```c
//  Basic Practice.h
//
//  Created by Li Cheng-En on 2018.
//  Copyright © 2018 Li Cheng-En. All rights reserved.
//
//  ---------------------<Outline>---------------------
//  Declaration                                  <Row 25>
//  For-loop                                     <Row 44>
//  Double For-loop & If                         <Row 59>
//  Break And Skip The Loop                      <Row 81>
//  Array with Single Dimension                  <Row 99>
//  Array                                       <Row 117>
//  Ask User To Type The Value(1)               <Row 162>
//  Ask User To Type The Value(2)               <Row 181>
//  Search The String                           <Row 203>

#ifndef Basic_Practice_h
#define Basic_Practice_h

// I need to put my functions I want to use in seperated header file, or I
 would implicitely declare the function which is not valid. I also could
 type this code before the function of main, but it is a little bit wierd.
 To get more information about this issue, please visit the website at
 "https://stackoverflow.com/questions/15850042/xcode-warning-implicit-
 declaration-of-function-is-invalid-in-c99".



// Declaration
int practiceDeclareSomeVariables(void) {
    int practiceInteger = 0;
    double practiceDouble = 1.0;
    char practiceCharacter[] = "I want to show a string instead of a
     character!";

    printf("%d \n", practiceInteger);
    printf("%f \n", practiceDouble);
    printf("%s \n", practiceCharacter);
    printf("\n");

    return 0;
}
// 1. In C, the type of data included integer, float and character.
// 2. If I want to declare a string, remember that the string is a kind of
 matrix of characters, so I need to add "[]".
// 3. There is no any function I could use to get the type of variable
 directly in c.



// For-loop
int aggregateIntegersFunction(void) {

    int integerUsedToAggregate = 0;
    for (int i = 1; i <= 100; i++) {
        integerUsedToAggregate = integerUsedToAggregate + i;
    }

    return integerUsedToAggregate;
}
// We need to declare the type of "i" in the for-loop, <= means "no larger
 than", and "i++" means that the i would increase gradually.



// Double For-loop & If
int ninetyNineMultiplicationTable(void) {

    for (int j = 2; j <= 9; j++) {
        for (int i = 1; i <= 9; i++) {
            if (i == 3 || i == 6) {
                printf("%d * %d = %d \n", j, i, i*j);
            } else if (i == 9) {
                printf("%d * %d = %d \n", j, i, i*j);
                printf("\n");
            } else {
                printf("%d * %d = %d \t  ", j, i, i*j);
            }
        }
    }

    return 0;
}
// When we want to print several integers inside a text, we need to type the
 structure of the text first, and then we could type the integers orderly.



// Break And Skip The Loop
int showTheSeriesOfOddNumber(void) {
    for (int i = 1; i < 100; i++) {
        if (i % 2 == 0) {
            continue;
        } if ( i == 11) {
            break;
        } else {
            printf("%d \n", i);
        }
    }

    printf("\n");
    return 0;
}



// Array with Single Dimension
int arrayWithSingleDimension(void) {
    int practiceDoubleArrayWithSingleDimension[] = {1, 2, 3};
    char practiceCharacterArrayWithSingleDimension[] = {"Cindy", "John"};
    char *practiceStringArrayWithSingleDimension[] = {"Cindy", "John",
     "Tina"};

    printf("%d \n", practiceDoubleArrayWithSingleDimension[1]);
    printf("%c \n", practiceCharacterArrayWithSingleDimension[1]);
    printf("%s \n", practiceStringArrayWithSingleDimension[1]);

    return 0;
}
// 1. I could create an array by adding "[]" in the end of the name of the
 variables. Remember that you need to add "*" before the name of the
 variables when you want to declare an array of string.
// 2. "*" means pointer of variable, please refer to the "Hard Practice.h"
 file to learn some knowledge about it.
// 3. To get more information about the array of character and that of
 string, please visit the website at "https://stackoverflow.com/questions/
 8732325/how-to-declare-strings-in-c".



// Array
int createArraysWithTwoDimension(void) {
    int rowOfMatrix = 3;
    int columnOfMatrix = 3;
    int identityMatrix[rowOfMatrix][columnOfMatrix];

    printf("\n");
    printf("Identity Matrix: \n");

    for (int row = 0; row < rowOfMatrix; row++) {
        for (int column = 0; column < columnOfMatrix; column++) {
            if (row == column) {
                identityMatrix[row][column] = 1;
            } else {
                identityMatrix[row][column] = 0;
            }
            printf("%d \t", identityMatrix[row][column]);
        }
        printf("\n");
    }
    printf("\n");


    int anotherMatrixWithTwoDimension[3][4] = {
        1, 2, 3, 4,
        5, 6, 7, 8,
        9, 10, 11, 12
    };

    printf("Another Matrix with three row and four column. \n");
    for (int row = 0; row < 3; row++) {
        for (int column = 0; column < 4; column++) {
            printf("%d \t", anotherMatrixWithTwoDimension[row][column]);
```

```c
        }
        printf("\n");
    }

    return 0;
}
// 1. We could add [number of row][number of column] behind the name of the
 matrix to declare it.
