ព្រះរាជាណាចក្រកម្ពុជា

ជាតិ សាសនា ព្រះមហាក្សត្រ

វិទ្យាស្ថានបច្ចេកវិទ្យាកម្ពុជា

ដេប៉ាតឺម៉ង់ថ្នាក់ឆ្នាំសិក្សាមូលដ្ឋាន

TP7-Using loop DO WHILE

Subject: Data strucure & Programming

Student name	ID	score
1.SOUNG VICHKA	e20240672	…..

Teacher: Mr.ROEUN Pachroth

Problem1: Write a program to display all even numbers in this range [2, 2000] using a DO WHILE loop.

Example:

Output:

All even numbers between [2, 2000] are: 2 4 6 8 10 …… 2000

Code:

problem1.c ...\TP7 | problem1.c ...\array.c | array.c | arraytes

Assignment (part 2) > TP6 > TP7 > problem1.c > main()

```c
#include <stdio.h>

int main() {
    int i = 1;
    do {
        if (i % 2 == 0) {
            printf(" %d", i);
        }
        i++;
    } while (i <= 2000);

    return 0;
}
```

Loop and Logic Explanation

```
int i = 1;
```

- Starts the counter `i` at 1.

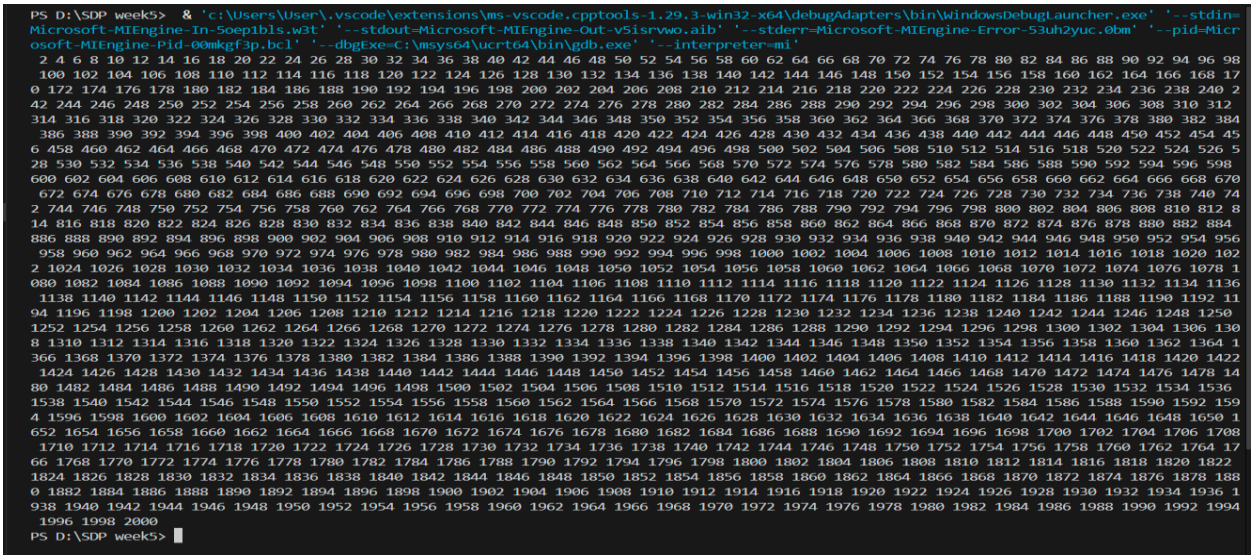
```
do {
   if (i % 2 == 0) {
      printf(" %d", i);
   }
   i++;
} while (i <= 2000);
```

- **`do { ... } while (i <= 2000);`** is a loop that:
  - Runs the code block **at least once**.
  - Keeps going as long as `i` is **less than or equal to 2000**.

Inside the loop:

- `if (i % 2 == 0)` checks if `i` is even:
  - `%` is the **modulus operator** — it gives the remainder.
  - If `i % 2 == 0`, then `i` is divisible by 2 → it's even.
- `printf(" %d", i);` prints the even number with a space before it.
- `i++;` increases `i` by 1 for the next round.

Rsult:



Problem2: Write a program that prints the powers of 2 up to 2^10 (1024) using a DO WHILE loop.

