Course: C Programming

- **Textbook**
  - H. Deitel and P. Deitel, C How to Program, 9th edition, Prentice-Hall, 2023.
- **References & Resources**
  - An introduction to the C programming language.
  - VS Code: https://code.visualstudio.com/
  - Code::Block: https://www.codeblocks.org/
  - Microsoft Visual Studio: https://visualstudio.microsoft.com/
  - A collection of problems. Uva Online Judge
  - CPE: http://cpe.cse.nsysu.edu.tw/
  - E-tutor: http://e-tutor.itsa.org.tw/e-Tutor/
- **Evaluation**
  - Grade = round(quiz * 0.2 + homework * 0.2 + midterm * 0.3 + final * 0.3 + bonus);
    - Quiz: 20% (**There is an online quiz from 9:20 to 9:40 every week. The quiz consists of questions regarding the C programming language and C program tracing.**)
    - Homework: 20% (**There are 2 to 4 assignments every week.**)
    - Midterm: 30%
    - Final: 30%
    - Bonus (Advanced exercises)

Special rule: This course encourages you to join the CPE test.

```
if(CPE>=2) {
  Grade = max(Grade, ((midterm>0) &&(final>0) &&(quiz>0) &&(homework>0)) ? ((final>=50) ? 85+(CPE-2)*4: 75):0));
} else if (CPE==1) {
  Grade = max(Grade, ((midterm>0)&&(final>0) &&(quiz>0) &&(homework>0)) ? ((final>=50) ? 75: 60):0));
}
```

Textbook web page: https://deitel.com/c-how-to-program-9-e/
Source code: https://github.com/pdeitel/CHowToProgram9e

C How to Program, Ninth Edition
with Case Studies Introducing Applications Programming and Systems Programming
by Paul Deitel & Harvey Deitel

PART 1 (Introductory) Programming Fundamentals Quickstart

1. Introduction to Computers and C
Intro to Hardware, Software & Internet; Test-Drive Microsoft Visual Studio, Apple Xcode, GNU gcc & GNU gcc in Docker

2. Intro to C Programming
Input, Output, Types, Arithmetic, Decision Making, Secure C

3. Structured Program Development
Algorithm Development, Problem Solving, `if`, `if/else`, `while`, Secure C

4. Program Control
`for`, `do/while`, `switch`, `break`, `continue`, Logical Operators, Secure C

5. Functions
Custom Functions, Simulation, Random-Number Generation, Enumerations, Function Call and Return Mechanism, Recursion, Recursive Factorial, Recursive Fibonacci, Secure C

- C is one of the world's most popular and senior programming languages
- C18/C11 standards
- Topical, innovative presentation
- Rich coverage of fundamentals
- Problem-solving/developing algorithms
- 20+ fun computer-science, data-science and artificial-intelligence case studies show C as it's intended to be used—some are fully implemented, some are partially implemented and some require students to do online research
- 147 complete working programs
- 350+ integrated self-check exercises with answers
- 445 end-of-chapter exercises/projects
- Use with Windows®, macOS®, Linux®
- Visual C++®, Xcode® and GNU™ gcc

PART 2 (Intermediate) Arrays, Pointers and Strings

6. Arrays
One- & Two-Dimensional Arrays, Passing Arrays to Functions, Searching, Binary Search Visualization, Sorting, Secure C

7. Pointers
Pointer operators & and *, Pass-By-Value vs. Pass-By-Reference, Array and Pointer Relationship, Secure C

8. Characters and Strings
C Standard Library String- and Character-Processing Functions, Secure C

PART 3 (Intermediate) Formatted Input/Output, Structs and File Processing

9. Formatted Input/Output
`scanf` and `printf` formatting, Secure C

10. Structures, Unions, Bit Manipulation and Enumerations
Creating Custom Types with `structs` and `unions`, Bitwise Operators, Enumeration Constants, Secure C

11. File Processing
Streams, Text and Binary Files, CSV Files, Sequential and Random-Access Files, Secure C

- Analysis of algorithms with Big O
- Enhanced security and data science coverage as per ACM/IEEE 2020 curricula recommendations
- Use free open-source libraries and tools
- Real-world examples and data
- Traditional or "flipped" classrooms
- Secure C Programming, privacy, ethics
- Case studies in systems programming and applications programming
- Think like a developer with GitHub®, open-source, StackOverflow and more

PART 4 (Advanced) Data Structures and Algorithms

12. Data Structures
Dynamic Memory Allocation, Lists, Stacks, Queues & Binary Trees, Secure C

13. Computer-Science Thinking: Sorting Algorithms and Big O
Insertion Sort, Selection Sort, Visualizing Merge Sort, Additional Algorithms including Quicksort in the Exercises

PART 5 (Advanced) Preprocessor and Other Topics

14. Preprocessor
`#include`, Conditional Compilation, Macros/Arguments, Assertions, Secure C

15. Other Topics
Variable-Length Argument Lists, Command-Line Arguments, Multiple-Source-File Programs, `extern`, `exit`/`atexit`, `calloc`/`realloc`, `goto`, Numeric Literal Suffixes, Signal Handling

Appendices

A. Operator Precedence
B. ASCII Character Set
C. Multithreading/Multicore and Other C11/C18 Topics
D. Intro to Object-Oriented Programming

Online Appendices

E. Number Systems
F–H. Using the Visual Studio, GNU gdb and Xcode Debuggers

- Emphasis on visualization
- Static code analysis tools
- Performance, multithreading, multicore
- Questions? `deitel@deitel.com`
- Updates and errata: `https://deitel.com/chtp9`

Systems Programming Case Studies

Systems Software
- Building Your Own Computer
- Building Your Own Compiler with Infix and Postfix Notation

Embedded Systems Programming
- Webots 3D Robotics Simulator

Performance: Threading/Multicore

Applications Programming Case Studies

Algorithm Development
- Counter-Controlled Iteration
- Sentinel-Controlled Iteration
- Nested Control Statements

Random-Number Simulation
- Building a Casino Game
- Card Shuffling/Dealing with Card Images
- The Tortoise and the Hare Race

Intro to Data Science
- Data Analysis: Mean, Median & Mode

Direct-Access File Processing
- Transaction-Processing System

Visualizing Searching & Sorting

Artificial Intelligence/Data Science
- Machine Learning, GNU Scientific Library, Plotting with gnuplot, CSV Files
- NLP: Who Wrote Shakespeare's Works?

Game Programming with raylib
- SpotOn and Cannon Games

Security Via Cryptography
- Secret-Key & RSA Public-Key Crypto

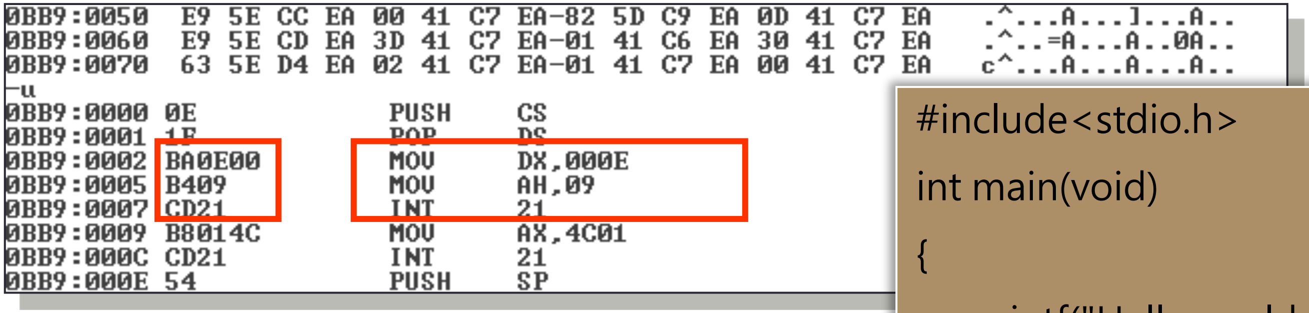
Visualization with raylib
- Law of Large Numbers Animation

Multimedia: Audio & Animation

Web Services, Mashups, Cloud
- Accessing Web Services with libcurl; OpenWeatherMap JSON Results
- Rapid Applications Development with Web-Service Mashups

機器語言、組合語言、高階語言

- 機器語言(Machine languages)
  - 數字代碼指令
- 組合語言(Assembly languages)
  - 符號指令
- 高階語言(High-level languages)
  - 近似英文



```
0BB9:0050  E9 5E CC EA 00 41 C7 EA-82 5D C9 EA 0D 41 C7 EA   .^...A...]...A..
0BB9:0060  E9 5E CD EA 3D 41 C7 EA-01 41 C6 EA 30 41 C7 EA   .^..=A...A..0A..
0BB9:0070  63 5E D4 EA 02 41 C7 EA-01 41 C7 EA 00 41 C7 EA   c^...A...A...A..
-u
0BB9:0000 0E            PUSH    CS
0BB9:0001 1F            POP     DS
0BB9:0002 BA0E00        MOV     DX,000E
0BB9:0005 B409          MOV     AH,09
0BB9:0007 CD21          INT     21
0BB9:0009 B8014C        MOV     AX,4C01
0BB9:000C CD21          INT     21
0BB9:000E 54            PUSH    SP
```

機器語言　　組合語言

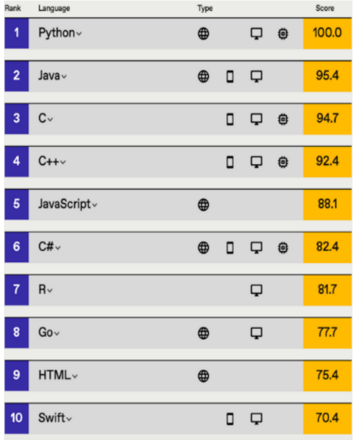
```
#include<stdio.h>
int main(void)
{
    printf("Hello world !\n");
    return 0;
}
```

C程式語言

其他高階語言

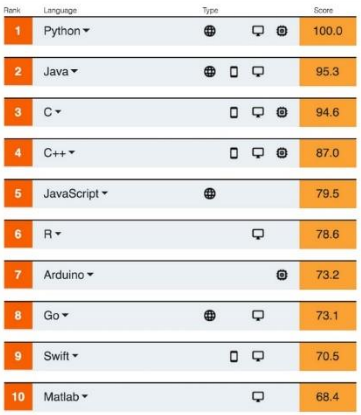
- C++
- Java
- C#
- Javascript
- Python

C-based programming languages



Rank	Language	Type	Score
1	Python	🌐 🖥 ▣	100.0
2	Java	🌐 📱 🖥	95.4
3	C	📱 🖥 ▣	94.7
4	C++	📱 🖥 ▣	92.4
5	JavaScript	🌐	88.1
6	C#	🌐 📱 🖥 ▣	82.4
7	R	🖥	81.7
8	Go	🌐 🖥	77.7
9	HTML	🌐	75.4
10	Swift	📱 🖥	70.4

