package main

import (

"fmt"

"time"

)

// 判断是否为闰年

func isLeapYear(year int) bool {

return (year%4 == 0 && year%100!= 0) || year%400 == 0

}

// 获取指定月份的天数

func getDaysInMonth(year int, month int) int {

days := 31

if month == 4 || month == 6 || month == 9 || month == 11 {

days = 30

} else if month == 2 {

if isLeapYear(year) {

days = 29

} else {

days = 28

}

}

return days

}

// 获取1900年1月1日到指定年份、月份1日的总天数

func getTotalDays(year int, month int) int {

totalDays := 0

for y := 1900; y < year; y++ {

if isLeapYear(y) {

totalDays += 366

} else {

totalDays += 365

}

}

for m := 1; m < month; m++ {

totalDays += getDaysInMonth(year, m)

}

return totalDays

}

func printCalendar(year int, month int) {

monthNames := []string{"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"}

daysInMonth := getDaysInMonth(year, month)

totalDays := getTotalDays(year, month)

fmt.Printf(" %s %d\n", monthNames[month-1], year)

fmt.Println("Sun Mon Tue Wed Thu Fri Sat")

firstDay := (totalDays + 1) % 7

for i := 0; i < firstDay; i++ {

fmt.Printf(" ")

}

for d := 1; d <= daysInMonth; d++ {

fmt.Printf("%4d", d)

if (firstDay+d)%7 == 0 {

fmt.Println()

}

}

if (firstDay+daysInMonth)%7!= 0 {

fmt.Println()

}

}

func main() {

var year, month int

fmt.Println("请输入年份:")

fmt.Scanln(&year)

fmt.Println("请输入月份:")

fmt.Scanln(&month)

printCalendar(year, month)

}

#include <stdio.h> #include <stdlib.h> #include <string.h> #include <time.h> #include <unistd.h> // for usleep() on Unix-like systems #ifdef \_WIN32 #include <windows.h> // for Sleep() on Windows #define usleep(x) Sleep(x / 1000) #endif #define WIDTH 80 #define HEIGHT 24 #define DROP\_SPEED 100000 // Microseconds (100000 us = 0.1 sec) #define CHARACTERS "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&\*()-\_=+[]{}|;:'\",.<>?/" void clear\_screen() { // ANSI escape code to clear the screen printf("\033[H\033[J"); } void move\_down(char screen[HEIGHT][WIDTH], int \*drops) { for (int i = HEIGHT - 1; i > 0; i--) { for (int j = 0; j < WIDTH; j++) { screen[i][j] = screen[i - 1][j]; } } // Clear the top row for (int j = 0; j < WIDTH; j++) { screen[0][j] = ' '; } // Update the drop positions for (int i = 0; i < HEIGHT; i++) { if (screen[i][drops[i] % WIDTH] == ' ') { screen[i][drops[i] % WIDTH] = CHARACTERS[rand() % strlen(CHARACTERS)]; } drops[i] += 1; } } int main() { srand(time(NULL)); char screen[HEIGHT][WIDTH]; int drops[HEIGHT]; // Initialize the screen and drop positions clear\_screen(); memset(screen, ' ', sizeof(screen)); for (int i = 0; i < HEIGHT; i++) { drops[i] = rand() % WIDTH; } while (1) { // Print the current screen for (int i = 0; i < HEIGHT; i++) { printf("%.\*s\n", WIDTH, screen[i]); } // Move the code down and update the screen move\_down(screen, drops); // Clear the screen before the next frame clear\_screen(); // Sleep for a short period to control the speed of the animation usleep(DROP\_SPEED); } return 0; }