Example:

Output:

```
2^0 = 0
2^1 = 2
2^2 = 4
2^3 = 8
2^4 = 16
2^5 = 32
2^6 = 64
2^7 = 128
2^8 = 256
2^9 = 512
2^10 = 1024
```

Code:

problem2.c ...\TP7 | problem3.c ...\TP6 | problem3.c ...\TP7 | problem4.c ...\TP

Assignment (part 2) > TP6 > TP7 > problem2.c > ...

```c
#include<stdio.h>
#include<math.h>
int main(){
 int i=0;
 do
 {
    int sum=pow(2,i);
    printf("2^%d=%d\n",i,sum);
    i++;
 } while (i<=10);

}
```

Logic of the Program

```
int i=0;
```

- Starts the counter `i` at 0.

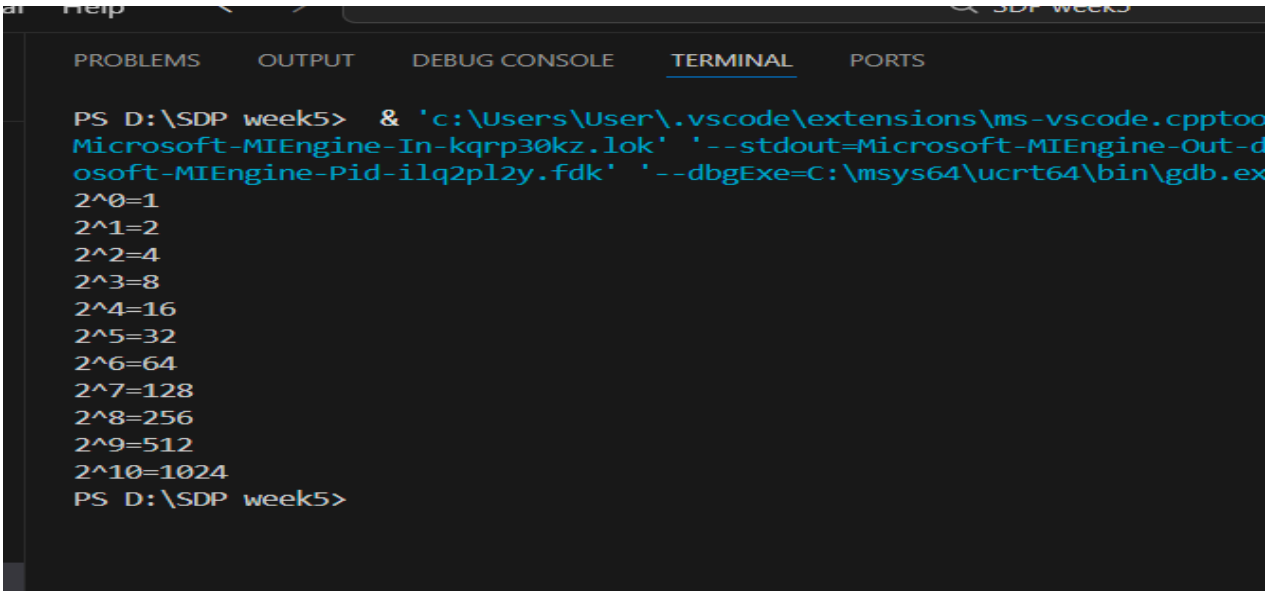
```
do {
    int sum = pow(2, i);
    printf("2^%d=%d\n", i, sum);
    i++;
} while (i <= 10);
```

- This is a **do...while loop**:
    - Runs the block at least once.
    - Keeps looping while `i <= 10`.

Inside the loop:

- `pow(2, i)` calculates (2^i) using the **math library**.
- The result is stored in `sum`.
- `printf("2^%d=%d\n", i, sum);` prints the exponent and the result in the format:
- 2^0=1
- 2^1=2
- 2^2=4
- ...
- 2^10=1024
- `i++` increases `i` by 1 for the next round.

Result:



Problem3: Write a program to ask a user for two integer numbers **m** and **n**. The program computes summation of numbers starting from m to n using a DO WHILE loop.

Examples:

INPUT:

Enter a number m: 5
Enter a number n: 10

Output:

The sum is: 5+6+7+8+9+10 = 45

.....

