Functions

Tai, Wei Hsuan

week 7

Announcement

Course Materials

Outline

Recap

Basic I/O

```cpp
#include <iostream>
using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    cout << a + b << endl;
    return 0;
}
```

Selection Statement

```cpp
#include <iostream>
using namespace std;

int main() {
    int a;
    cin >> a;
    if (a > 0) {
        cout << "Positive" << endl;
    } else if (a < 0) {
        cout << "Negative" << endl;
    } else {
        cout << "Zero" << endl;
    }
    return 0;
}
```

While Loop

```cpp
#include <iostream>
using namespace std;

int main() {
    int i = 0;
    while (i < 10) {
        cout << i << " ";
        i++;
    }
    cout << endl;
    return 0;
}
```

For Loop

```cpp
#include <iostream>
using namespace std;

int main() {
    for (int i = 0; i < 10; i++) {
        cout << i << " ";
    }
    cout << endl;
    return 0;
}
```

Nested Loop

```cpp
#include <iostream>
using namespace std;

int main() {
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            cout << "(" << i << ", " << j << ") ";
        }
        cout << endl;
    }
    return 0;
}
```

1D Array

```cpp
#include <iostream>
using namespace std;

int main() {
    int arr[5];
    for (int i = 0; i < 5; i++) {
        cin >> arr[i];
    }
    cout << endl;
    return 0;
}
```

2D Array

Basic Concept

Sometimes we need to store data in a table format, for instance, a chessboard or a matrix.
To achieve this, we can use a 2D array.
The concept of 2D array is easy. We can generate an array with any data type, how about an array of arrays?

Declaration and Initialization

```cpp
int matrix[3][4]; // Declare a 2D array with 3 rows and
↪ 4 columns
for(int i = 0; i < 3; i++) {
    for(int j = 0; j < 4; j++) {
        cin >>matrix[i][j];
    }
}
for(int i = 0; i < 3; i++) {
    for(int j = 0; j < 4; j++) {
        cout << matrix[i][j] << " ";
    }
    cout << endl;
}
```

The utilization of a nD array is similar to a 1D array, just with more indices.

Functions

Basic Concept

Sometimes our code may be very long and complicated. To make them more organized and reusable, we can use functions.
A function is a block of code that performs a specific task. It can take inputs, process them, and return an output.
While developing a project, we often split our code into multiple functions to shorten the main program. This makes our code more readable and easier to debug.

An example: Minesweeper

In the game Minesweeper, we have multiple repeated tasks. For example, checking the number of mines around a cell, output the game board, and handling user input, ... and so on.

Components of a function

A function consists of several components:

- Function Declaration (Prototype)
- Function Definition
- Function Call

Function Declaration

A function declaration tells the compiler about a function’s name, return type, and parameters. It is usually placed at the beginning of the program or in a header file. Following is a simple function declaration:

```c
int add(int a, int b);
```

This declares a function named `add` that takes two integer parameters and returns an integer.

Function Definition

A function definition provides the actual body of the function. It specifies what the function does. After some complex calculations, it returns the result to the caller. No matter what you did in this function, it would only return the object you put after `return`.
Here is the definition of the `add` function:

```cpp
int add(int a, int b) {
    return a + b;
}
```

This function takes two integers as input and returns their sum. Now you know why there's a `return 0` statement in the `main` function! (Some big projects utilize the whole cpp file as a function, so you need to return an integer to indicate the program's exit status.)

Function Call

A function call is used to execute a function. When a function is called, the program control is transferred to the function, and after the function execution is completed, the control returns to the point where the function was called.

Here is how to call the `add` function:

```cpp
            int result = add(5, 3);
            cout << "The sum is: " << result << endl;
```

This calls the `add` function with arguments 5 and 3, and stores the returned value in the variable `result`.

An example of function

```cpp
#include <iostream>
using namespace std;

int add(int a, int b) {
    return a + b;
}

int main() {
    int result = add(5, 3);
    cout << "The sum is: " << result << endl;
    return 0;
}
```