16. Break and Continue

- In C programming, break and continue are control flow statements used within loops and switch statements to alter the normal flow of execution.

Break

- The break statement is used to terminate the loop or switch statement immediately. When a break is encountered, the control exits the loop or switch and continues with the statement following the loop or switch.
- Example:

```
int main() {
    for(int i = 1; i <= 10; i++) {
        if(i == 5) {
            break; // Exit the loop when i is 5
        }
        printf("%d\n", i);
    }
    return 0;
}
```

Continue

- The continue statement skips the current iteration of the loop and proceeds with the next iteration. When continue is encountered, the rest of the code inside the loop for the current iteration is skipped, and the loop proceeds with the next iteration.
- Example:

```
int main() {
    for(int i = 1; i <= 10; i++) {
        if(i == 5) {
            continue; // Skip the current iteration when i is 5
        }
        printf("%d\n", i);
    }
    return 0;
}
```

17. Array

- An array is a variable that can store multiple values.
- Declare array: dataType arrayName[arraySize];
- Initialization of array
    - int mark[5] = {9, 1, 8, 7, 3};
    - int mark[] = {9, 1, 8, 7, 3};

12 mark[0]	3 mark[1]	0 mark[2]	2 mark[3]	1 mark[4]

- Array has 0 as the first index, not 1.
- If the size of an array is n, to access the last element, the n-1 index is used.
- Size of the above array is 5.
- Change Value of Array
    - mark[2] = -1 // make the value of the third element to -1
    - Mark[4] = 2 // make the value of last element to 2
- Access array element
    - secondElement = mark[1]
- Example

```
// Program to take 5 values from the user and store them in an array
// Print the elements stored in the array

int main() {

  int array[5];

  printf("Enter 5 integers: ");

  // taking input and storing it in an array
  for(int i = 0; i < 5; ++i) {
    scanf("%d", &array[i]);
  }

  printf("Displaying integers: ");

  // printing elements of an array
  for(int i = 0; i < 5; ++i) {
    printf("%d\n", values[i]);
  }
  return 0;
}
```

Multidimensional array

- array of arrays that stores homogeneous data in tabular form.
- Declaration: data_type array_name[size1][size2].
- Initialization of 2d array
  - int array[2][3] = {{1, 3, 0}, {-1, 5, 9}};
  - int array[][3] = {{1, 3, 0}, {-1, 5, 9}};
  - int array[2][3] = {1, 3, 0, -1, 5, 9};
- Change value in array
  - array[1][2] = 99; // update the value at row1, column 2
- float x[3][4] // 3 rows and 4 columns, total 12 elements

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

- Access array element
  - selectedElement = x[1][2]
- Example:

```
int main() {
    // Define a 2x3 2D array and initialise it with values
    int array[2][3] = {
        {1, 2, 3},
        {4, 5, 6}
    };

    // Print the values of the 2D array
    printf("2x3 2D array values:\n");
    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 3; j++) {
            printf("%d ", array[i][j]);
        }
        printf("\n"); // New line after each row
    }

    return 0;
}
```

18. Pointer

- These are special variables that are used to store addresses rather than values.
- Syntax:
  - int* q; / int *p1; / int * p2;
- Assigning addresses to pointers

```
int* pc, c;
c = 5;
pc = &c;
// Here, 5 is assigned to the c variable. And, the address of c is assigned to the pc pointer.
```

- Get value pointed by pointers

```
int* pointer, c;
c = 5;
pc = &c;
printf("%d", *pc);   // Output: 5
// the address of c is assigned to the pc pointer. To get the value stored in that address, we used *pc
```

- Changing value

```
int* pc, c;
c = 5;
pc = &c;
c = 1;
printf("%d", c);    // Output: 1
printf("%d", *pc);  // Output: 1
```

- Example

```
int* pc, c;
c = 22;
printf("Address of c: %p\n", &c);
printf("Value of c: %d\n\n", c);  // 22

pc = &c;
printf("Address of pointer pc: %p\n", pc);
printf("Content of pointer pc: %d\n\n", *pc); // 22

c = 11;
printf("Address of pointer pc: %p\n", pc);
printf("Content of pointer pc: %d\n\n", *pc); // 11

*pc = 2;
printf("Address of c: %p\n", &c);
printf("Value of c: %d\n\n", c); // 2
```

19. Some notes

- C language is widely used for system programming on Unix-based operating systems.
- C99 introduced single-line comments with //, making it easier to write and manage comments within code.
- Type casting allows you to convert a variable from one data type to another.
- Function pointers are pointers that point to functions instead of data.
- C99 introduced Variable-Length Arrays (VLA), where the size of the array can be determined at runtime.
- There are 32 keywords/reserved words in the original C89/C90 standard
- The file extension typically used for C programming files is `.c`