INPUT:

Enter a number m: 8
Enter a number n: 2

Output:

Oop! We can not compute the summation. n must be bigger than m.

Code:

TP6 | problem1 copy.c ...\TP7 | problem1 copy 2.c | problem2.c ...\TP6 | problem2.c ...\TP7 | prob

Assignment (part 2) > TP6 > TP7 > problem3.c > main()

```c
#include<stdio.h>
int main(){
int n,m,sum=0;
printf("Inter a number m :");
scanf("%d",&m);
printf("Inter a number n:");
scanf("%d",&n);
if(n>=m){
    int i=m;
    printf("The sum is:");
    do {
        sum+=i;
        printf("%d ",i);
        if(i<n){
            printf("+");
        }
        i++;
    }while (i<=n);
    printf("=%d\n",sum);
} else{
    printf("Oop! We can not compute the summation. n must be bigger than m.");
}

}
```

Explanation

```
int n, m, sum=0;
```

- Declares two integers `n` and `m` (the range limits).
- Initializes `sum` to 0 (to accumulate the total).

```
printf("Inter a number m :");
scanf("%d",&m);
printf("Inter a number n:");
scanf("%d",&n);
```

- Prompts the user to enter values for `m` and `n`.

```
if(n >= m) {
```

- Ensures that `n` is **greater than or equal to** `m`.
- If not, the program prints an error message.

```
int i = m;
printf("The sum is:");
do {
   sum += i;          // add current i to sum
   printf("%d ", i);  // print the number
   if(i < n) {
      printf("+");   // print "+" between numbers
   }
   i++;              // move to next number
} while (i <= n);
```

- Starts with `i = m`.
- Uses a **do...while loop** to:
  - Add each number from `m` to `n` into `sum`.
  - Print the number and a `+` sign (except after the last one).
- Stops when `i` exceeds `n`.

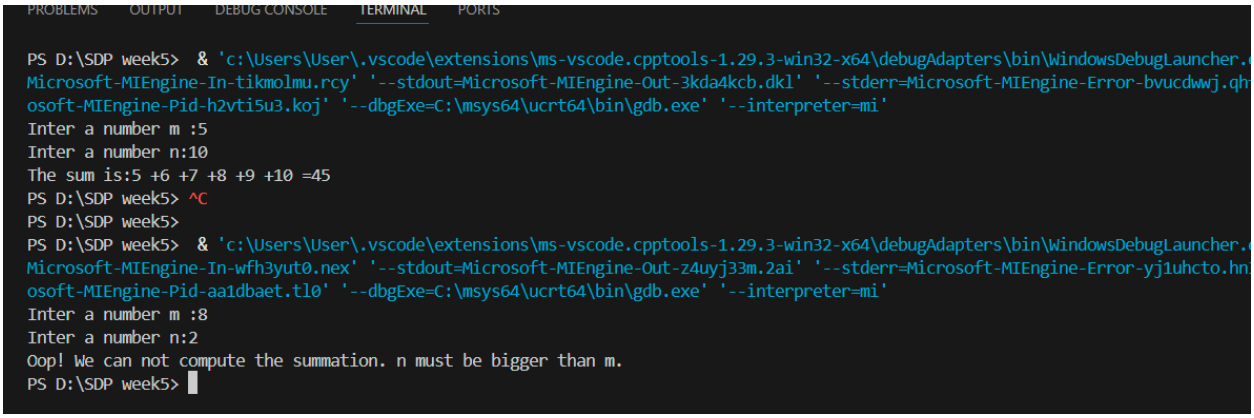
```
printf("=%d\n", sum);
```

- After the loop, prints the final result of the summation.

```
} else {
   printf("Oop! We can not compute the summation. n must be bigger than m.");
}
```

- If n < m, shows an error message.

Result:



Problem4: Write a program to ask a user to input a number. Keep asking the user to input another number if yet not input the number 0. When the user input 0, the program stops and display total summation of all input numbers.

Examples:

INPUT:

Enter a number: 5
Enter a number: 10
Enter a number: 2
Enter a number: 1
Enter a number: -1
Enter a number: 0

Output:

The sum of all input numbers is 17

.....