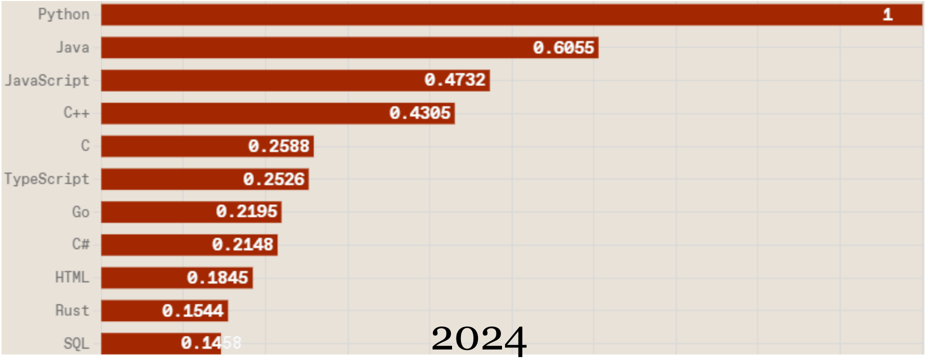
2021



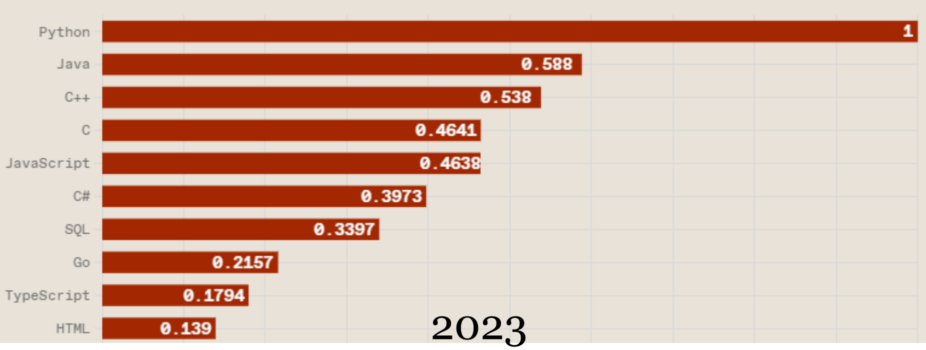
Rank	Language	Type	Score
1	Python	🌐 🖥 ▣	100.0
2	Java	🌐 📱 🖥	95.3
3	C	📱 🖥 ▣	94.6
4	C++	📱 🖥 ▣	87.0
5	JavaScript	🌐	79.5
6	R	🖥	78.6
7	Arduino	▣	73.2
8	Go	🌐 🖥	73.1
9	Swift	📱 🖥	70.5
10	Matlab	🖥	68.4

2020

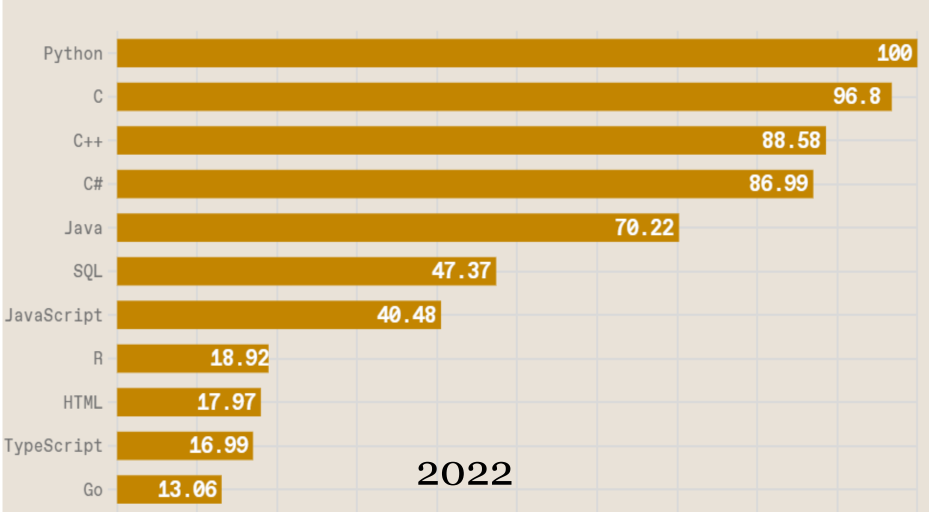
IEEE Spectrum's Top Ten Languages



2024

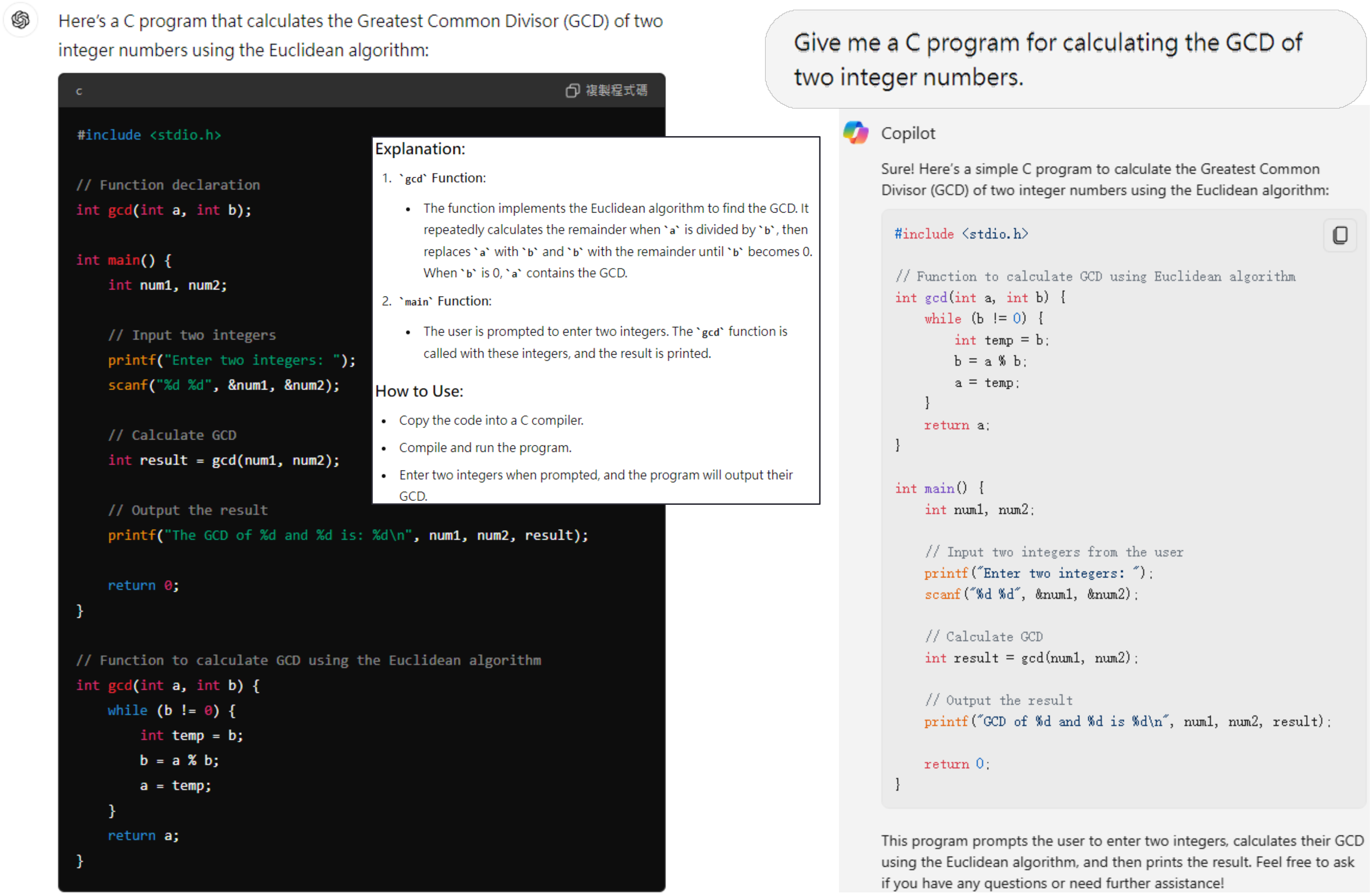


2023



2022

AI-based Code Generation



Here's a C program that calculates the Greatest Common Divisor (GCD) of two integer numbers using the Euclidean algorithm:

```c
#include <stdio.h>

// Function declaration
int gcd(int a, int b);

int main() {
    int num1, num2;

    // Input two integers
    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);

    // Calculate GCD
    int result = gcd(num1, num2);

    // Output the result
    printf("The GCD of %d and %d is: %d\n", num1, num2, result);

    return 0;
}

// Function to calculate GCD using the Euclidean algorithm
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
```

Explanation:

1. `gcd` Function:
    - The function implements the Euclidean algorithm to find the GCD. It repeatedly calculates the remainder when `a` is divided by `b`, then replaces `a` with `b` and `b` with the remainder until `b` becomes 0. When `b` is 0, `a` contains the GCD.
2. `main` Function:
    - The user is prompted to enter two integers. The `gcd` function is called with these integers, and the result is printed.

How to Use:

- Copy the code into a C compiler.
- Compile and run the program.
- Enter two integers when prompted, and the program will output their GCD.

Give me a C program for calculating the GCD of two integer numbers.

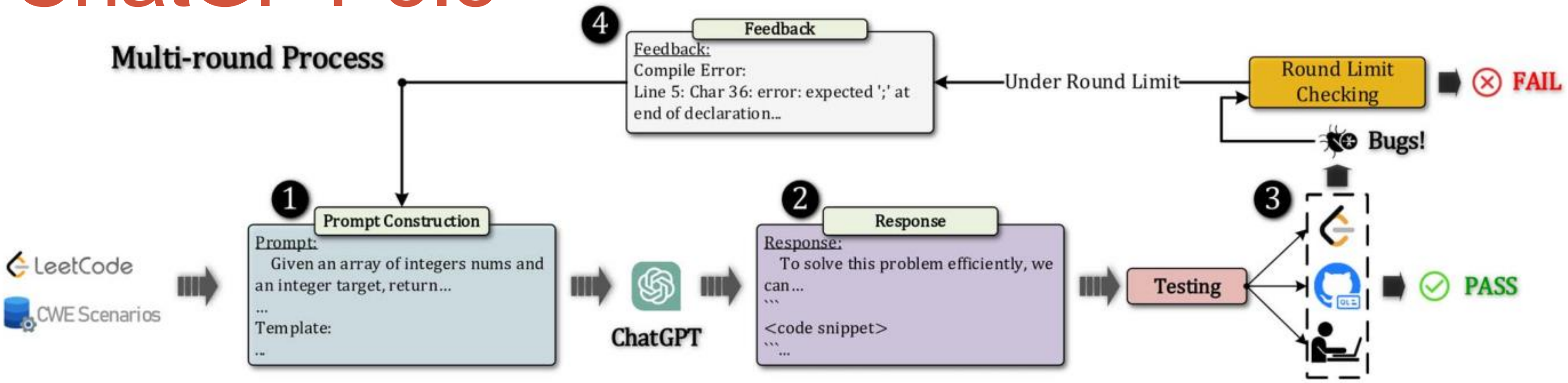
Copilot

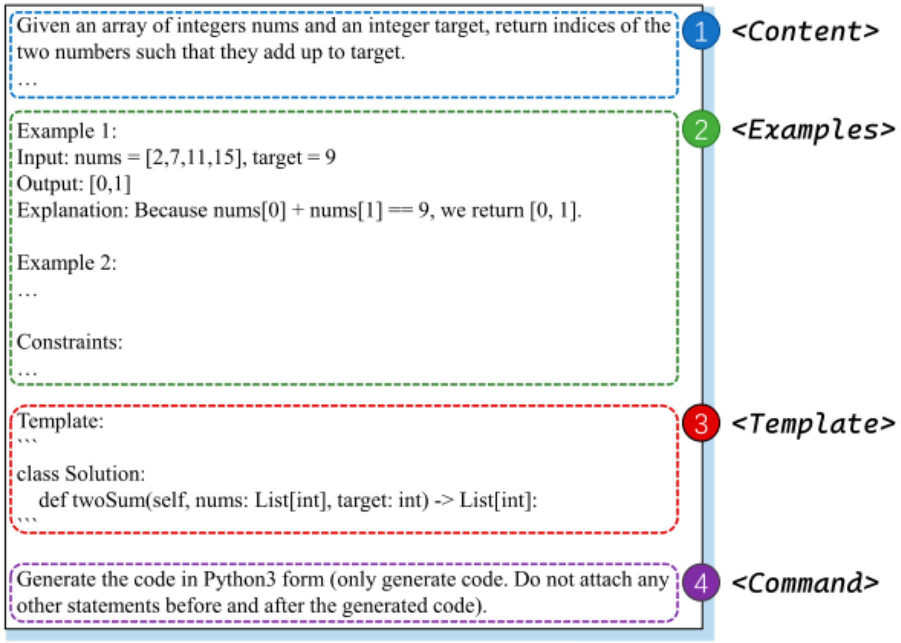
Sure! Here's a simple C program to calculate the Greatest Common Divisor (GCD) of two integer numbers using the Euclidean algorithm:

```c
#include <stdio.h>

// Function to calculate GCD using Euclidean algorithm
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

int main() {
    int num1, num2;

    // Input two integers from the user
    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);

    // Calculate GCD
    int result = gcd(num1, num2);

    // Output the result
    printf("GCD of %d and %d is %d\n", num1, num2, result);

    return 0;
}
```

