Review

# The compiler

To compile C++ program, we use g++ compiler.

# Variables

## 2.1 variable Name:

Name convention: a variable name can contain letter, digits, and underscore. A variable name **can’t** begin with a digits, but **can** begin with underscore.

## 2.2 Data Type:

int, double, float, bool, char, sting

## 2.3 Declaration:

give datatype and identifier.

## 2.4 Initialization:

the first time time we assign a variable a value

# Operators and operations

Operators: + - \* / %. Pay attention to / operator, For example:

int z;

int x = 8;

int y = 3;

z = x/y; // **note here: z is 2, not 2.666**

Pay attention to pre-operation and post operation:

i++ is post-operation, which mean we first use the variable i and then increment i. For example:

int i = 5;

int y;

y = i++;

cout << y << endl; // **Here y is 5**

++i is pre-operation, which means we first increment variable i and then use it. For example:

int i = 5;

int y;

y = ++i;

cout << y << endl; // **Her y is 6**

# Truth table:

Truth table:

/\*

A B. A or B. A||B. A|B

F. F. F

F. T. T

T. F. T

T. T. T

\*/

Truth table:

/\*

A B. A and B. A &&B. A&B

F. F. F

F. T. F

T. F. F

T. T. T

\*/

Not Unary operation

One operand A

Truth table:

/\*

A not A !A (exclamation point). ~A

F. T

T. F

\*/

# If statement

If statement **selects** one block of code to execute every time. The selected code is determined by the truth value of the predicate. The code syntax is like this:

if(predicate){

Code Block 1;}

else{

Code Block2;}

# switch statement

switch statement also selects a block of code to execute every time. The syntax is like that:

switch(val){

case 1:

code1;

break;

case 2:

code2;

break;

default:

code3;

}

Note that: **val** in switch statement **can be an integer or a char**, but **can’t be a double or float**

# For loop

The syntax of for loop is like that:

for(int i = 0; i<n; i++){

code;

}

An example is that:

for(int i = 0; i<3; i++){

cout << “I is: “ << i << endl

}

Note that the execution order of the above code:

1. int i=0;
2. evaluate i<3. Here i is 0, 0 = 3 is true
3. cout 0
4. execute i++, which make i become 1
5. evaluate i<3. Here i is 1, 1 = 3 is true
6. cout 1
7. execute i++, which make i become 2
8. evaluate i<3. Here i is 2, 2 = 3 is true
9. Cout 2
10. Execute i++, which make I become 3
11. evaluate i<3. Here i is 3, 3 = 3 is false. stop

# while loop

The syntax of while loop is:

while(predicate){

code;}

An example is as following:

int i = 0;

while (i<3){

cout << i<< endl;

i++;

}

Note that the execution order of the above code:

1. int i=0;
2. evaluate i<3. Here i is 0, 0 = 3 is true
3. cout 0
4. execute i++, which make i become 1
5. evaluate i<3. Here i is 1, 1 = 3 is true
6. cout 1
7. execute i++, which make i become 2
8. evaluate i<3. Here i is 2, 2 = 3 is true
9. Cout 2
10. Execute i++, which make I become 3
11. evaluate i<3. Here i is 3, 3 = 3 is false. stop

You can see while loop has the same execution order as for loop. So while loop and for loop are interchangeable.

# Function

## 9.1 Function overview

A screenshot of a computer program

Description automatically generated

## 9.2 Function Type

1. Void: The function does some operation, such as print some characters in the screen, but the function has no return value.

2. int/float/double: The function return a value with int/float/double. If a function has a return value, we can assign the value to other variable.

# Pass by value and pass by reference

“Pass by value” means we copy the value of a variable and pass the copied value to the function. The original variable and the actual parameter for the function are two independent variable. Any operation in the function **does not** affect the original variable.

“Pass by reference” means we pass the address of a variable into the function. The actual parameter in the function and the original variable are actually the same variable. You can think the actual parameter in the function are just a alias for the original variable.

For example:

//define a function with passing by reference

void function2(int &z){ //&z get the address of actual parameter

z = 2 \* z;

}

//call the function;

int x = 4

function2(x); // in function call when we pass a variable into the function,

// the function n access the address of the variable,

// it can modify the variable itself

# Struct

structure in c++ is form of aggregation. We have 3 steps to use a struct: declare a struct, create instances of the struct and access/use the attributes of an instance

For example:

// (1) declare a struct named “student”

struct student

{

string name;

int age;

float gpa;

float height;

float weight;

};

The “struct” is like a package and the package has a name student. Defining a “struct student” is just like declaring a new type with five things associate with it.

After declare it, we can create instance of this type.

// (2)create an instance of struct student

student \_s;

And then, we can access the attribute of the instance \_s

// (3) here we print the “age” attribute of instance “\_s”

cout << "\_s.age: " << \_s.age << endl;

# Class

Class is a formal aggregation type in C++.

## Difference between struct and class

The only difference between class and structure is the default visibility is different. **In struct, all the struct members are public; in class, the default visibility specifier is private**. To access the private members in class, we can define some public method to get or set the private members. These public methods are often called getter or setter/mutator.

## 12.2 class constructor

Class constructors should have exactly the same name with the class name and class constructor does not need to have return type.

A class can have more than one constructor, but these constructors must have different argument lists.