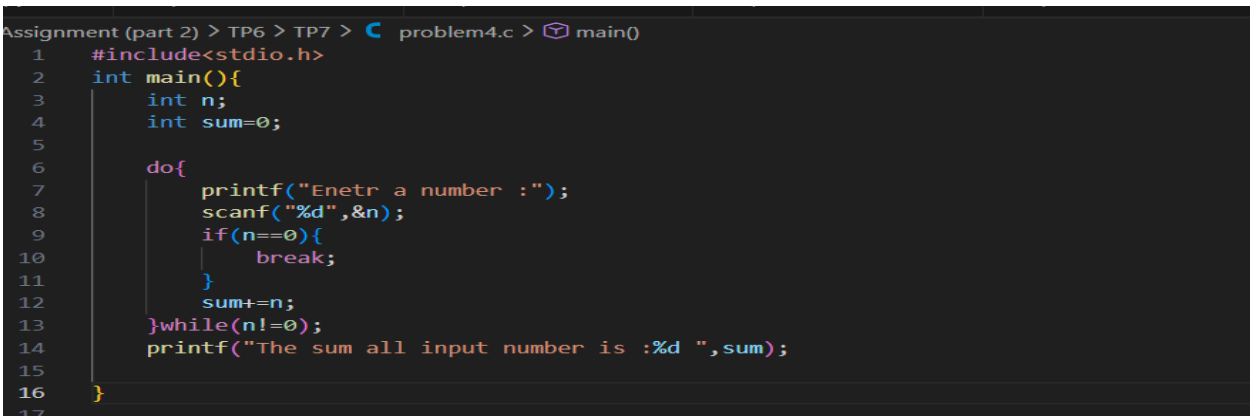
INPUT:

Enter a number: 20
Enter a number: 0

Output:

The sum of all input numbers is 20.

Code:



Explanation

```
int n;
int sum = 0;
```

- n stores each number the user enters.
- sum keeps track of the total, starting at 0.

```
do {
   printf("Enetr a number :");
   scanf("%d", &n);
```

- Prompts the user to enter a number.
- Reads the number into n.

```
   if(n == 0) {
      break;
   }
```

- If the user enters 0, the loop **breaks immediately** (stops).
- This is the **exit condition**.

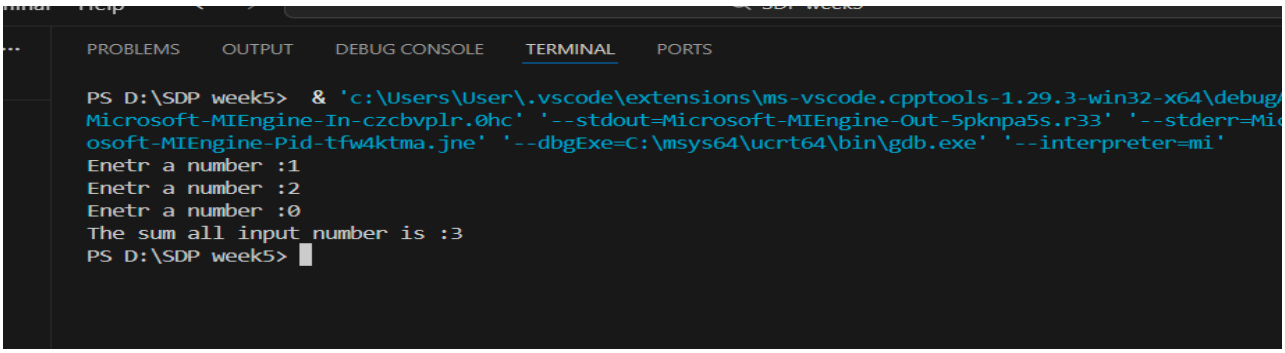
```
   sum += n;
} while(n != 0);
```

- If the number isn't 0, it gets added to sum.
- The loop continues until n == 0.

```
printf("The sum all input number is :%d ", sum);
```

- After the loop ends, prints the total sum of all numbers entered.

Result



Problem5: Write a program to ask a user for an integer number n less than 10000. The program tells the user how many digits that this number has. **Hint**: Use a DO WHILE loop. Divide the number by 10 until reaching 0. Each time, try to increase a counter variable which represents the number of digits.