This program prompts the user to enter two integers, calculates their GCD using the Euclidean algorithm, and then prints the result. Feel free to ask if you have any questions or need further assistance!

The Quality of Code Generated by ChatGPT 3.5




Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.
…

Example 1:
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

Example 2:
…

Constraints:
…

Template:
```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
~~~

Generate the code in Python3 form (only generate code. Do not attach any other statements before and after the generated code).

Z. Liu, Y. Tang, X. Luo, Y. Zhou and L. F. Zhang, "No Need to Lift a Finger Anymore? Assessing the Quality of Code Generation by ChatGPT," in IEEE Transactions on Software Engineering, vol. 50, no. 6, pp. 1548-1584, June 2024, doi: 10.1109/TSE.2024.3392499.
```

A. Accepted
W.A. Wrong Answer
C.E. Compile Error
T.L.E. Time-Limit Exceeded
R.E. Runtime Error

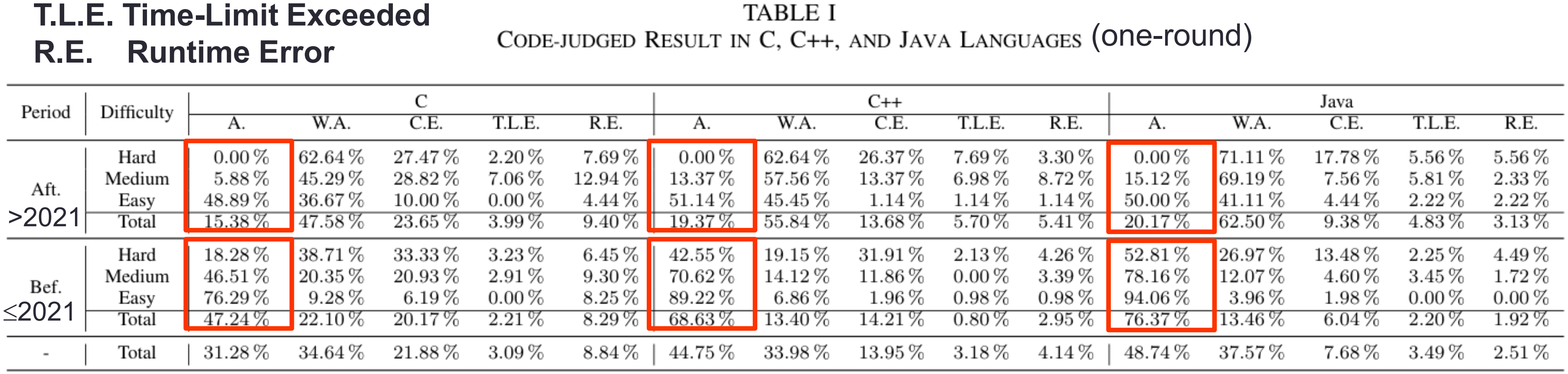


TABLE I
CODE-JUDGED RESULT IN C, C++, AND JAVA LANGUAGES (one-round)

Period	Difficulty	C					C++					Java				
		A.	W.A.	C.E.	T.L.E.	R.E.	A.	W.A.	C.E.	T.L.E.	R.E.	A.	W.A.	C.E.	T.L.E.	R.E.
Aft. >2021	Hard	0.00 %	62.64 %	27.47 %	2.20 %	7.69 %	0.00 %	62.64 %	26.37 %	7.69 %	3.30 %	0.00 %	71.11 %	17.78 %	5.56 %	5.56 %
	Medium	5.88 %	45.29 %	28.82 %	7.06 %	12.94 %	13.37 %	57.56 %	13.37 %	6.98 %	8.72 %	15.12 %	69.19 %	7.56 %	5.81 %	2.33 %
	Easy	48.89 %	36.67 %	10.00 %	0.00 %	4.44 %	51.14 %	45.45 %	1.14 %	1.14 %	1.14 %	50.00 %	41.11 %	4.44 %	2.22 %	2.22 %
	Total	15.38 %	47.58 %	23.65 %	3.99 %	9.40 %	19.37 %	55.84 %	13.68 %	5.70 %	5.41 %	20.17 %	62.50 %	9.38 %	4.83 %	3.13 %
Bef. ≤2021	Hard	18.28 %	38.71 %	33.33 %	3.23 %	6.45 %	42.55 %	19.15 %	31.91 %	2.13 %	4.26 %	52.81 %	26.97 %	13.48 %	2.25 %	4.49 %
	Medium	46.51 %	20.35 %	20.93 %	2.91 %	9.30 %	70.62 %	14.12 %	11.86 %	0.00 %	3.39 %	78.16 %	12.07 %	4.60 %	3.45 %	1.72 %
	Easy	76.29 %	9.28 %	6.19 %	0.00 %	8.25 %	89.22 %	6.86 %	1.96 %	0.98 %	0.98 %	94.06 %	3.96 %	1.98 %	0.00 %	0.00 %
	Total	47.24 %	22.10 %	20.17 %	2.21 %	8.29 %	68.63 %	13.40 %	14.21 %	0.80 %	2.95 %	76.37 %	13.46 %	6.04 %	2.20 %	1.92 %
-	Total	31.28 %	34.64 %	21.88 %	3.09 %	8.84 %	44.75 %	33.98 %	13.95 %	3.18 %	4.14 %	48.74 %	37.57 %	7.68 %	3.49 %	2.51 %

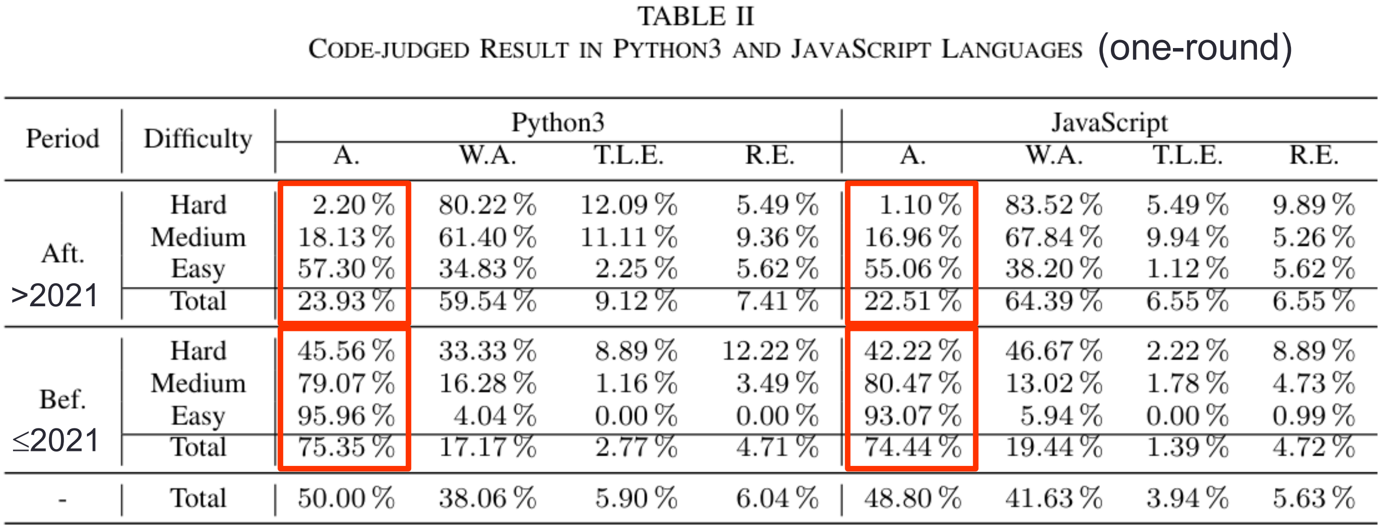


TABLE II
CODE-JUDGED RESULT IN PYTHON3 AND JAVASCRIPT LANGUAGES (one-round)

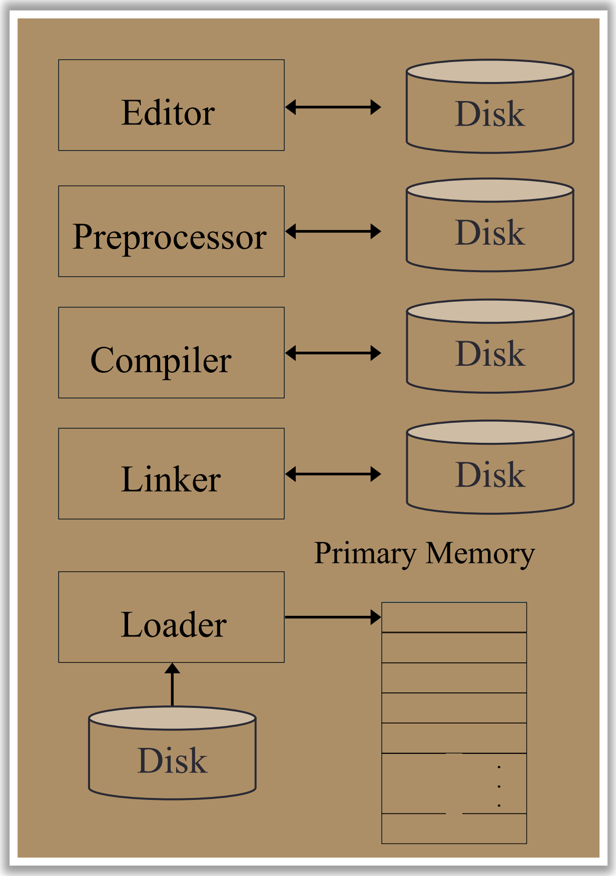
Period	Difficulty	Python3				JavaScript			
		A.	W.A.	T.L.E.	R.E.	A.	W.A.	T.L.E.	R.E.
Aft. >2021	Hard	2.20 %	80.22 %	12.09 %	5.49 %	1.10 %	83.52 %	5.49 %	9.89 %
	Medium	18.13 %	61.40 %	11.11 %	9.36 %	16.96 %	67.84 %	9.94 %	5.26 %
	Easy	57.30 %	34.83 %	2.25 %	5.62 %	55.06 %	38.20 %	1.12 %	5.62 %
	Total	23.93 %	59.54 %	9.12 %	7.41 %	22.51 %	64.39 %	6.55 %	6.55 %
Bef. ≤2021	Hard	45.56 %	33.33 %	8.89 %	12.22 %	42.22 %	46.67 %	2.22 %	8.89 %
	Medium	79.07 %	16.28 %	1.16 %	3.49 %	80.47 %	13.02 %	1.78 %	4.73 %
	Easy	95.96 %	4.04 %	0.00 %	0.00 %	93.07 %	5.94 %	0.00 %	0.99 %
	Total	75.35 %	17.17 %	2.77 %	4.71 %	74.44 %	19.44 %	1.39 %	4.72 %
-	Total	50.00 %	38.06 %	5.90 %	6.04 %	48.80 %	41.63 %	3.94 %	5.63 %

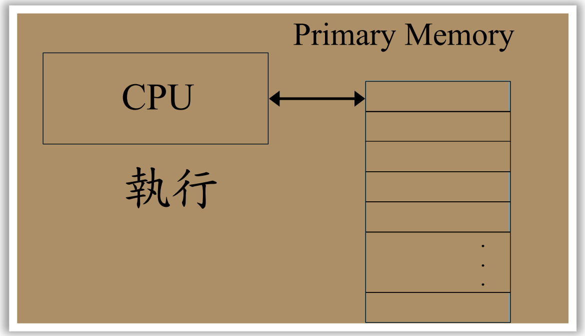
CHAPTER 2 – INTRODUCTION TO C PROGRAMMING

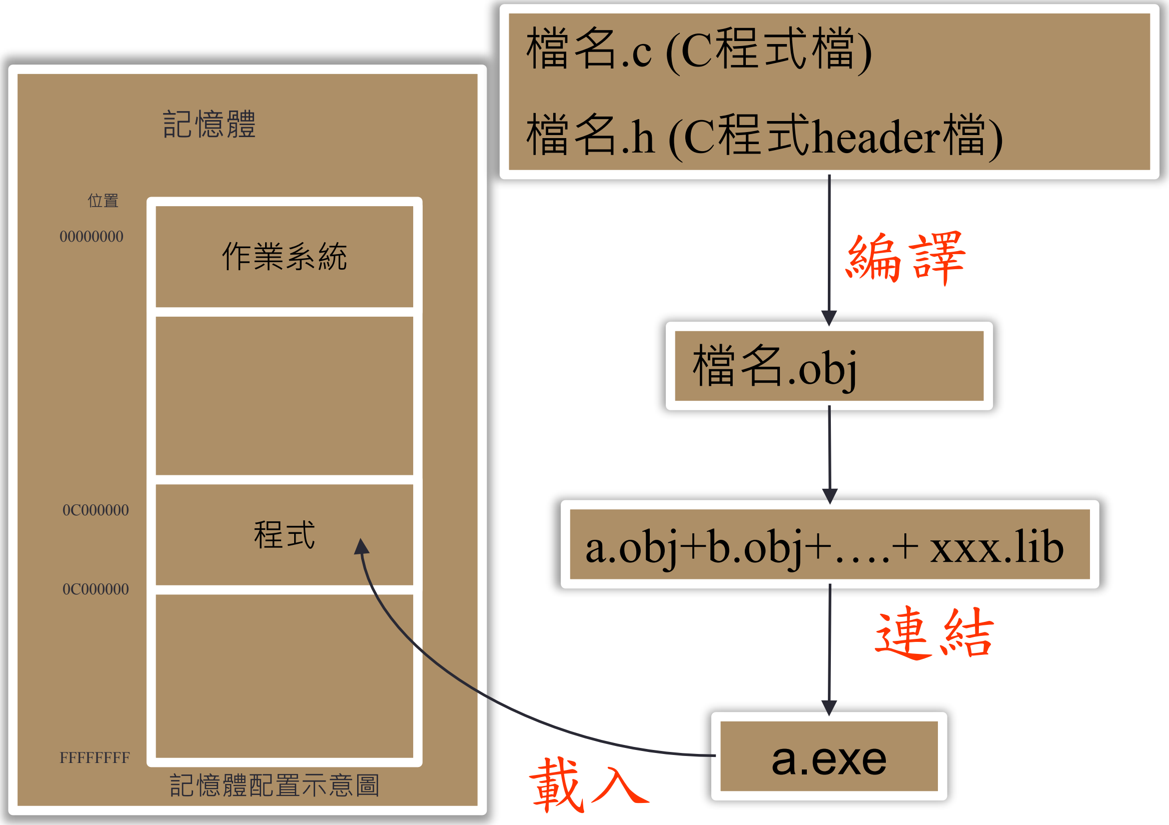
撰寫C程式

- 處理C程式的階段

1. 編輯C程式(*Edit*)
2. 前處理(*Preprocess*)
   #include
   #define
   #ifdef
   …
3. 編譯(*Compile*)
4. 連結(*Link*)
5. 載入(*Load*)
6. 執行(*Execute*)







記憶體

位置

00000000

作業系統

0C000000

程式

0C000000

FFFFFFFF

記憶體配置示意圖

檔名.c (C程式檔)

檔名.h (C程式header檔)

編譯

檔名.obj

a.obj+b.obj+….+ xxx.lib

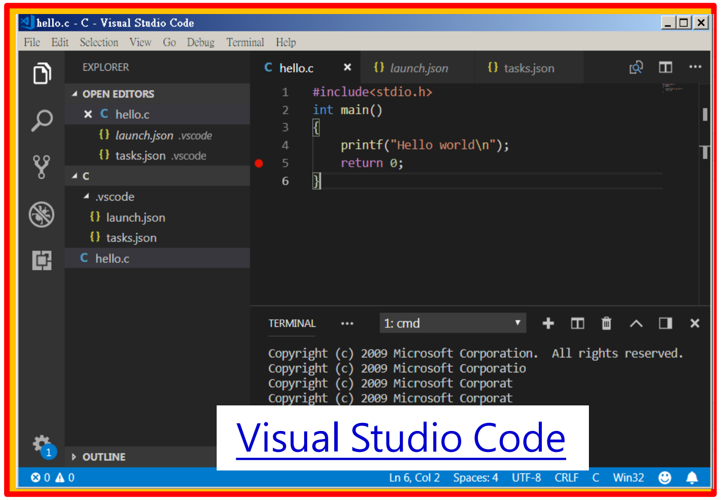
連結

a.exe

載入

整合開發環境 (IDE)

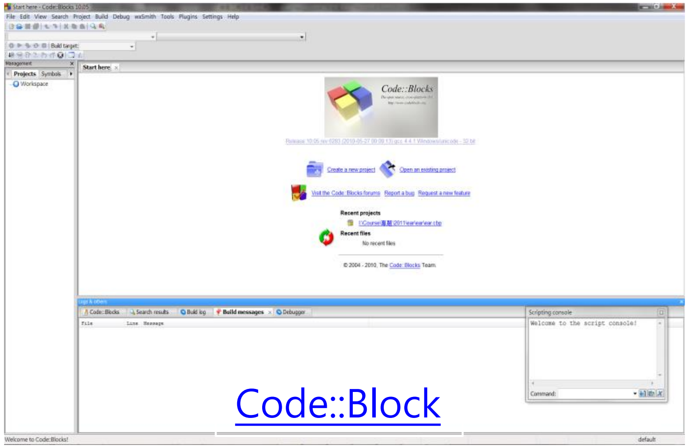
整合開發環境: 整合編輯，編譯，連結，除錯，測試，效能評估於一體。



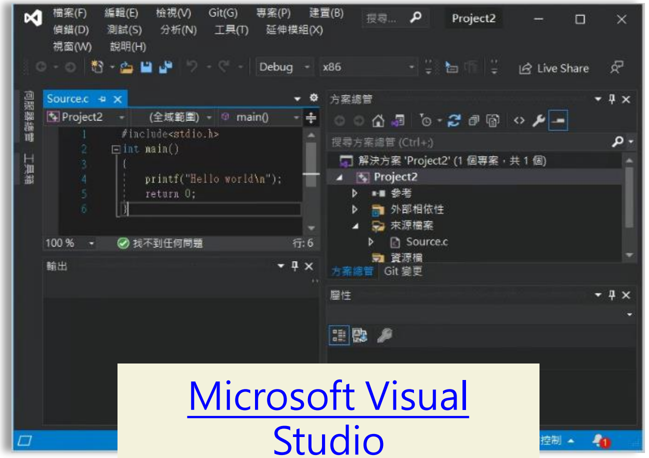
```c
#include<stdio.h>
int main()
{
    printf("Hello world\n");
    return 0;
}
```

```
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.
Copyright (c) 2009 Microsoft Corporatio
Copyright (c) 2009 Microsoft Corporat
Copyright (c) 2009 Microsoft Corporat
```

Visual Studio Code



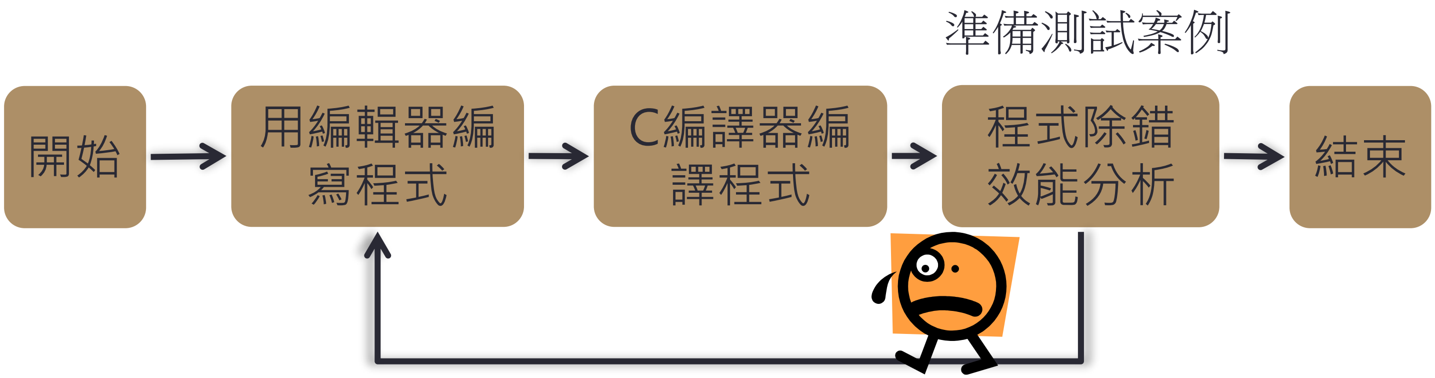
Code::Block



