C語言練習心得報告

讀書會組員:09唐崙祥、35俞煥崙、36吳劉軒

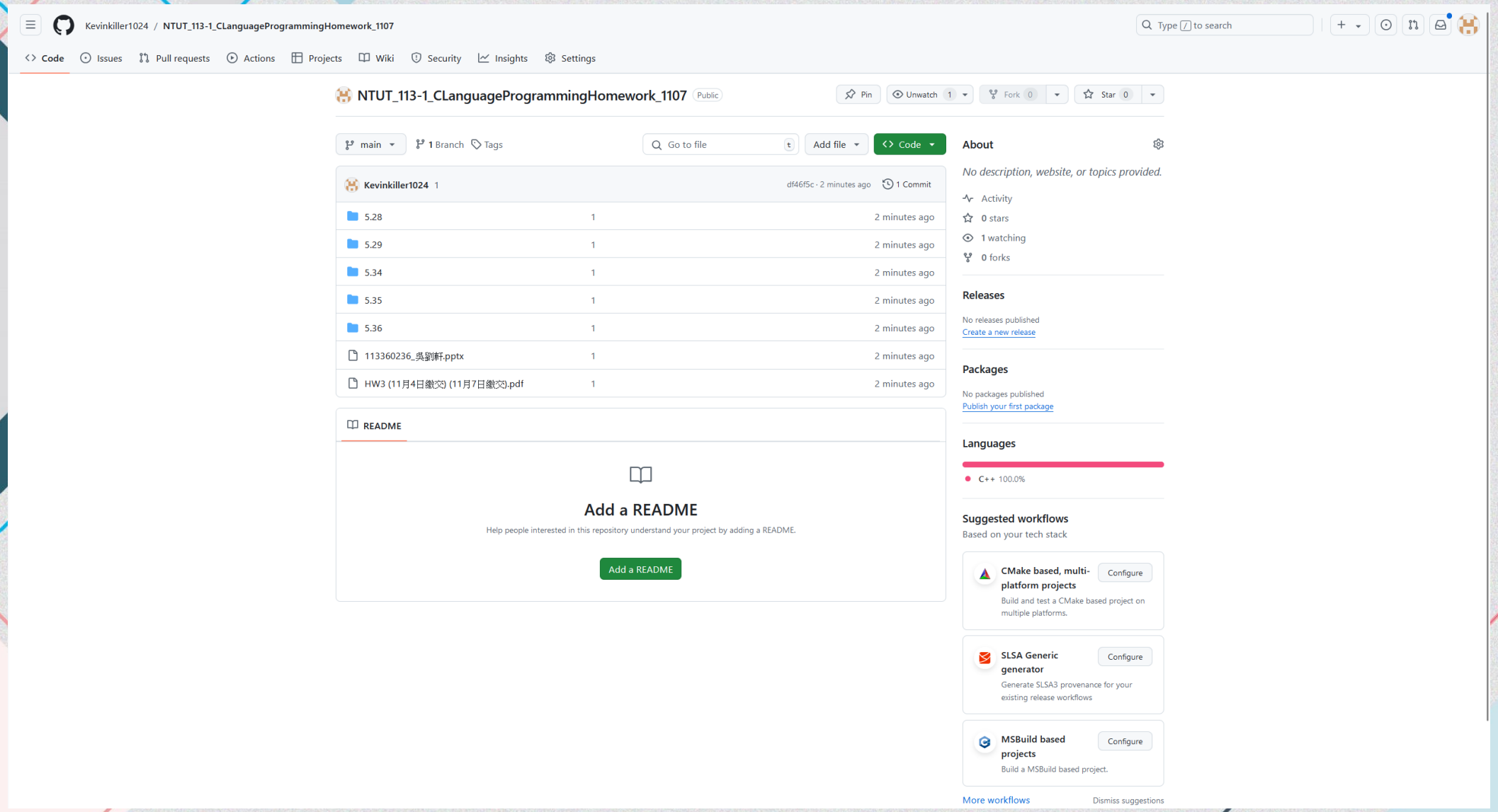
討論時間:2024/11/1

地點:二教203

GitHub程式連結

https://github.com/Kevinkiller1024/NTUT_113-1_CLanguageProgrammingHomework_1107.git