Examples:

INPUT:

Enter a number n: 552

Output:

This number 552 has 3 digits.

.....

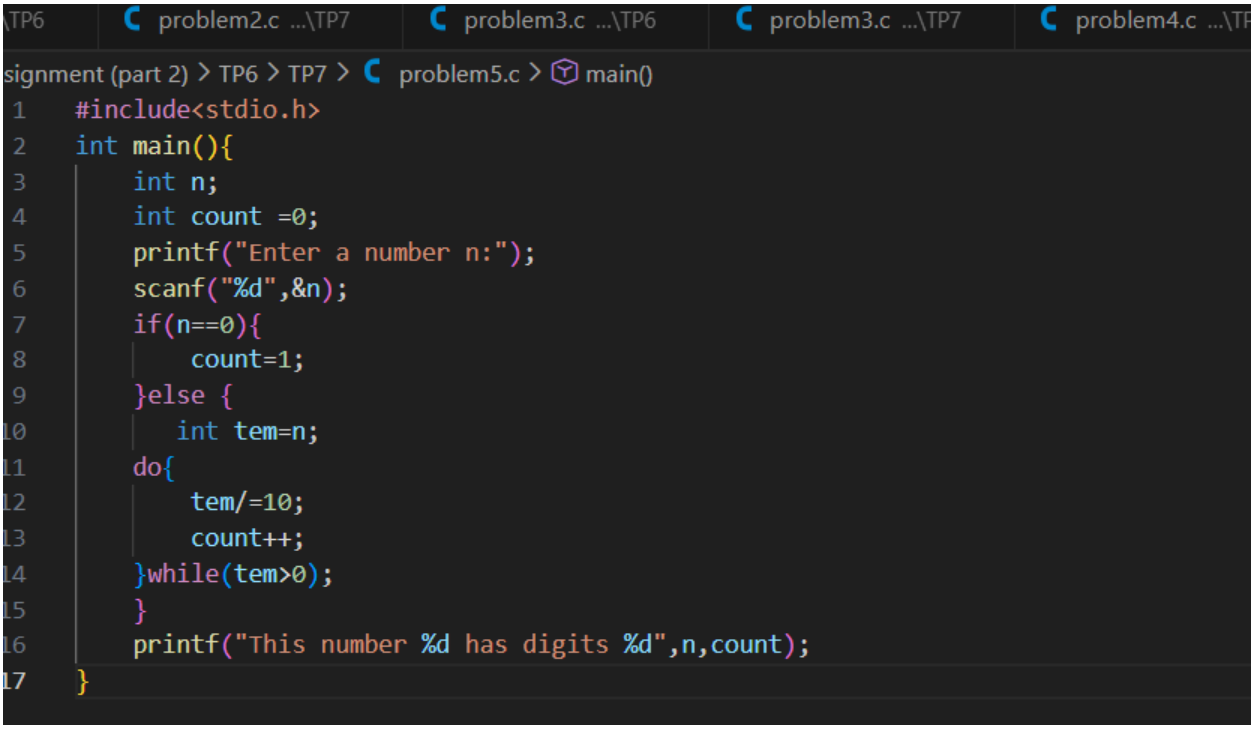
INPUT:

Enter a number n: 20021

Output:

This number 20021 has 5 digits.

Code:



Explanation

```
int n;
int count = 0;
```

- n stores the number entered by the user.
- count will store the number of digits.

```
printf("Enter a number n:");
scanf("%d", &n);
```

- Prompts the user to input a number and stores it in n.

=

```
if(n == 0) {
   count = 1;
}
```

- Special case: if the user enters 0, the program sets count = 1 because 0 has **one digit**.