```c
#include<stdio.h>
int main()
{
    printf("Hello world\n");
    return 0;
}
```

Microsoft Visual Studio

C程式撰寫流程



準備測試案例

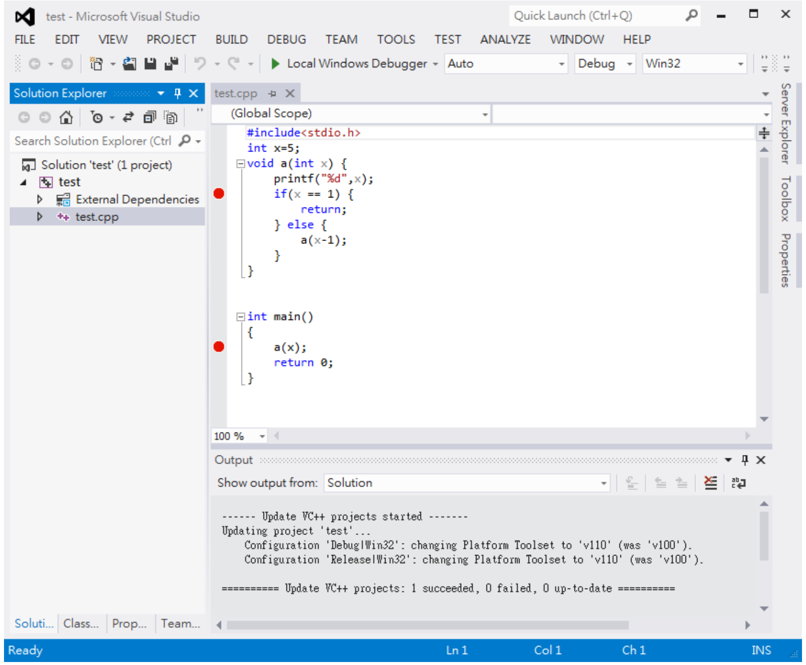
開始

用編輯器編寫程式

C編譯器編譯程式

程式除錯效能分析

結束



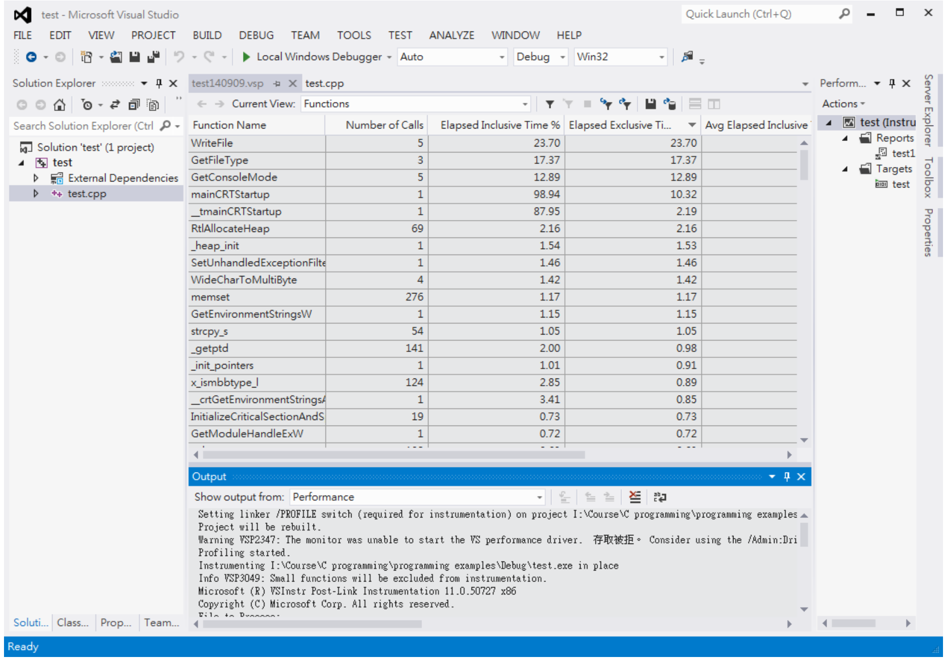
test - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Local Windows Debugger
Auto
Debug
Win32
Solution Explorer
Search Solution Explorer (Ctrl
Solution 'test' (1 project)
test
External Dependencies
test.cpp
(Global Scope)
Server Explorer
Toolbox
Properties
100 %
Soluti... Class... Prop... Team...
Ready
Ln 1
Col 1
Ch 1
INS

```
#include<stdio.h>
int x=5;
void a(int x) {
    printf("%d",x);
    if(x == 1) {
        return;
    } else {
        a(x-1);
    }
}


int main()
{
    a(x);
    return 0;
}
```

Output

Show output from: Solution

```
------ Update VC++ projects started -------
Updating project 'test'...
    Configuration 'Debug|Win32': changing Platform Toolset to 'v110' (was 'v100').
    Configuration 'Release|Win32': changing Platform Toolset to 'v110' (was 'v100').

========== Update VC++ projects: 1 succeeded, 0 failed, 0 up-to-date ==========
```



Current View: Functions

Function Name	Number of Calls	Elapsed Inclusive Time %	Elapsed Exclusive Ti...	Avg Elapsed Inclusive
WriteFile	5	23.70	23.70	
GetFileType	3	17.37	17.37	
GetConsoleMode	5	12.89	12.89	
mainCRTStartup	1	98.94	10.32	
__tmainCRTStartup	1	87.95	2.19	
RtlAllocateHeap	69	2.16	2.16	
_heap_init	1	1.54	1.53	
SetUnhandledExceptionFilte	1	1.46	1.46	
WideCharToMultiByte	4	1.42	1.42	
memset	276	1.17	1.17	
GetEnvironmentStringsW	1	1.15	1.15	
strcpy_s	54	1.05	1.05	
_getptd	141	2.00	0.98	
_init_pointers	1	1.01	0.91	
x_ismbbtype_l	124	2.85	0.89	
__crtGetEnvironmentStrings/	1	3.41	0.85	
InitializeCriticalSectionAndS	19	0.73	0.73	
GetModuleHandleExW	1	0.72	0.72	

Output

Show output from: Performance

```
Setting linker /PROFILE switch (required for instrumentation) on project I:\Course\C programming\programming examples
Project will be rebuilt.
Warning VSP2347: The monitor was unable to start the VS performance driver.  存取被拒。 Consider using the /Admin:Dri
Profiling started.
Instrumenting I:\Course\C programming\programming examples\Debug\test.exe in place
Info VSP3049: Small functions will be excluded from instrumentation.
Microsoft (R) VSInstr Post-Link Instrumentation 11.0.50727 x86
Copyright (C) Microsoft Corp. All rights reserved.
```

C程式設計學習要點

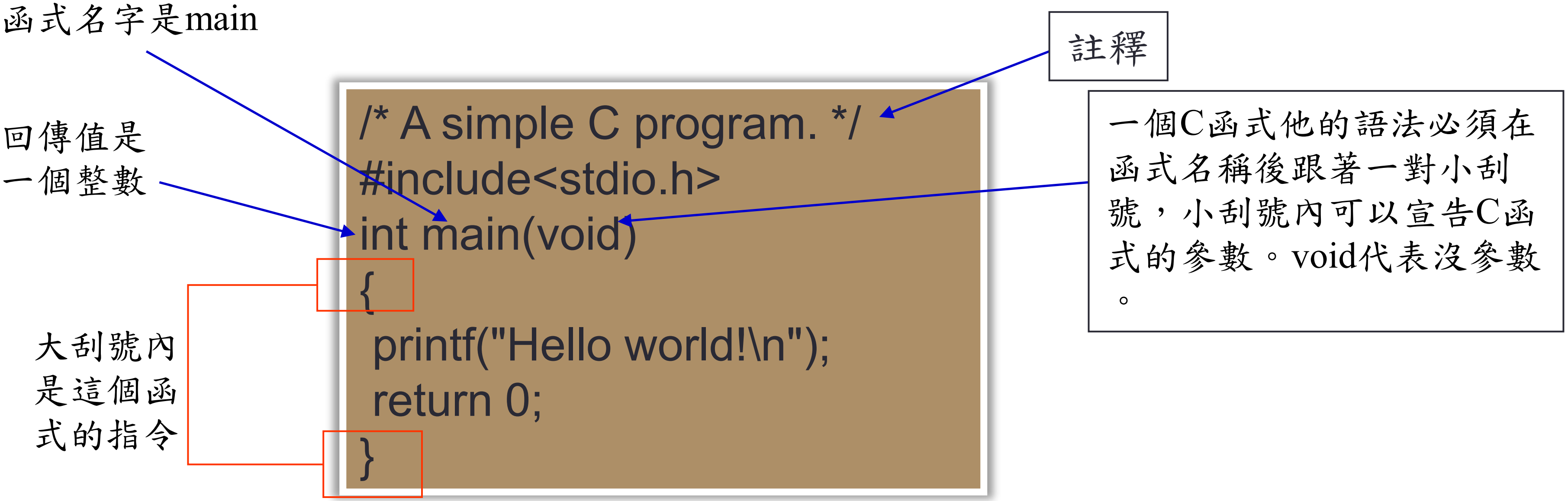
- C程式語言語法
- C程式設計概念: C程式皆是用下面結構組合而成。
  - 循序
  - 選擇
  - 反覆(迴圈)
  - 遞迴
- C程式追蹤
  - 程式閱讀
  - 演算法理解
- C程式測試與除錯
  - 除錯器使用
  - 設計測試案例
- 使用AI-based程式產生工具
  - 程式的組裝
  - 程式的檢查

C程式語言學習重點

- 資料型態
  - 基本資料型態: 如 char;int;unsigned;long long;float;double;long double;
  - 陣列(array)
  - 指標(pointer)
  - 自訂資料型態:struct;union;enum;typedef
  - 資料型態間的轉換
- 變數宣告 (全域與區域變數的差異與用法)
  - 變數可用範圍(scope)
  - 變數有效時間(duration)
- 控制結構
  - 循序結構
  - 選擇結構: if; if else; switch; ?:;
  - 反覆結構:for-loop; while-loop; do-while-loop
  - 遞迴結構
- 函式
  - 自訂函式
  - 包含遞迴函式的寫法。
  - 標準程式庫裡的函式，主要有
    - 標準輸入輸出 (stdio.h):printf,scanf,getchar,putchar,gets等。
    - 數學函式 (math.h): sin,cos,tan,pow,exp,log等。
    - 字串處理函式 (string.h): strcpy,strcat,strcmp,strtok,atol,atof,sprintf,sscanf等。
    - 有關記憶體操作的函式 (stdlib.h,string.h): memmov,memcpy,memcmp,memset,malloc,free等。
    - 檔案處理函式 (stdio.h): fopen,fclose,fprintf,fscanf,fwrite,fread,fseek等。
- 使用C程式語言解決問題，不單是要熟C程式語言，並且要熟知各種有效組織處理資料的方式(修資料結構)，熟讀各種有效的演算方式及其適用問題(修演算法)，當然也要熟悉你要解的問題(專業領域知識)，知道處理它最妥當的演算方式，然後才能寫出有效率的程式。

```
#include<stdio.h>
int main()
{
    int n,u;
    scanf("%d",&n);
    for(u = 0; u < n; u++) {
        printf(“Hello world!\n");
    }
    return 0;
}
```

簡易C程式



函式名字是main

回傳值是
一個整數

大刮號內
是這個函
式的指令

註釋

一個C函式他的語法必須在函式名稱後跟著一對小刮號，小刮號內可以宣告C函式的參數。void代表沒參數。

```
/* A simple C program. */
#include<stdio.h>
int main(void)
{
 printf("Hello world!\n");
 return 0;
}
```

- `int main(void)`
  - 一個C程式就是由**main**這個函式開始被被執行的。
  - 一個C程式可以有許多函式(function)。
    - 其中只能有一個叫 main的函式。(why??)
  - 一個C函式計算完畢後可以傳回一個值。

簡易C程式 (cont’d)

- **`printf("Hello world!\n");`**
    - 呼叫printf這個函式做一些事
        - printf不是C程式語言的一部份，它是C標準程式庫裡的函式。
        - 函式呼叫：**printf(   );**
        - 函式參數：**`"Hello world!\n"`**（**雙引號內文字代表一個字串**）
            - **`\n`** 表示換行字元
    - 整行稱作一個敘述(statement)
        - C語言裡所有敘述必須以**分號(;)**作結束
- **`return 0;`**
    - 函式結束並傳回0

簡易C程式 (cont’d)

- 前處理器處理前處理指令
  - 以 # 開始的指令。
  - 就此例他會根據#include<stdio.h>指令載入stdio.h這個header。

- 編譯
  - 編譯器將C程式翻譯成目標程式(Object program)
    - 產生.obj檔

- 連結
  - 連結器會從程式庫裡找到printf的程式碼加到目標程式裡，整合成一個執行檔(executable file)

  - 若函式名字拼錯了會怎樣??

練習

- 這兩個程式會輸出什麼？

```
#include <stdio.h>
int main(void)
{
   printf( "Hello " );
   printf( "world!\n" );
   return 0;
}
```

```
#include <stdio.h>
int main(void)
{
   printf( "Hello\nworld!\n" );
   return 0;
}
```

C程式的一般結構

這部份指令是由preprocessor處理。如指令#include<stdio.h>是載入header file: stdio.h。

directives

declarations (global variables)

other functions

int main(void)

{

declarations (local variables);

statements;

}

變數宣告。

一個C程式必須有一個且只能有一個main函式，其中int表示其計算的結果為整數，void代表此函式沒有參數。

程式執行時的命令。例如printf("Hello world\n");與return 0;為兩個敘述。每個敘述必須有分號;作為敘述結束。

```
/* A simple C program. */
#include<stdio.h>
int max(int x,int y)
{
    return x>=y? x : y;
}
int main(void)
{
    printf("Hello world\n");
    return 0;
}
```

註釋 (comments)

- /* … */ 範圍內的都是註解，編譯器會忽略這些文字，不產生指令。一些與程式相關的資訊可以擺在註釋裡，如程式名字，作者，使用的方法，修改歷程等。
  - 寫給人看的!!!
  - 例子

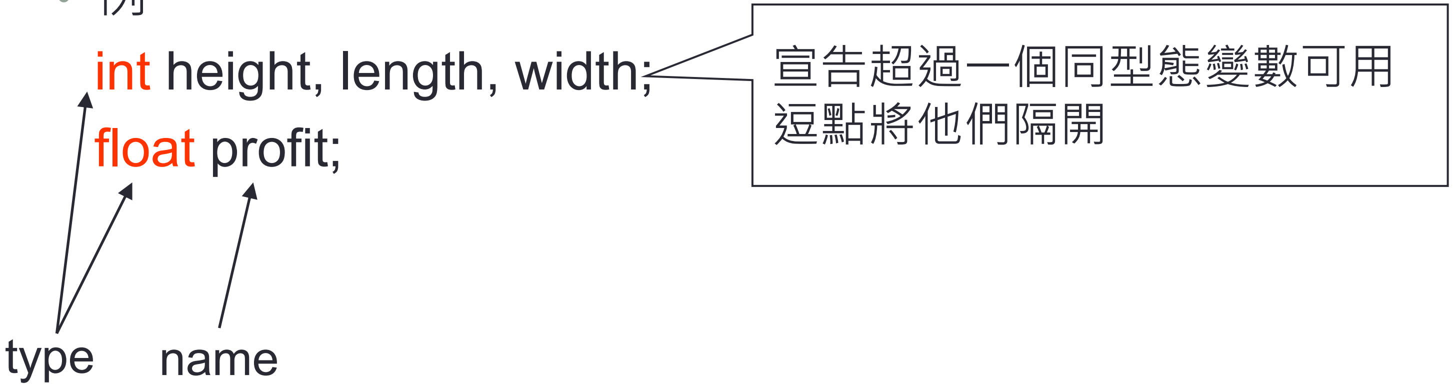
```
/* This is a comment. */
/*
   This is a comment, too.
*/
```

- C99也提供//來做註釋。從//至行尾都是註釋。
  - 例子

```
printf("Hello world!\n"); // This is a comment.
```

變數與變數宣告

- C程式語言裡有變數(variables)可作為資料儲存與操作。
- 變數必須有名字(name)與型態(type)，使用前必須先宣告(declaration)。
  - 例



```
int height, length, width;
float profit;
```

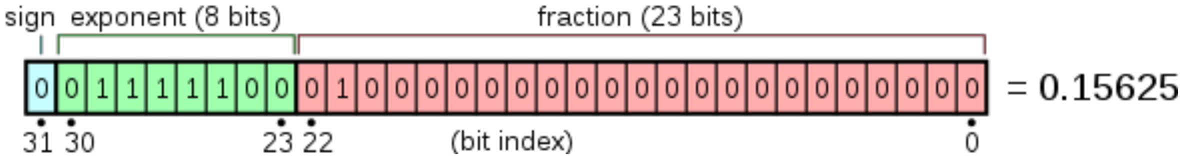
宣告超過一個同型態變數可用逗點將他們隔開

變數的性質:
Type:它是什麼
Scope:此變數在程式那些地方可以直接用到它
Livetime:此變數時麼時候生出(建構(C++))，甚麼時候消滅(解構(C++))

```
#include<stdio.h>

float f(float x,float a,float b,float c)
{
   // calculate a*x*x+b*x+c
    return a*x*x+b*x+c;
}

int main(void)
{
    float x,a,b,c;

    scanf("%f%f%f%f“,&x,&a,&b,&c);
    printf("%.2f\n",f(x,a,b,c));

    return 0;
}
```

Data Representation

- int
  - 2's complement
- float



識別符號(Identifier)

- 函式名稱，變數名稱，巨集名稱，標籤(label)名稱皆稱為識別符號(identifier)。
- 識別符號命名規則
  - 識別符號可由字母（A-Z, a-z）數字（0-9）與(_)所組成，但是不能以數字為開頭。
    - For example,
      - A1, a_234, and bx123 are identifiers.
      - 1x34, hello world, and hello@cs.ntou are not identifiers.
    - **分大小寫**
      - A1 and a1 are different identifiers.
  - 長度不限，但是C89只要求編譯器記住前31字元，而C99要求編譯器記住前63字元。

關鍵字(Keywords)

- **這些keywords是C程式語言保留的字。**
- **你不可以用這些字作為函式或變數名稱.**

auto	enum	restrict$	unsigned
break	extern	return	void
case	float	short	volatile
char	for	signed	while
const	goto	sizeof	_Bool$
continue	if	static	_Complex$
default	inline$	struct	_Imaginary$
do	int	switch	
double	long	typedef	
else	register	union	

$ C99 only

變數概念

- 變數
  - 變數名稱相當於記憶體的位置。
  - 每個變數都有名稱與型態。
  - 當新的值放到變數裡將會蓋掉先前變數的內容。
  - 讀取變數內容不會改變變數內容

integer1 45

C編譯器將會安排integer1在程式裡的位置，至於將來程式執行時integer1真正代表的那個記憶體的位置則還需連結器(Linker)與載入器(Loader)才能決定。

改變變數內容

- 可透過= (Assignment operator)來改變變數內容。
  - 注意=不是表示相等而是**將一個值塞到一個變數裡**
    (= is a binary operator (運算子) and has two operands(運算元))

  例如將height，length，width內容設定為常數(constant) 8，12，10。其指令如下：

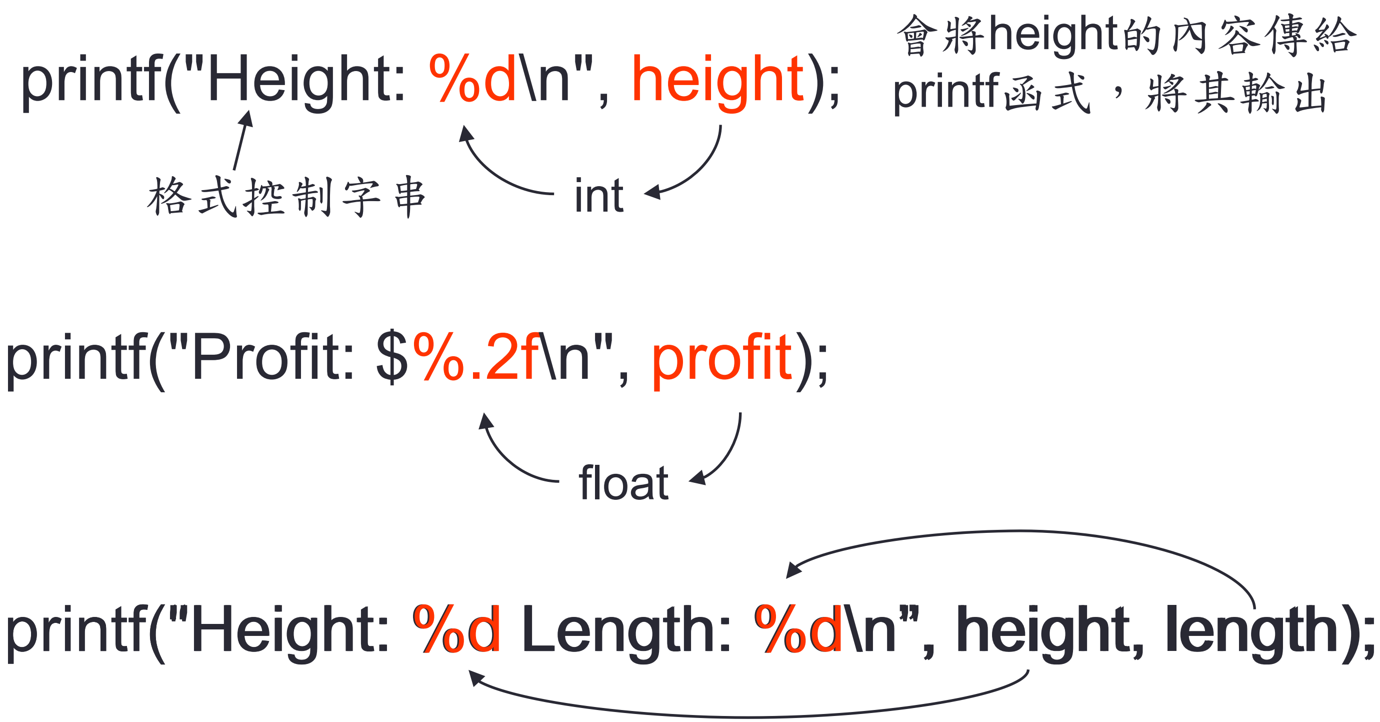
  ```
  height = 8; // read as assigning 8 to height
  length = 12;
  width = 10;
  ```

  - 思考下面指令在幹什麼？
    10 = width /* ?? */

