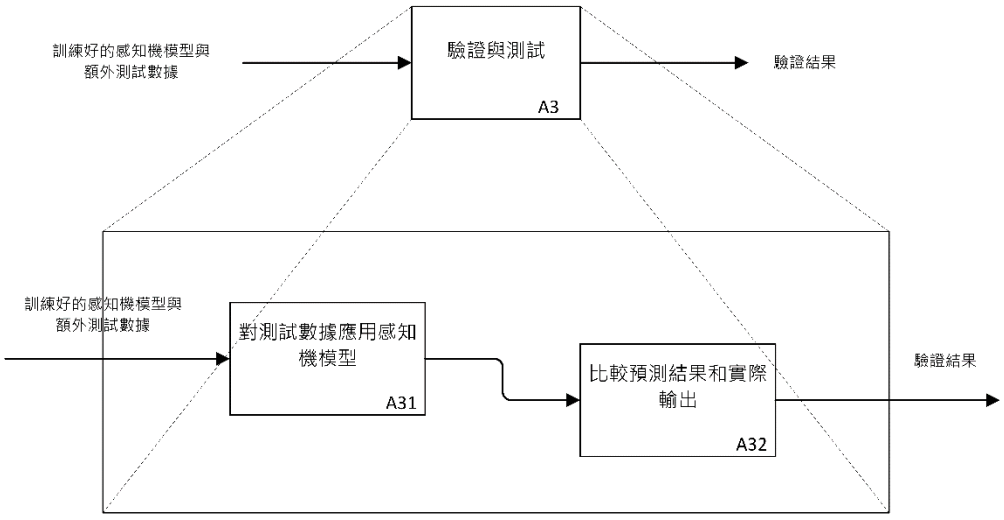
細部設計之一：構建感知機結構（輸入：特徵數量與預期輸出；輸出：感知機結構，包含權重、偏差和激活函數）



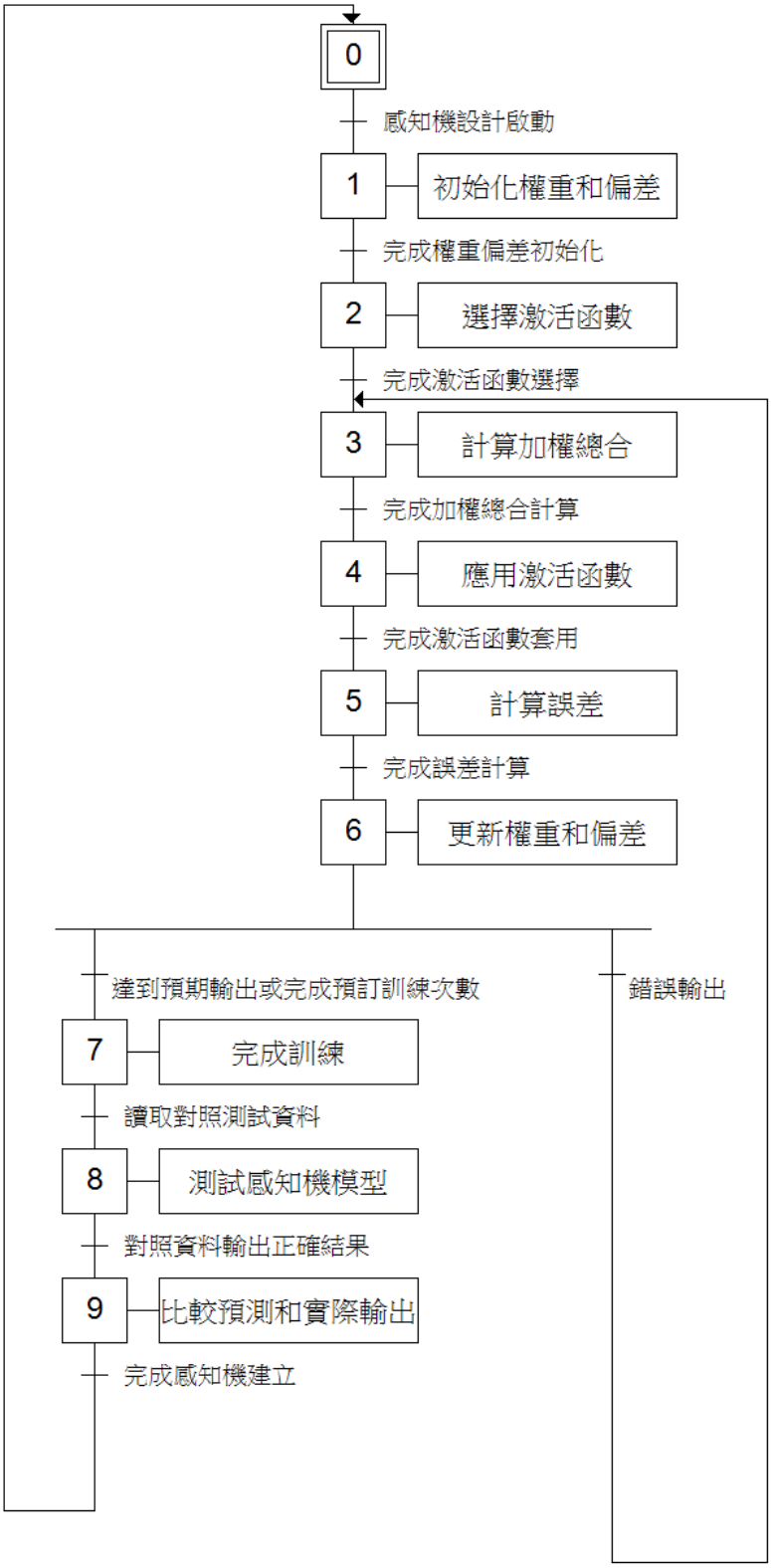
細部設計之二：訓練感知機（輸入：感知機結構與訓練數據；輸出：訓練好的感知機模型）



細部設計之三：驗證與測試（輸入：訓練好的感知機模型與額外測試數據；輸出：驗證結果）

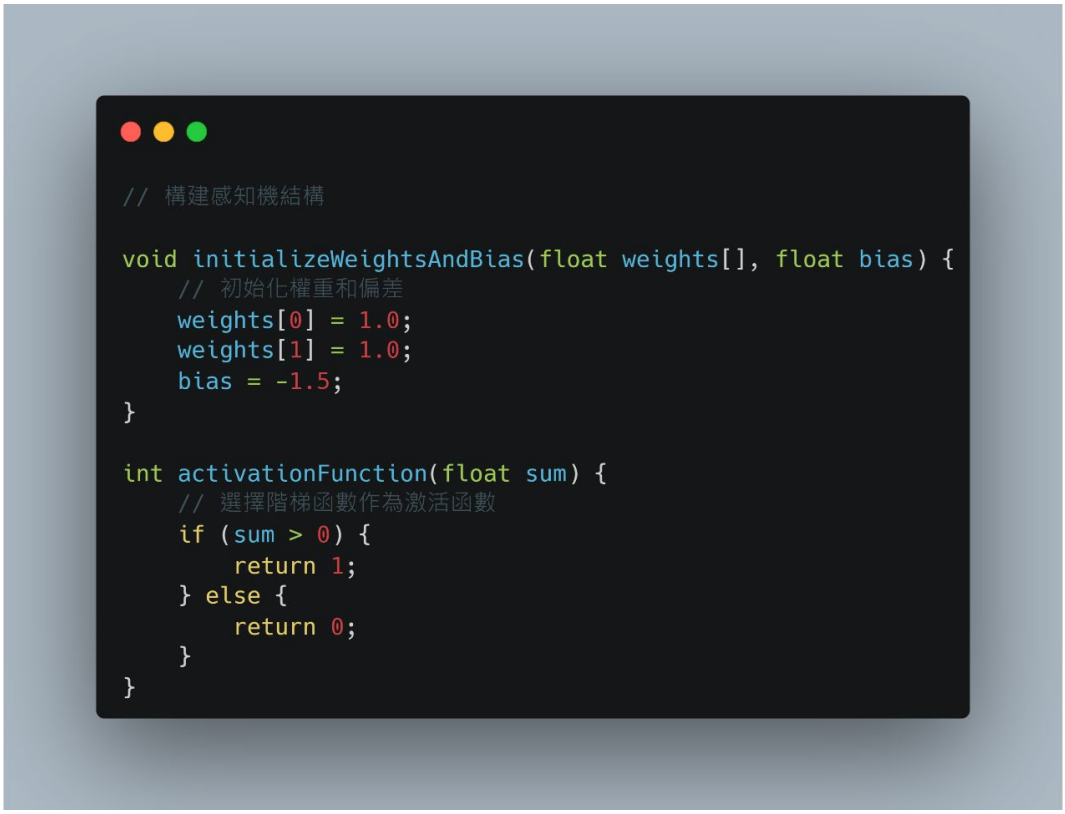


Grafcet 設計離散事件模型



以 MIAT 方法論合成 C Code

✓ C Code：構建感知機結構

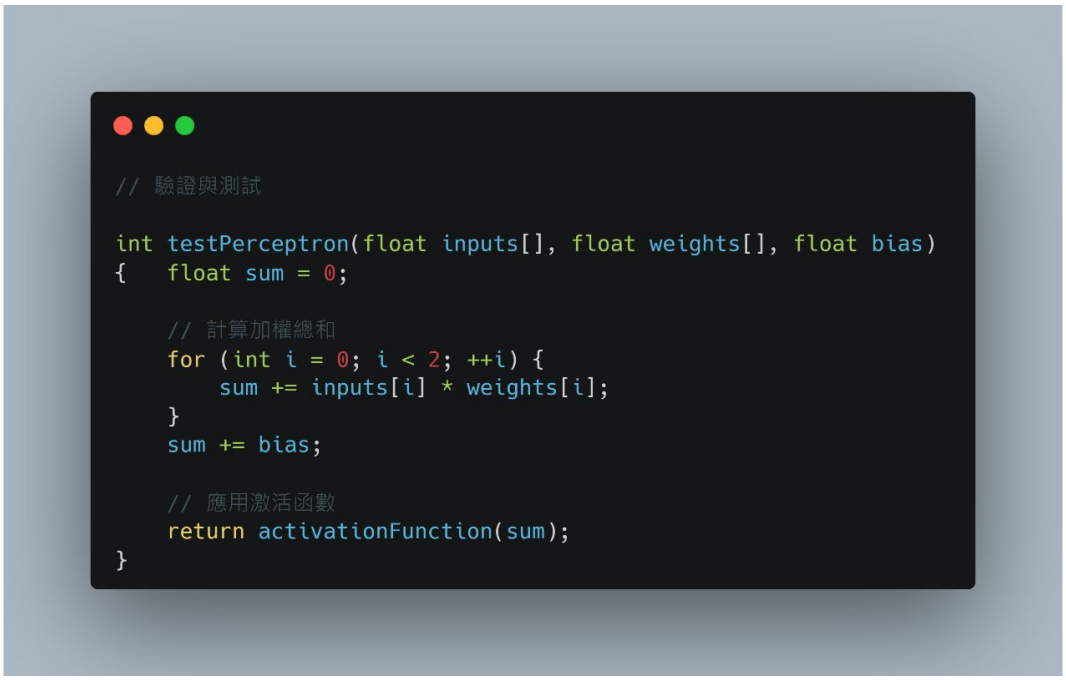


```c
// 構建感知機結構

void initializeWeightsAndBias(float weights[], float bias) {
    // 初始化權重和偏差
    weights[0] = 1.0;
    weights[1] = 1.0;
    bias = -1.5;
}

int activationFunction(float sum) {
    // 選擇階梯函數作為激活函數
    if (sum > 0) {
        return 1;
    } else {
        return 0;
    }
}
```

✓ C Code：訓練感知機

```c
// 訓練感知機

void trainPerceptron(float inputs[], float weights[], float bias, int expectedOutput) {
    float sum = 0;

    // 計算加權總和
    for (int i = 0; i < 2; ++i) {
        sum += inputs[i] * weights[i];
    }

    sum += bias;

    // 應用激活函數
    int output = activationFunction(sum);

    // 計算誤差並更新權重和偏差
    float error = expectedOutput - output;

    for (int i = 0; i < 2; ++i) {
        weights[i] += error * inputs[i];
    }

    bias += error;
}
```