// 2. When we want to print out the matrix, we need to print out the element
 of the matrix respectively.
// 3. We could use "\t" to indent the string.



// Ask User To Type The Value (1)
int requireUserToGiveDataOfNameAndAge(void) {
    int requiredDouble;
    char *requireNameString[50];

    printf("Please enter your name without dash or space: ");
    scanf("%s", requireNameString[0]);
    printf("Please enter the number of your age: ");
    scanf("%d", &requiredDouble);

    printf("%s are %d years old! \n", requireNameString[0], requiredDouble);

    return 0;
}
// 1. Function "scanf(type, variable)" is used to ask user to type the data
 manually.
// 2. We need to add "&" before the name of the variables when we want to
 change the data of double variable on the scanf() function, or the value
 could not be edited.



// Ask User To Type The Value (2)
int AskUsersToTypeTheValue(void) {
    char TheWordIWantToSayToMyFriend[50];
    char TheSentenceIWantToSayToMyFriend[100];

    fgets(TheSentenceIWantToSayToMyFriend,
     (sizeof(TheSentenceIWantToSayToMyFriend) /
     sizeof(TheSentenceIWantToSayToMyFriend[0])), stdin);
    printf("%s", TheSentenceIWantToSayToMyFriend);

    scanf("%s", TheWordIWantToSayToMyFriend);
    printf("%s \n", TheWordIWantToSayToMyFriend);
    //gets(TheSentenceIWantToSayToMyFriend);     Error would occure if you
     run this code.

    return 0;
}
// 1. fgets() function could be used to ask user to type something.
// 2. scanf() function could be used to ask users to type something, too.
 However, the disadvantages of it were more than that of fgets() function.
 First of all, it is less secure. Moreover, it only would show "one" words &
 string & number; it would see space & return as a sign of the end of the
 function. Thus, I had better avoid to use scanf() function.
// 3. I had better not to use gets() function, the reason is that if user
 type too much things or I assume that the users would type too more things,
 I would face warnings, and the program would end immediately.
// 4. I had better not to put the fgets() function behind the scanf()
 function, the fact is that the output of scanf() function would involve a
 "\n", which may disable the fgets() behind the scanf()/
// 5. To get more information about the suspension of program because of the
 relative order of fgets() and scanf() function, please visit the website at
 https://stackoverflow.com/questions/4929338/problem-with-scanf-and-fgets or
 https://www.ptt.cc/bbs/C_and_CPP/M.1310481378.A.137.html



// Search The String
int searchTheStringInArray(void) {
    char sourceOfStringArray[] = "ABCDEFGHIJ";
    char theStringIWantToCompare[] = "JACKY";
    char anotherStringIWantToCompare[] = "ABCDEFGHIJ";

    printf("result: %d \n", strcmp(sourceOfStringArray,
     theStringIWantToCompare));
    printf("result: %d \n", strcmp(sourceOfStringArray,
     anotherStringIWantToCompare));


    char theStringIWantToSearch = 'C';
    printf("%lu \n", strchr(sourceOfStringArray, theStringIWantToSearch));

    for (int i = 0; i < strlen(sourceOfStringArray); i++) {
        if (theStringIWantToSearch == sourceOfStringArray[i]) {
            printf("%d", i + 1);
        }
    } /* It could be used to mock the rationale of strstr() function. */

    return 0;
}
// 1. strcmp() function could be used to compare two strings. If these two
 strings are the same, then the output would be 0, or the output would be
 some positive numbers or negative numbers.
// 2. strchr() function could be used to track whether there's any the same
 element in the original longer string. If the answer is yes, then the
 output would be the address of that element in that string, or the output
 would be null.
// 3. It seems that strstr() function could be used to check whether there's
 any the same element in the original longer string, too. If the answer is
 yes, then the output of this function would be the order of the same
 element in that original string. However, it is a little hard for me to use
 this function. Therefore, I used another way, which is the combination of
 for-loop and if, to substitute strstr() function.
```

```
#endif /* Basic_Practice_h */
```