```
height = 8;

height = 10;
```

顯示變數內容

- 呼叫標準函式printf來輸出變數內容。
  例如



```
printf("Height: %d\n", height);
```

會將height的內容傳給printf函式，將其輸出

格式控制字串

int

```
printf("Profit: $%.2f\n", profit);
```

float

```
printf("Height: %d Length: %d\n”, height, length);
```

```c
/* Fig. 2.5: fig02_05.c
   Addition program */
#include <stdio.h>

int main()
{
   int integer1, integer2, sum;          /* declaration */

   printf( "Enter first integer\n" );  /* prompt */
   scanf( "%d", &integer1 );           /* read an integer */
   printf( "Enter second integer\n" ); /* prompt */
   scanf( "%d", &integer2 );           /* read an integer */
   sum = integer1 + integer2;          /* assignment of sum */
   printf( "Sum is %d\n", sum );       /* print sum */

   return 0;  /* indicate that program ended successfully */
}
```

宣告變數

輸入兩個整數

計算和

輸出結果

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

輸入資料

- 呼叫標準函式scanf來輸入資料。

  例如

  ```
  int integer1;
  float profit;
  ```

scanf("%d", &integer1);

scanf為C標準程式庫裡的函式，它是用來輸入資料，這個scanf有兩個參數:

(1) "%d",*format control string,* 設定輸入資料的型態。其中**%d(*conversion specifier*)** –表示資料是10進位數字

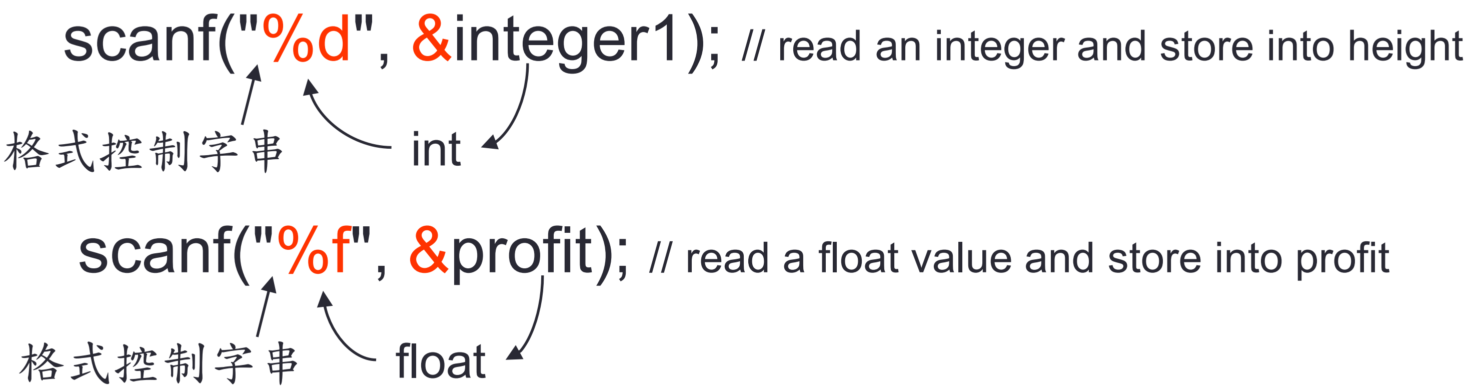
(2) &integer1– 輸入的資料要擺在integer1所代表的位置的記憶體裡

& address operator

&變數 會得到此變數的位置

將integer1的位置傳送給scanf，因此scanf才能知道要將讀入的資料送到記憶體哪個位置。

&是取得變數的位置。



```
scanf("%d", &integer1); // read an integer and store into height
```

```
scanf("%f", &profit); // read a float value and store into profit
```

```
scanf("%d%f", &integer1, &profit);
// read an integer and a float, and store them into integer1 and profit
```

練習

- int variable1, variable2;
- 問題一: scanf("%d", &variable2);會有什麼結果?
- 問題二: scanf("%d", variable2);會有什麼結果?
- 問題三: variable2=&variable1; scanf("%d", variable2);會有什麼結果?

Program:華氏轉攝氏

```
#include<stdio.h>
#define FREEZING_PT 32.0f
#define SCALE_FACTOR (5.0f / 9.0f)
int main(void)
{
   float fahrenheit, celsius;
   printf("Enter Fahrenheit temperature: ");
   scanf("%f", &fahrenheit);
   celsius = (fahrenheit –        32.0f        ) *      (5.0f / 9.0f)      ;
   printf("Celsius equivalent: %.1f\n", celsius);
   return 0;
}
```

應用巨集指令定義常數

文字置換

顯示單精確度浮點數至小數點下一位

應用巨集定義常數

- 例如

```
#define SCALE_FACTOR (5.0f / 9.0f)
```

  - 只要前處理器發現SCALE_FACTOR就會置換它為(5.0f / 9.0f)
    - 為什麼要加() ???不加會發生什麼事?
  - 必須在一行內完成
  - 若要跨行則在行尾加 \

```
#define myM(x,y)    (x-y)* \
                     (x+y)
```

- 程式裡用常數的名字可增加程式可讀性。
- 當常數值改變時只要改變其巨集定義即可。

例若SCALE_FACTOR的值要改變為5.0f/10.0f，那麼只要改變定義SCALE_FACTOR的巨集即可。

```
#define SCALE_FACTOR (5.0f / 10.0f)
```

基本型態

- C語言裡，資料必須有型態。基本資料型態如下：
  - 整數型態 (integer type)
  - 浮點數型態 (floating-point type)
  - 字元型態 (character type):後面章節再討論
- 另外有指標(pointer)與自定型態(typedef、struct、union)，還有一種沒型態的型態(void)。
  - 後面章節再討論

整數型態

- 整數型態
  - short int
  - unsigned short int
  - int $-2^{31} \sim 2^{31}-1$
  - unsigned int
  - long int
  - unsigned long int
  - long long unsigned $0 \sim 2^{64}-1$
  - long long int $-2^{63} \sim 2^{63}-1$
- 各型態整數範圍可查limits.h

整數常數寫法

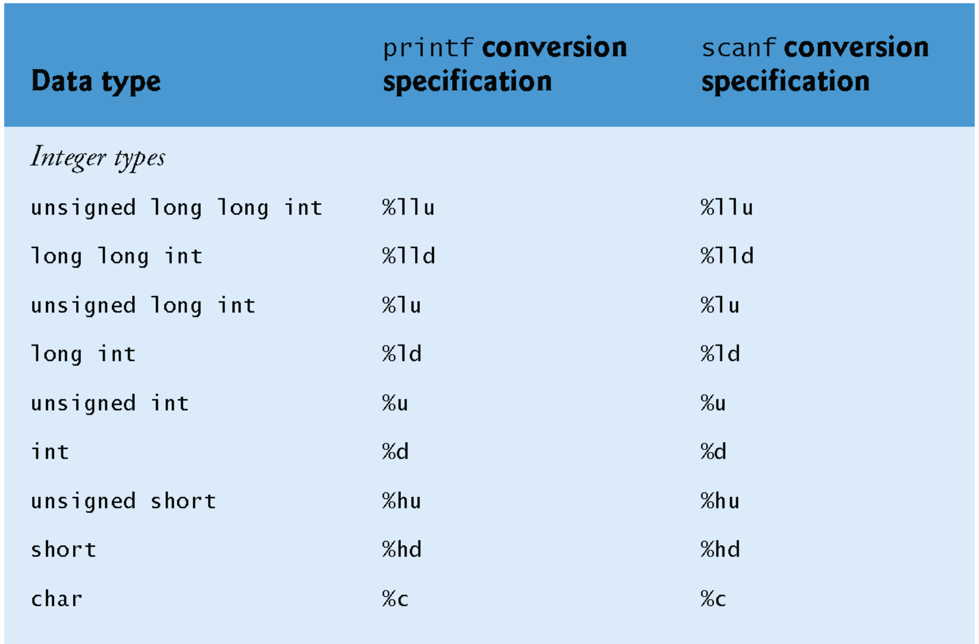
- 8進位數字
  - 只包含0~7的字元，開頭第一個字元必須是0
- 16進位數字
  - 只包含0~9,A~F的字元，開頭必須是0x
- 一般寫十進位整數它的型態是int，若值大到int無法表示，就會換成long int，再不行就換unsigned long int。
- 對8進位與16進位嘗試的順序為int，unsigned int，long int，unsigned long int。
- 若要表示long int常數在後面加L(l)
  - 15L 0377L 0x7fffL
- 若要表示unsigned常數在後面加U(u)
  - 15U
- 若要表示unsigned long int在後面加UL(ul)。
  - UL可任意大小寫與順序

溢位(overflow)

- 當int(or unsigned)不夠表示處理的數字時就發生溢位。
  - int溢位結果沒定義。
  - unsigned溢位結果就如同 % 2^n。

格式化輸出/入各式整數

- 轉換指引
  - %u unsigned
  - %o 8進位
  - %x 16進位
- 加h,l來表示short或long
  - %hd short int
  - %ld long int
- 加ll表示long long



Data type	`printf` **conversion specification**	`scanf` **conversion specification**
Integer types		
`unsigned long long int`	`%llu`	`%llu`
`long long int`	`%lld`	`%lld`
`unsigned long int`	`%lu`	`%lu`
`long int`	`%ld`	`%ld`
`unsigned int`	`%u`	`%u`
`int`	`%d`	`%d`
`unsigned short`	`%hu`	`%hu`
`short`	`%hd`	`%hd`
`char`	`%c`	`%c`

Fig. 5.5 | Arithmetic data types and their conversion specifications. (Part 2 of 2.)

浮點數型態

- 浮點數型態
  - float
  - double
  - long double
- 各型態浮點數範圍可查 float.h

浮點數常數寫法

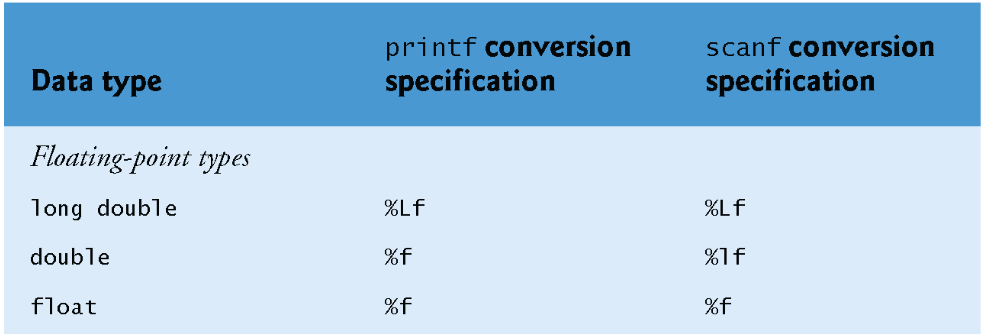
- 寫法

  67.0, 67., 67.0e0, 67e0, 6.7e1, 6.7e+1, 670e-1,…

- 浮點數常數內定為double型態。若要為float型態，數字尾加上f。

格式化輸出/入各式浮點數

- 轉換指引
  - printf輸出不管float或double都用%f, %e,%g
  - scanf使用%f, %e,%g輸入float，使用%lf, %le,%lg輸入double。
  - 對long double，不管printf或scanf皆用%Lf,%Lg,%Le。(L必須大寫)



Data type	`printf` **conversion specification**	`scanf` **conversion specification**
Floating-point types		
`long double`	`%Lf`	`%Lf`
`double`	`%f`	`%lf`
`float`	`%f`	`%f`

Fig. 5.5 | Arithmetic data types and their conversion specifications. (Part 1 of 2.)

C算術指令

- 數學運算子:+,-,*,/,%
  - + 加法
  - - 減法
  - * 乘法
  - / 除法
    - 整數除法只留商，餘數捨棄
      - 7 / 5 為 1
  - % 計算餘數
    - 7 % 5 為 2
- 運算子優先順序:
  - 有些運算子比其他運算子先計算(先*/%後+-)
    - 用()表示優先計算()內運算式
  - Example:
    下面哪一個可以正確計算 a, b, c的平均??
    - (i) a + b + c / 3 (ii) (a + b + c) / 3 (iii) 以上皆是 (iv)以上皆非
- 若整數與浮點數混在一起計算，最後結果為浮點數

z=p*r%q+w/x-y

C evaluates these operators in this sequence 6 1 2 4 3 5