✓ C Code：驗證與測試



```
// 驗證與測試

int testPerceptron(float inputs[], float weights[], float bias)
{   float sum = 0;

    // 計算加權總和
    for (int i = 0; i < 2; ++i) {
        sum += inputs[i] * weights[i];
    }
    sum += bias;

    // 應用激活函數
    return activationFunction(sum);
}
```

結合底層 API 呼叫完成軟體驗證

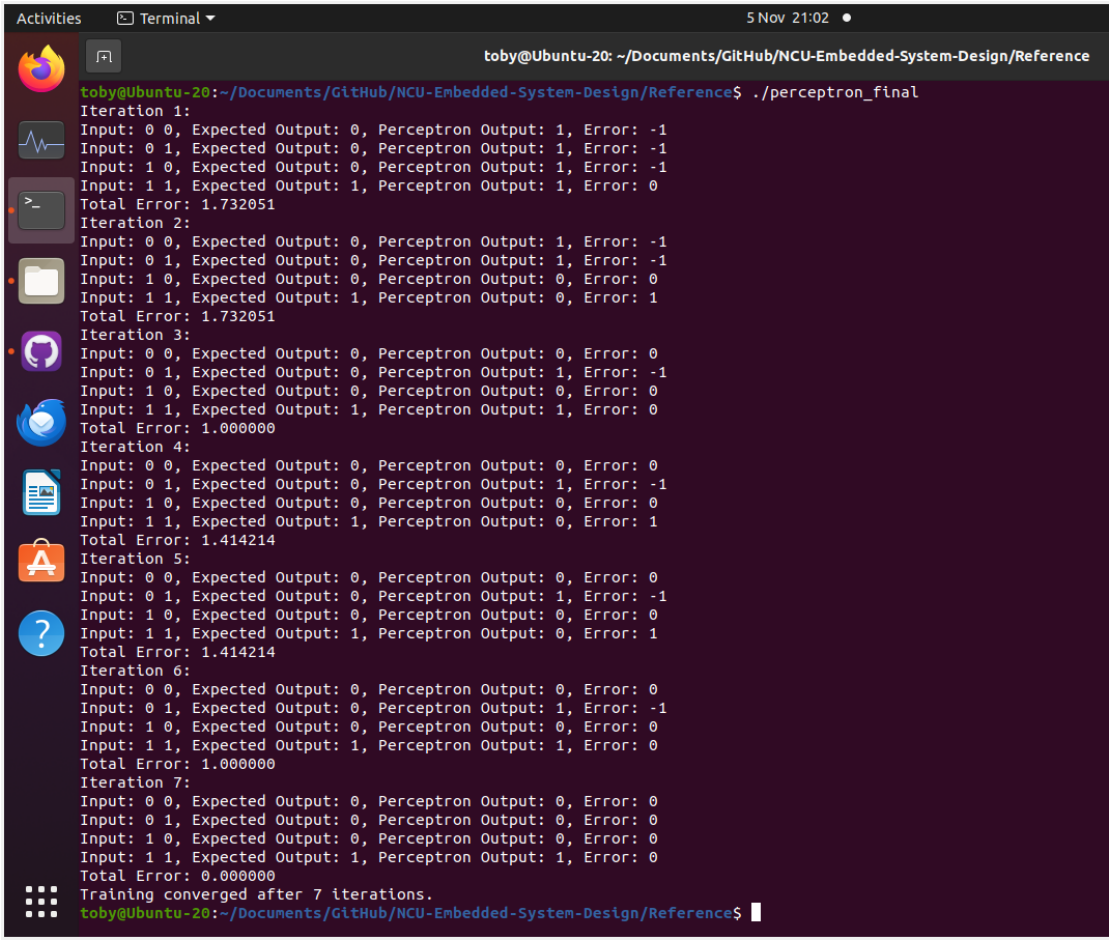
```c
#include <stdio.h>
#include <math.h>

// 構建感知機結構
void initializeWeightsAndBias(float weights[], float bias) {
    weights[0] = 1.0;
    weights[1] = 1.0;
    bias = -1.5;
}

int activationFunction(float sum) {
    if (sum > 0) {
        return 1;
    } else {
        return 0;
    }
}

// 訓練感知機
void trainPerceptron(float inputs[], float weights[], float bias, int expectedOutput) {
    float sum = 0;

    for (int i = 0; i < 2; ++i) {
        sum += inputs[i] * weights[i];
    }

    sum += bias;

    int output = activationFunction(sum);

    float error = expectedOutput - output;

    for (int i = 0; i < 2; ++i) {
        weights[i] += error * inputs[i];
    }

    bias += error;
}

// 驗證與測試
int testPerceptron(float inputs[], float weights[], float bias) {
    float sum = 0;

    for (int i = 0; i < 2; ++i) {
        sum += inputs[i] * weights[i];
    }

    sum += bias;

    return activationFunction(sum);
}

int main() {
    float weights[2];
    float bias;
    float inputs[4][2] = {{0, 0}, {0, 1}, {1, 0}, {1, 1}};
    int expectedOutputs[4] = {0, 0, 0, 1};

    initializeWeightsAndBias(weights, bias);

    for (int i = 0; i < 4; ++i) {
        trainPerceptron(inputs[i], weights, bias, expectedOutputs[i]);
    }

    printf("AND Logic Test:\n");

    for (int i = 0; i < 4; ++i) {
        int output = testPerceptron(inputs[i], weights, bias);
        printf("Input: %d, %d, Output: %d\n", (int)inputs[i][0], (int)inputs[i][1],
output);
    }

    return 0
}
```

