程式設計

教課教授：謝東儒

助教 :蔡詠聿、吳品頤

Chapter 11_project 1

Modify Programming Project 7 from Chapter 2 so that it includes the following function:

```
void pay_amount(int dollars, int *twenties, int *tens, int *fives, int *ones);
```

The function determines the smallest number of $20, $10, $5, and $1 bills necessary to pay the amount represented by the dollars parameter. The twenties parameter points to a variable in which the function will store the number of $20 bills required. The tens, fives, and ones parameters are similar.

```
Enter a dollar amount : 76

$20 bills : 3
$10 bills : 1
$5 bills : 1
$1 bills : 1

--------------------------------
Process exited after 2.823 seconds with return value 0
請按任意鍵繼續 . . .
```

Solution

```c
//  dollar change

#include <stdio.h>


void pay_amount(int dollars, int *twenties, int *tens, int *fives, int *ones);

int main(void){

    int amount, twenties, tens, fives, ones;

    printf("Enter a dollar amount : ");
    scanf("%d", &amount);

    pay_amount(amount, [redacted]);

    printf("\n");

    printf("$20 bills : %d\n", twenties);
    printf("$10 bills : %d\n", tens);
    printf("$5 bills : %d\n", fives);
    printf("$1 bills : %d\n", ones);

    return 0;
}
```

Solution

```
26
27 void pay_amount(int dollars, int *twenties, int *tens, int *fives, int *ones){
28
29     *twenties = dollars / 20;
30     dollars -= *twenties * 20;
31
32     *tens = dollars / 10;
33     dollars -= *tens * 10;
34
35     *fives = dollars / 5;
36     *ones =        % 5;
37 }
38
```

Chapter 11_project 2

Modify Programming Project 8 from Chapter 5 so that it includes the following function:

```
void find_closest_flight(int desired_time, int *departure_time, int *arrival_time);
```

This function will find the flight whose departure time is closest to **desired_time** (expressed in minutes since midnight). It will store the departure and arrival times of this flight (also expressed in minutes since midnight) in the variables pointed to by **departure_time** and **arrival_time**, respectively.

Departure time	*Arrival time*
8:00 a.m.	10:16 a.m.
9:43 a.m.	11:52 a.m.
11:19 a.m.	1:31 p.m.
12:47 p.m.	3:00 p.m.
2:00 p.m.	4:08 p.m.
3:45 p.m.	5:55 p.m.
7:00 p.m.	9:20 p.m.
9:45 p.m.	11:58 p.m.

```
Enter a 24-hour time : 13:15
Closest departure time is 12:47 p.m. ,arriving at 3:00 p.m.

--------------------------------
Process exited after 1.689 seconds with return value 0
請按任意鍵繼續 . . .
```

Solution

```c
// flight

#include <stdio.h>

#define HOURS_PER_HALF_DAY 12
#define MINUTES_PER_HOUR 60
#define MINUTES_PER_HALF_DAY  (HOURS_PER_HALF_DAY * MINUTES_PER_HOUR)

#define SIZE ((int)(sizeof(departures) / sizeof(departures[0])))

void find_closest_flight(int desired_time, int *departure_time, int *arrival_time);


int main(void){

    int hours, minutes, desired_time, departure_time,
            departure_hour, arrival_time, arrival_hour;

    printf("Enter a 24-hour time : ");
    scanf("%d:%d", &hours, &minutes);

    desired_time = hours * MINUTES_PER_HOUR + minutes;

    find_closest_flight(desired_time, &departure_time, &arrival_time);
```

Solution

```
25
26      printf("Closest departure time is ");
27
28      departure_time = departure_time / MINUTES_PER_HOUR;
29      if(departure_hour == 0){
30          departure_hour = HOURS_PER_HALF_DAY;
31      }else if(departure_hour > HOURS_PER_HALF_DAY){
32          departure_hour -= HOURS_PER_HALF_DAY;
33      }
34      printf("%d:%.2d ", departure_hour, departure_time % MINUTES_PER_HOUR);
35
36      if(departure_time <                    ){
37          printf("a.m.");
38      }else{
39          printf("p.m.");
40      }
41
42
```

Solution

```
42
43      printf(" ,arriving at ");
44
45      arrival_hour = arrival_time / MINUTES_PER_HOUR;
46      if(arrival_hour == 0){
47          arrival_hour = HOURS_PER_HALF_DAY;
48      }else if(arrival_hour > HOURS_PER_HALF_DAY){
49          arrival_hour -= HOURS_PER_HALF_DAY;
50      }
51      printf("%d:%.2d ", arrival_hour, arrival_time % MINUTES_PER_HOUR);
52
53      if(arrival_time <                    ){
54          printf("a.m.");
55      }else{
56          printf("p.m.");
57      }
58      printf("\n");
59
60
61      return 0;
62  }
63
64
```

Solution

```
64
65  void find_closest_flight(int desired_time, int *departure_time, int *arrival_time){
66
67      int departures[] = {480, 583, 679, 767, 840, 945, 1140, 1305},
68          arrivals[] = {616, 712, 811, 900, 968, 1075, 1280, 1438}, closest;
69
70      if(desired_time <= departures[0]){
71          closest = [redacted];
72      }else if(desired_time > departures[SIZE - 1]){
73          closest = [redacted];
74      }else{
75          closest = 0;
76          while(desired_time > departures[closest + 1]){
77              closest++;
78          }
79          if((departures[closest + 1] - desired_time) < (desired_time - departures[closest])){
80              [redacted];
81          }
82      }
83
84      *departure_time = departures[closest];
85      *arrival_time = arrivals[closest];
86  }
```

Chapter 11_project 3

Modify Programming Project 3 from Chapter 6 so that it includes the following function:

```
void reduce(int numerator, int denominator,
            int *reduces_numerator,
            int * reduces_denominator);
```

numerator and **denominator** are the numerator and denominator of a fraction, **reduced_numerator** and **reduced_ denominator** are pointers to variables in which the function will store the numerator and denominator of the fraction once it has been reduced to lowest terms.

```
Enter a fraction : 18/45
In lowest terms : 2/5

--------------------------------
Process exited after 14.99 seconds with return value 0
請按任意鍵繼續 . . .
```

Solution

```c
// reduce_fraction

#include <stdio.h>

#define STACK_SIZE 100

int find_gcd(int m, int n);
void reduce(int numerator, int denominator,
            int *reduces_numerator,
            int * reduces_denominator);

int main(void){

    int num, denom;

    printf("Enter a fraction : ");
    scanf("%d/%d", &num, &denom);

    reduce(num, denom, [redacted], [redacted]);
    printf("In lowest terms : %d/%d\n", num, denom);

    return 0;
}
```

Solution

```c

int find_gcd(int m, int n){

    while(n != 0){
        int remainder = m % n;
        m = n;
        n = remainder;
    }
    return m;
}

void reduce(int numerator, int denominator,
            int *reduces_numerator,
            int * reduces_denominator){

    int gcd = find_gcd(numerator, denominator);

    /* Divide both numerator and denominator by GCD */
    *reduces_numerator = numerator / gcd;
    *reduces_denominator = denominator / gcd;

    /*  Ensure that denominator is positive */
    if(*reduces_denominator < 0){
        [redacted] *= -1;
        [redacted] *= -1;
    }
}

```