```
#include<stdio.h>
int main()
{
        printf("10*20%%3+5/2-1=%d\n",10*20%3+5/2-1);
}
```

/ 與 % 注意事項

- 1 / 2為 0不是0.5。
    - 當整數/整數，除法結果會捨棄小數部分。因為1與2是整數，所以除法結果是0。
    - 當浮點數/整數或整數/浮點數，此時除法結果為浮點數，不會捨棄小數部分。
    - 若想得到0.5必須怎麼寫呢?
- **% 只適用於整數%整數。不然編譯器會提示錯誤。**
- 除數是0(/0 或 %0)時，程式執行時會發生錯誤。
    - 程式會產生中斷。
- 當整數/整數或整數%整數，若其中一個是負數時C89對計算結果沒明確定義(同學你可以試試手邊的編譯器)。
    - C99規則是整數/整數往0的方向去尾。因此 -9/7為-1;
    - C99規則是i % j的正負號決定於i，因此-9%7為-2。

整數除法

- int a, m=9, n=4;

- 無條件捨去法

  a = m/n; //a==2

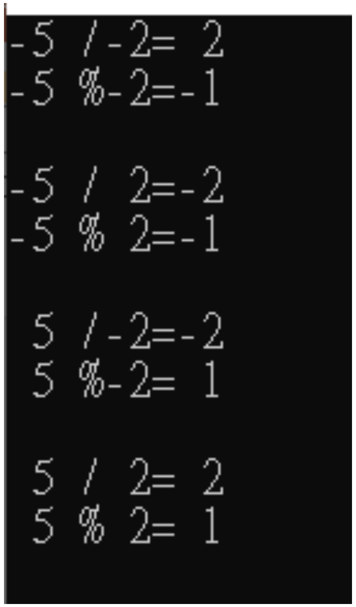
- 四捨五入法:

  a = (m+n/2)/n; //a==2

- 無條件進位法

  a = (m+n-1)/n; //a==3

```
#include<stdio.h>
int main()
{
    int x0 = 5, y0 = 2;
    int s1,s2;

    for(s1 = -1; s1 <= 1; s1+=2) {
       for(s2 = -1; s2 <= 1; s2+=2) {
            int x = x0*s1;
            int y = y0*s2;
            int z;
            z = x / y;
            printf("%2d /%2d=%2d\n",x,y,z);

            z = x % y;
            printf("%2d %%%2d=%2d\n\n",x,y,z);
       }
    }

    return 0;
}
```



```
-5 /-2= 2
-5 %-2=-1

-5 / 2=-2
-5 % 2=-1

 5 /-2=-2
 5 %-2= 1

 5 / 2= 2
 5 % 2= 1
```

Program: Computing a UPC check digit

```
#include <stdio.h>

int main(void)
{
  int d, i1, i2, i3, i4, i5, j1, j2, j3, j4, j5,
      first_sum, second_sum, total;

  printf("Enter the first (single) digit: ");
  scanf("%1d", &d);
  printf("Enter first group of five digits: ");
  scanf("%1d%1d%1d%1d%1d", &i1, &i2, &i3, &i4, &i5);
  printf("Enter second group of five digits: ");
  scanf("%1d%1d%1d%1d%1d", &j1, &j2, &j3, &j4, &j5);

  first_sum = d + i2 + i4 + j1 + j3 + j5;
  second_sum = i1 + i3 + i5 + j2 + j4;
  total = 3 * first_sum + second_sum;

  printf("Check digit: %d\n", 9 - ((total - 1) % 10));

  return 0;
}
```

1d告知scanf只要讀入一個digit

Assignment (=) operator

- j = 0; //將0存入j
- i = j = k = 5;
  - 與i = (j = (k = 5));同義。注意是由右到左。
- 注意此例

```
float x;
int i;
x=i=3.5;  //結果為i為3，x為3.0而不是3.5。
i=x=3.5;  //結果為i為3，x為3.5。
```

- 能擺到=左邊的稱lvalue。唯有變數才能擺到=左邊，其他如運算式(i+j)，常數都不行。以下都是錯的

```
12=i;
i+j=10;
-i=j;
```

The sizeof operator

- 語法

  sizeof (type-name)
  sizeof (variable name)
  sizeof (expression)

  - 得知type，variable或運算式結果佔幾個bytes

```
int x;
printf("the size of int is %d, the size of x is %d\n",sizeof(int), sizeof(x));
```

```
#include<stdio.h>
int main()
{
    printf(" sizeof(char):%d\n", sizeof(char));
    printf(" sizeof(short):%d\n",sizeof(short));
    printf(" sizeof(int):%d\n",sizeof(int));
    printf(" sizeof(unsigned):%d\n",sizeof(unsigned));
    printf(" sizeof(long long int):%d\n",sizeof(long long int));
    printf(" sizeof(float):%d\n",sizeof(float));
    printf(" sizeof(double):%d\n",sizeof(double));
    printf(" sizeof(long double):%d\n",sizeof(long double));

    return 0;
}
```

Code::Blocks 16.01(GCC)	**x86/x64 Visual C++ 2017**	**msys64\mingw64\bin\gcc**
sizeof(char):1	sizeof(char):1	sizeof(char):1
sizeof(short):2	sizeof(short):2	sizeof(short):2
sizeof(int):4	sizeof(int):4	sizeof(int):4
sizeof(unsigned):4	sizeof(unsigned):4	sizeof(unsigned):4
sizeof(long long int):8	sizeof(long long int):8	sizeof(long long int):8
sizeof(float):4	sizeof(float):4	sizeof(float):4
sizeof(double):8	sizeof(double):8	sizeof(double):8
sizeof(long double):12	sizeof(long double):8	sizeof(long double):16

型態轉換 (Type casting)

- **不同型態的資料能互相轉換嗎?**
  x = y; //將y的內容塞到x內，若兩者容量不同怎麼塞?

- **編譯器自動作型態轉換 (implicit conversion)**
  - 運算元型態不符，而且編譯器知道轉換規則。
    - 運算元型態需相符，運算子才能運作。
    例子
    float x; int y;
    x < y (必須將y轉成float，<才能進行比較)
  - **對非assignment(=)運算子的型態轉換原則：型態簡單的轉往型態複雜的，因為這個轉換數值大小基本上不會有問題。**
    - 運算元皆為浮點數
      - float→double→long double (運算元型態位在此順序較左邊的轉)
    - 運算元皆為整數
      - int→unsigned int→long int→unsigned long int(運算元型態位在此順序較左邊的轉)
    - 運算元一個為浮點數，另一個為整數
      - 整數→浮點數
      int x;
      long double z, y;
      z = y + x; //x→long double
  - **assignment(=)運算子的型態轉換原則: =右邊的運算元型態轉成=左邊的運算元型態。**
    例子
    float x; int y;
    y = x; // the value of x is converted to int.
    - 不可指定超過範圍的常數數字給變數
    char c;
    C = 10000; // wrong

- **直接寫出轉換方式(explicit conversion)**
  延續上面例子
  y = (float) x;

型態轉換 (Type casting)

- 語法

  (type-name) expression
  - 將expression的形態轉換為type-name型態。
  - 其有最高運算優先權

  例子

  int x, y;
  float z;
  z = x / y;
  /* 將先進行整數除法，再將結果轉成float型態 */

  z = (float) x / y;
  /* 將先把x的值轉成float，再進行除法，由於/兩邊型態不一致，編譯器會再將y的值型態轉成float，然後執行除法，最後將結果給z */

  x = 1;
  y = 2;
  z = x/y; // z is now 0
  z =(float) x/y; // z is now 0.5

The if statement

- 語法

```
if (logical expression) statement

if (logical expression) {
  //If logical expression is true, execute statements.
          statements
}
if (logical expression) {
          statements
} else {
          statements
}
```

- 例子
  - `if (i != 5) printf("%d\n", i);`
  - If (i < 5 && j != 0) {

```
    i = i / j;
    printf("%d\n", i);
  }
```

邏輯運算式 (Logical expression)

- 邏輯運算式 (logical or Boolean type)的值為下面其中一個
  - True (真)
  - False (偽)
  - C語言裡，若運算式的結果不是0，以邏輯運算式來解讀其值為就是true。運算式的結果是0，以邏輯運算式來解讀其值為false;
  - 例子: x = 1; if(x+1) {…} // x+1 is true
  - if(x-1) {} // x-1 is equal to 0 and is false
- 透過關係運算子，相等運算子，邏輯運算子構成邏輯運算式。

邏輯運算子(logical operators)

- 邏輯運算子(logical operators)
  - ! 否: if (! expr){ }
  - && 且: if (expr1 && expr2) {}
  - || 或: if (expr1 || expr2) {}
- 透過邏輯運算子可以結合邏輯運算式
  - (i != 0) && (j / i >0)

&&	T	F
T	T	F
F	F	F

||	T	F
T	T	T
F	T	F

!	T	F
	F	T

關係運算子(relational operators)

- 關係運算子(relational operators)
  - >　大於
  - <　小於
  - >=　不小於
  - <=　不大於
- 可以用來比較整數或浮點數間大小關係

例子

10 < 11結果為真(整個運算式的值其實為1)
11 < 10結果為偽(整個運算式的值其實為0)

- 關係運算子的運算優先權比算數運算子低，因此i+j<u+v其實就是(i+j) < (u+v)
- **關係運算子為左結合律，**
  因此10 < 9 < 8其實為(10<9)<8 ⇒ 0 < 8 ⇒ 1 為真
  - **若要描述x > 5且 x < 9，要寫成x > 5 && x < 9，而不是5<x<9。**

相等運算子(equality operators)

- 相等運算子(equality operators)
  - == 等於 (為兩個==)
  - != 不等於
  - 切記==不要寫成=
    - 例如 i == 100 寫成 i = 100，原本是判斷i是否為100，變成100設定至i。
    - 為了避免這種不小心的錯，可寫為100==i。因為若寫成100=i，編譯器會抓到這個錯誤。
- 相等運算子運算優先權低於關係運算子
  - i < j == j < k 其實是 (i<j)==(j<k)

安全的使用printf

- 針對安全性考量，使用printf時要避免單一引數。
  - 例子一
    - printf("Hello world\n"); //單一引數，改寫之。
    - puts("Hello world");
  - 例子二
    - printf("Hello world"); //單一引數，改寫之。
    - printf("%s","Hello world");

typedef: 定義型態別名 (synonym)

已有的型態 別名

- 語法

  typedef int Bool;

  - 編譯器會加入這個型態並將識別此型態別名。
  - 與用巨集有什麼差別呢？

  #define Bool int

- 好處
  - 提高程式可讀性
    - 型態名稱更達意

    typedef float Dollars; Dollars cash_in, cash_out;
  - 提高程式移植度
    - typedef unsigned char BYTE;   // define a 8-bit data type
    - typedef unsigned short WORD; // define a 16-bit data type
    - C標準程式庫也因此用這個技巧定義了size_t, ptrdiff_t, wchar_t
  - 方便程式修改