基於已完成之軟體驗證進行實驗

✓ 基於實驗要求進行微調

```c
#include <stdio.h>
#include <math.h>

typedef struct {
    float weights[2];
    float bias;
} Perceptron;

void initializePerceptron(Perceptron* perceptron) {
    perceptron->weights[0] = 0;
    perceptron->weights[1] = 0.4;
    perceptron->bias = 0.3;
}

int activationFunction(float sum) {
    return (sum > 0) ? 1 : 0;
}

void trainPerceptron(Perceptron* perceptron, float inputs[][2], int expectedOutputs[], int numIterations)
{
    for (int iteration = 1; iteration <= numIterations; ++iteration) {
        printf("Iteration %d:\n", iteration);
        float totalError = 0;
        for (int i = 0; i < 4; ++i) {
            float sum = inputs[i][0] * perceptron->weights[0] + inputs[i][1] * perceptron->weights[1] +
perceptron->bias;
            int output = activationFunction(sum);
            int error = expectedOutputs[i] - output;
            totalError += pow(error, 2);

            // 更新權重和偏差
            perceptron->weights[0] += 0.1 * error * inputs[i][0];
            perceptron->weights[1] += 0.1 * error * inputs[i][1];
            perceptron->bias += 0.1 * error;

            printf("Input: %d %d, Expected Output: %d, Perceptron Output: %d, Error: %d\n",
                   (int)inputs[i][0], (int)inputs[i][1], expectedOutputs[i], output, error);
        }
        totalError = sqrt(totalError);
        printf("Total Error: %f\n", totalError);
        if (totalError == 0) {
            printf("Training converged after %d iterations.\n", iteration);
            break;
        }
    }
}

int main() {
    Perceptron perceptron;
    float inputs[4][2] = {{0, 0}, {0, 1}, {1, 0}, {1, 1}};
    int expectedOutputs[4] = {0, 0, 0, 1};

    initializePerceptron(&perceptron);

    trainPerceptron(&perceptron, inputs, expectedOutputs, 50);

    return 0;
}
```