GitHub截圖


Kevinkiller1024 / NTUT_113-1_CLanguageProgrammingHomework_1107
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
NTUT_113-1_CLanguageProgrammingHomework_1107
Public
Pin
Unwatch 1
Fork 0
Star 0
main
1 Branch
Tags
Go to file
t
Add file
Code
Kevinkiller1024 1
df46f5c · 2 minutes ago
1 Commit
5.28 1 2 minutes ago
5.29 1 2 minutes ago
5.34 1 2 minutes ago
5.35 1 2 minutes ago
5.36 1 2 minutes ago
113360236_吳劉軒.pptx 1 2 minutes ago
HW3 (11月4日繳交) (11月7日繳交).pdf 1 2 minutes ago
README
Add a README
Help people interested in this repository understand your project by adding a README.
Add a README
About
No description, website, or topics provided.
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
C++ 100.0%
Suggested workflows
Based on your tech stack
CMake based, multi-platform projects
Configure
Build and test a CMake based project on multiple platforms.
SLSA Generic generator
Configure
Generate SLSA3 provenance for your existing release workflows
MSBuild based projects
Configure
Build a MSBuild based project.
More workflows
Dismiss suggestions

心得

$$\mathrm{LCM}(a,b) = \frac{|a \times b|}{\mathrm{GCD}(a,b)}$$

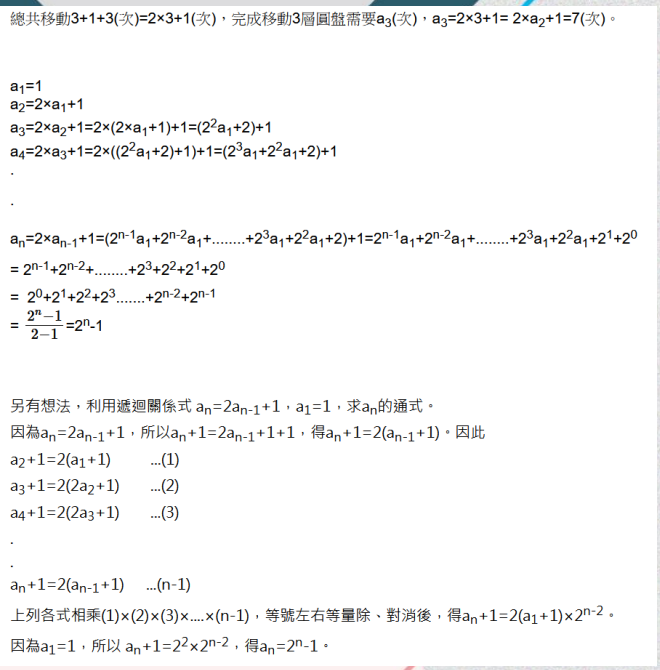
這次的作業我覺得有一點點難，尤其是5.36，先講5.28，我自己是使用ascii碼來做的，就是判斷輸入的字在0x41到0x5A之間的就是大寫，要轉化成小寫就把他加上0x20，反過來就是減掉，除此之外我只能想到用if或switch來做了。

5.29的最小公倍數花了我一點時間，但是我還記得跟他類似的最大公因數的程式，但一直連結不再一起，後來問GPT才知道最小公倍數＝兩數相乘／最大公因數，這才解開我的問題。

5.34就要開始用遞迴了。

5.35的費氏數列它好像有規定說不能用遞迴，雖然我覺得用遞迴要把他一項一項的列出來比較難，但是可以很快知道該項的結果，可是題目說不能用，而且還是要一項一項列出來，所以我最後用了for迴圈做。

5.36的河內塔在高中階段只是有玩過，還沒有寫過程式去解決它，也只知道要使用遞迴來解而已，接著上網查到了程式，但我不知道他的運作原理，後來把程式丟給GPT請他講解給我看，我才明白它的原理，還有原來河內塔是有公式的，也才知道河內塔原本是一個數學問題。



總共移動3+1+3(次)=2×3+1(次)，完成移動3層圓盤需要a_3(次)，$a_3=2\times3+1=2\times a_2+1=7$(次)。

$a_1=1$
$a_2=2\times a_1+1$
$a_3=2\times a_2+1=2\times(2\times a_1+1)+1=(2^2a_1+2)+1$
$a_4=2\times a_3+1=2\times((2^2a_1+2)+1)+1=(2^3a_1+2^2a_1+2)+1$
.
.

