Programming Fundamentals

Lecture 03, 04 Variables , Data Types and Calculations

Variables

- Retain a number say 4 in your mental memory,
- Retain the number 3 at the same time.
- You have just stored two different values in your memory.
- Now, add 1 to the first number, I said,
- Now you have in memory 5 & 3
- Subtract second number from first one. The result is 2
- The whole process that you have just done with your mental memory is similar to what a computer can do with two variables.
- The same process can be expressed in C++ with the following instruction set:

```
a = 4
b = 3;
a = a + 1;
result = a - b;
```

we can define a **variable** as a portion of memory to store a determined value

Each variable needs an **identifier** that distinguishes it from the others, for example, a, b and result are variable identifiers

Using Variables

- To do more programming than only printing a string as output, we need ability to store data items in a program
- This facility is provided by variables
- A **variable** is an area in memory that is identified by a name that we specify, and we can use to store an item of data of a particular type

Using Variables

- Specifying a variable requires two things:
  1. Type of data to be stored
  2. Name

Variable Names

- Name can consist of any combination of:
  - Upper and lower case letters
  - Underscores
  - Digits
- But a variable name cannot begin with a digit
- This means 8Balls and 7UP are not valid variable names

Variable Names

- C++ is a case sensitive language
- This means **myName** and **myname** are two different variables
- Variable names starting with underscore are reserved for use within the libraries
- Variable names should be indicative of the kind of data that they hold, For instance,
- **my_name** is going to mean a whole a lot more than **mn**

Naming Conventions

- CamelCase
  - structs and classes Example; MyName
- camelCase
  - Functions & variables; Example; myName
- c-style
  - Variables; Example; my_name
- It can be useful to use prefixes for certain types of dat a to remind you what they are: for instance
  - For pointers p_name
  - For static variables s_name
  - For global variables g_name

Variable Names

- Variable name should not match with any of the keyword
- Following is the list of standard reserved keywords

asm, auto, bool, break, case, catch, char, class, const, const_cast, continue , default, delete,do, double, dynamic_cast, else, enum, explicit, export, exter n, false, float, for, friend, goto, if, inline, int, long, mutable, namespace, new, operator, private, protected, public, register, reinterpret_cast, return, short, s igned, sizeof, static, static_cast, struct, switch, template, this, throw, true, try , typedef, typeid, typename, union, unsigned, using, virtual, void,volatile, wc har_t, while

Basic Data Types

- C++ is a strongly typed language
  - Every data item in your program has a type associated with it

Basic Variable Types

Type	Memory size per variable (in bytes)
char	1
short	2
int	4
long	4
bool	1
float	4
double	8
long double	8

Variable Declaration & Definition

- Variable Declaration

  int apples; **//** It introduces a name i.e. **apples**

- Variable Definition

  int apples = 10;

Exercise 1

- Define two numbers of type int
- Subtract first from second number
- Display the result

Assignment Operator (=)

- Used to store result of a calculation in a variable.
- For example:
  - `int total = 0;`
  - `var1 = var2 = 10;`
  - `total = var1 + var2;`
  - `total = (var1 = 10) + (var2 = 10);`

The assignment operator is right associative, so in 2nd statement, value 10 will be assigned to var2 and then to var1

op= Assignment Operator

- **op=** assignment operator is so called, because it is composed of an operator and an equal sign

- For example:
  - sum += 2; // sum = sum + 2;
  - ans *= var1; // ans = ans * var1;

Increment and Decrement

- Increment operator (++)
- Decrement operator (--)
  - For example:
    - count = count +1; // add 1 to count
    - count += 1; // add 1 to count
    - ++count ; // add 1 to count
  - *What is the difference between **++count** and **count++** ?*

const keyword

- Sometimes we have values in our programs which do not change e.g. number of days in March, value of π
- Also we want to avoid "*magic numbers*" in our program
  - Magic numbers are numbers that appear directly in the code without an obvious reason
    - e.g. a = a + 5.5
  - By avoiding magic numbers, maintenance of programs become easy
    - e.g. commission rate changes very often, then you have to change the number everywhere in the program
  - Descriptively name your variables
  - Also it makes easier to spot the use of a particular number and differentiate between numbers with the same value that mean different things

const keyword

- If you use a variable to hold a constant, then you want to prevent any accidental modifications
- Use the **const** keyword
  - **const** int maxNo = 100**;**
- You can define any kind of variable as **const**

Floating Point Data Types

- float
- double

- We can specify a floating point variable using the keyword float or double

- **const** double PI = **3.14159265358979323846;**

Exercise02

- Calculate and display the area and circumfer ence of a Circle using the following formulas
- Area = πr^2
- Circumference = $2 \pi r$

Characters

- We can write a character literal as the character that we require, placed between the pair of single quotes. Like ‘z’, ‘3’ or ‘?’
  - char letter = ‘A’
- This letter will have the decimal value 65 (ASCII value)
  - char letter = 65;
- We can operate on letter as an integer
  - Letter += 2;

Assignments and Different Types

- If the type of an expression on the right of the assignment operator is different from that of the variable on the left,
- The result evaluating expression on the right hand side will automatically be converted to the type of the variable on the left before it is stored
- For example,
  - **double** root = 1.732;
  - **int** value = root;
- Conversion of the value of *root* (double) to int results in **1** being stored in variable *value*

Explicit Casting

- To cast the value of an expression to a given type, we write the cast in the form
- ***static_cast*** <the type to convert to> (expression)
- The keyword ***static_cast*** reflects the fact that the cast is checked statically (when our program is compiled)
- For example
  - **double** value1 = 10.5;
  - **int** value2 = ***static_cast*** <**int**> (value1);

The Lifetime of a Variable

- All variables have a finite lifetime when a program executes
- They come into existence from the point at which they are declared and then, at some point, they disappear
- How long a variable lasts is determined by a property called its ***storage duration***
- Three kinds of storage duration
  - Automatic storage duration
  - Static storage duration
  - Dynamic storage duration (to be discussed later)
- Another property that a variable has is its **scope**
  - You can not access a variable outside of its scope

Automatic Variables

- The variables that we have declared so far have been declared within a block – that is between a pair of curly braces { }
- They have **local** or **block scope**
- The automatic variable is born when it is declared and automatically destroys at the end of the block containing the declaration

Global Variables

- **Static Variables**
- If we want to have a variable that's defined and accessible locally within a block, but which continues to exists even after exiting the block in which it is declared, we need to give it **static** storage
- They exists till the end of the program's execution
- The **static** specifier is used to do so
- **Static** variables are automatically initialized to 0 unlike automatic variables
- For example,
  - ***Static int*** count;

Global Variables

- **External Variables**
- Within a program there are many files, if we want to access global variables from one source file that is declared in another source file, we can make use of ***extern*** keyword

Summary

- Numeric and character constants are called **literals**
- Variable names in C++ are case sensitive
- Names that begin with Underscore followed by a capital letter are reserved for use within standard library
- The name and type of a variable are appear in a declaration statement
- Variables may be given initial value when they are declared, and it's a good programming practice
- Explicit conversion of a variable from one type to another can be done using ***static_cast***
- The ***extern*** keyword allows to access a variable declared in another file