✓ 輸出感知機邏輯訓練結果

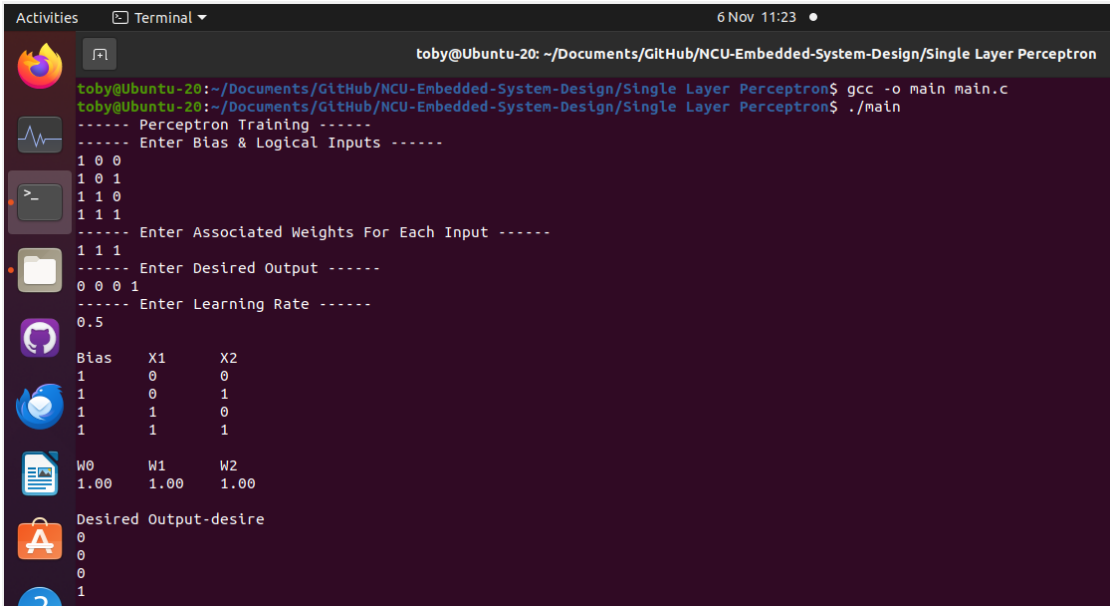


額外補充：通用邏輯運算之設計

✓ 基於通用設計進行調整

```c
#include <stdio.h>

int main() {
    int i, j;
    int x[4][3];
    int actual[4];
    int desire[4];
    int error;
    int epoch = 0;
    float net = 0.0;
    float learning_rate;
    float weight[3];

    printf("------ Perceptron Training ------\n");
    printf("------ Enter Bias & Logical Inputs ------\n");
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 3; j++) {
            scanf("%d", &x[i][j]);
        }
    }
    printf("------ Enter Associated Weights For Each Input ------\n");
    for (i = 0; i < 3; i++) {
        scanf("%f", &weight[i]);
    }
    printf("------ Enter Desired Output ------\n");
    for (i = 0; i < 4; i++) {
        scanf("%d", &desire[i]);
    }
    printf("------ Enter Learning Rate ------\n");
    scanf("%f", &learning_rate);
    printf("\n");
    printf("Bias\tX1\tX2\n");
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 3; j++) {
            printf("%d \t", x[i][j]);
        }

        printf("\n");
    }
    printf("\n");
    printf("W0\tW1\tW2\n");
    for (i = 0; i < 3; i++) {
        printf("%.2f \t", weight[i]);
    }
    printf("\n");
    printf("\n");
    printf("Desired Output-desire\n");
    for (i = 0; i < 4; i++) {
        printf("%d \n", desire[i]);
    }
    printf("\n");
    printf("Learning Rate: %.2f\n", learning_rate);
    printf("W0\tW1\tW2\tNet Output\tactual\tdesire\n");
    do {
        for (i = 0; i < 4; i++) {
            for (j = 0; j < 3; j++) {
                net = net + (weight[j] * x[i][j]);
            }
            if (net >= 0) {
                actual[i] = 1;
            } else {
                actual[i] = 0;
            }
            error = desire[i] - actual[i];
            for (j = 0; j < 3; j++) {
                weight[j] = weight[j] + (learning_rate * error * x[i][j]);
                printf("%.2f \t", weight[j]);
            }
            printf("%.2f \t\t %d \t %d \n", net, actual[i], desire[i]);
        }
        epoch++;
    } while (actual[0] != desire[0] || actual[1] != desire[1] || actual[2] != desire[2] || actual[3] !=
desire[3]);
    printf("\nFor Learning Rate: %.2f, Number Of Epochs: %d\n", learning_rate, epoch);
    return 0;
}
```

✓ 輸入所需的邏輯運算（下圖以 AND 為例）



✓ 輸出感知機邏輯訓練結果