$a_n=2\times a_{n-1}+1=(2^{n-1}a_1+2^{n-2}a_1+.....+2^3a_1+2^2a_1+2)+1=2^{n-1}a_1+2^{n-2}a_1+.....+2^3a_1+2^2a_1+2^1+2^0$
$=2^{n-1}+2^{n-2}+.....+2^3+2^2+2^1+2^0$
$=2^0+2^1+2^2+2^3.....+2^{n-2}+2^{n-1}$
$=\frac{2^n-1}{2-1}=2^n-1$

另有想法，利用遞迴關係式 $a_n=2a_{n-1}+1$，$a_1=1$，求a_n的通式。
因為$a_n=2a_{n-1}+1$，所以$a_n+1=2a_{n-1}+1+1$，得$a_n+1=2(a_{n-1}+1)$，因此
$a_2+1=2(a_1+1)$ ...(1)
$a_3+1=2(2a_2+1)$ ...(2)
$a_4+1=2(2a_3+1)$ ...(3)
.
.
$a_n+1=2(a_{n-1}+1)$ ...(n-1)
上列各式相乘(1)×(2)×(3)×....×(n-1)，等號左右等量除、對消後，得$a_n+1=2(a_1+1)\times2^{n-2}$。
因為$a_1=1$，所以 $a_n+1=2^2\times2^{n-2}$，得$a_n=2^n-1$。

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

int moved = 0;  // 记录移动的次数

// hanoi函数，用来解决汉诺塔问题
void hanoi(int n, char A, char B, char C) {
    if (n == 1) {
        // 基本情况：只需将一个盘子从 A 移到 C
        printf("把 %c 移到 %c\n", A, C);
        moved++;
    }
    else {
        // 递归步骤：

        // Step 1: 将 n-1 个盘子从 A 移到 B，C 作辅助
        hanoi(n - 1, A, C, B);

        // Step 2: 将第 n 个盘子从 A 移到 C
        hanoi(1, A, B, C);

        // Step 3: 将 n-1 个盘子从 B 移到 C，A 作辅助
        hanoi(n - 1, B, A, C);
    }
}

int main() {
    int n;
    printf("請輸入有幾層：");
    scanf_s("%d", &n);  // 读取盘子的层数
    hanoi(n, 'A', 'B', 'C');  // 调用 hanoi 函数
    printf("\n總共移動 %d 步\n\n", moved);
    system("pause");
    return 0;
}
```

递归调用过程

假设 `n = 3` 时，`hanoi(3, 'A', 'B', 'C')` 的执行过程如下：

1. 调用 `hanoi(2, 'A', 'C', 'B')`：先将 2 个盘子从 A 移动到 B。
   - 调用 `hanoi(1, 'A', 'B', 'C')`：将 1 号盘子从 A 移动到 C。
   - 移动 2 号盘子从 A 到 B。
   - 调用 `hanoi(1, 'C', 'A', 'B')`：将 1 号盘子从 C 移动到 B。
2. 移动 3 号盘子从 A 到 C。
3. 调用 `hanoi(2, 'B', 'A', 'C')`：将 2 个盘子从 B 移动到 C。
   - 调用 `hanoi(1, 'B', 'C', 'A')`：将 1 号盘子从 B 移动到 A。
   - 移动 2 号盘子从 B 到 C。
   - 调用 `hanoi(1, 'A', 'B', 'C')`：将 1 号盘子从 A 移动到 C。

整个过程中递归调用不断进行，每次 `moved` 计数器会加一。

递归退出条件

当 `n == 1` 时，函数不再递归，直接执行移动操作，并退出。这是递归的基本情况，也是递归调用的终止条件。

输出示例

假设输入 `n = 3`，输出可能是：

```css
請輸入有幾層：3
把 A 移到 C
把 A 移到 B
把 C 移到 B
把 A 移到 C
把 B 移到 A
把 B 移到 C
把 A 移到 C

總共移動 7 步
```

总结

该程序通过递归将 `n` 层盘子分解为三步操作，利用辅助柱子逐步完成汉诺塔的移动，同时使用 `moved` 变量记录移动的总次数。