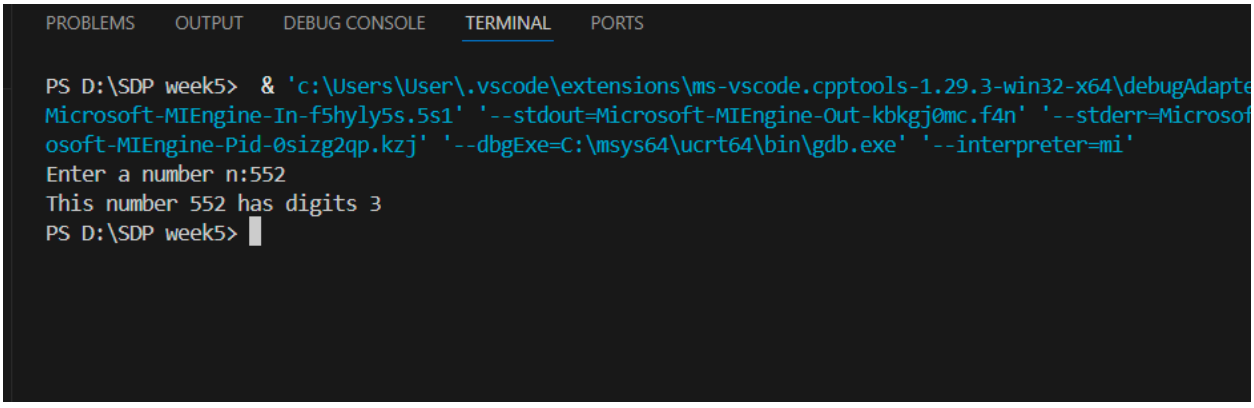
```
else {
   int tem = n;
   do {
      tem /= 10;   // remove the last digit
      count++;     // increase digit count
   } while(tem > 0);
}
```

- For any non-zero number:
  - A temporary variable tem is set equal to n.
  - In each loop iteration:
    - tem /= 10 removes the last digit (e.g., 123 → 12 → 1 → 0).
    - count++ increases the digit counter.
  - The loop continues until tem becomes 0.

```
printf("This number %d has digits %d", n, count);
```

- Prints the original number and the total digit count.

Result:



Problem6: Generate one random number. Then create a simple number guessing game where the user has to guess a number between 1 and 10. Use a while loop to continue prompting the user until they guess correctly. The program stops when the user guesses it correctly and tells the user how many that he/she has tried guessing.

Hint:
-The program randoms a number between 1 and 15. The user input a number to guess. In your program, you add a logic that when the user input is the same as the randomized number, the program stops.
-In order to generate a random number in C programming, we use the **rand()** function. To get the different number when random again, the **srand(time(0))** is used. You can find a simple code for generating a random number in the range of [1, 20].

```c
#include<stdio.h>
#include<time.h>
main(){
    srand(time(0));
    int start=1, end=20;
    int n;

    n = rand()%end + start;
    printf("%d ",n);
}
```

Example: Assume that the random number generated from the program is 10.

```
Enter a guess number between 1-15: 8
        Not correct! Try to guess again!

Enter a guess number between 1-15: 5
        Not correct! Try to guess again!

Enter a guess number between 1-15: 15
        Not correct! Try to guess again!

Enter a guess number between 1-15: 10
       *** Congratulations! Your guess number is correct!
You have try 4 time to get it right!

Process returned 0 (0x0)   execution time : 15.957 s
Press any key to continue.
```

Code:



Explanation

int randomNumber, guess, attempts = 0;

- randomNumber → stores the secret number.
- guess → stores the user's input.
- attempts → counts how many guesses the user makes.

srand(time(0));
randomNumber = rand() % 10 + 1;

- srand(time(0)) seeds the random number generator with the current time, so the number changes each run.
- rand() % 10 + 1 generates a random integer between 1 and 10:
  - % 10 gives a remainder between 0–9.
  - + 1 shifts it to 1–10.

printf("Guess the number between 1 and 10:\n");

- Displays instructions to the user.

```
do {
   printf("Enter your guess: ");
   scanf("%d", &guess);
   attempts++;
```

- Prompts the user for a guess.
- Reads the guess into guess.
- Increases attempts by 1.

```
   if (guess == randomNumber) {
      printf("Correct! You guessed it in %d attempt(s).\n", attempts);
      break;
   } else {
      printf("Wrong guess. Try again!\n");
   }
} while (guess != randomNumber);
```

- If the guess is correct:
  - Prints success message with the number of attempts.
  - break exits the loop.
- If wrong:
  - Prints "Wrong guess" and continues.
- The loop repeats until the guess matches the random number.

```
return 0;
```

- Ends the program successfully.

Result:

