IT1010 - Note for Lecture 02

Variables in C

What is a variable in computer program?

A variable in a computer program is a named location in memory that can store a value. Variables are used to store data that can be used throughout the program. For example, you might have a variable to store the user's name, or the number of items in a shopping cart.

How variables can be declared in C language?

Variables are declared using a special syntax that tells the compiler or interpreter what type of data the variable can store.

In C language, variables are declared using the following syntax:

data_type variable_name;

For example, the following code declares a variable named my_variable of type int:

```
int my_variable;
```

Once a variable has been declared, it can be assigned a value using the following syntax:

variable_name = value;

For example, the following code assigns the value 10 to the variable my_variable:

```
my_variable = 10;
```

Variables can also be initialized when they are declared. Initialization is the process of assigning a value to a variable at the same time that it is declared. To initialize a variable, simply add an equal sign (=) and the value to the declaration. For example, the following code declares and initializes a variable named my_variable of type int with the value 10:

```
int my_variable = 10;
```

Here are some rules to follow when declaring variables in C:

- Variable names must start with a letter or an underscore.
- Variable names cannot contain spaces or special characters.
- Variable names are case sensitive.
- A variable cannot have the same name as a function or another variable.
- A variable must be declared before it can be used.

Data Types in C

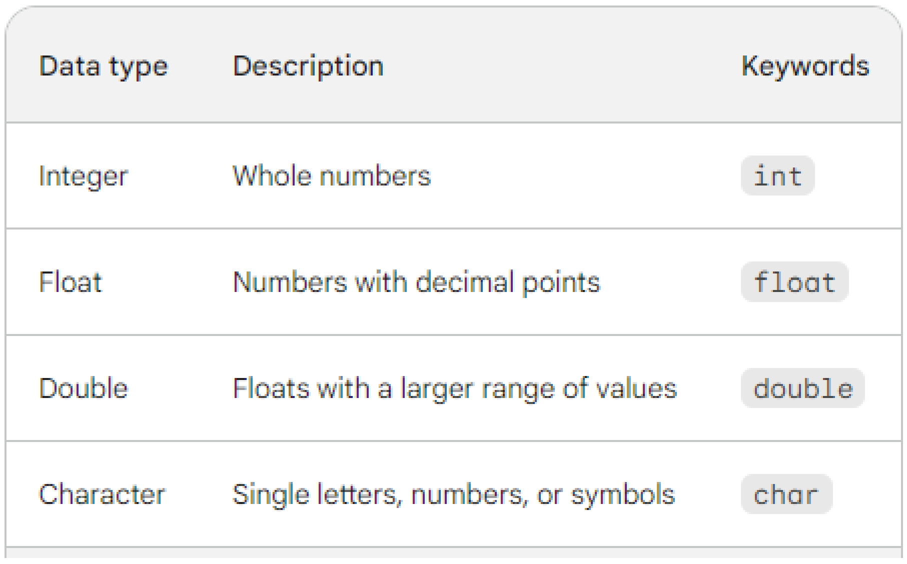
What are the available data types in C programming?

There are 4 basic data types in C programming,

- **Integer**: Integers are whole numbers, positive or negative. They can be represented by the keyword int. For example, int x = 10; declares a variable x of type int and assigns it the value 10.
- **Float**: Floats are numbers with decimal points. They can be represented by the keyword float. For example, float y = 3.14; declares a variable y of type float and assigns it the value 3.14.
- **Double**: Doubles are floats with a larger range of values. They can be represented by the keyword double. For example, double z = 1.0e10; declares a variable z of type double and assigns it the value 100,000,000,000.
- **Character**: Characters are single letters, numbers, or symbols. They can be represented by the keyword char. For example, char c = 'a'; declares a variable c of type char and assigns it the value a.

In addition to these basic data types, C also supports a number of derived data types, such as arrays, structures, and unions. These data types are created by combining basic data types in different ways.

Here is a table that summarizes the available data types in C programming:



Data type	Description	Keywords
Integer	Whole numbers	`int`
Float	Numbers with decimal points	`float`
Double	Floats with a larger range of values	`double`
Character	Single letters, numbers, or symbols	`char`

Handling output in C

How to handle output in C programming?

To handle output in C programming, you can use the printf() function. The printf() function is a library function that is used to print formatted output to the screen.

The syntax for the printf() function is as follows:

    printf("Formatted string", arguments);

For example, the following code will print the string "Hello, world!" to the screen:

```
printf("Hello, world!");
```

The Formatted string is a string that contains formatting directives. The formatting directives tell the printf() function how to format the output. The Arguments are the values that will be formatted and printed.

The following code will print the value of the variable x to the screen:

```
printf("The value of x is: %d\n", x);
```

The %d in the format string tells the printf() function to print the value of x as an integer. The \n at the end of the format string tells the printf() function to print a newline character.

The printf() function can be used to print a variety of data types, including integers, floats, doubles, and strings. For more information on the printf() function, you can refer to the C programming language documentation.

Reading input in C

How to read input in C programming?

To read input in C programming, you can use the scanf() function. The scanf() function is a library function that is used to read formatted input from the standard input stream. The syntax for the scanf() function is as follows:

    scanf("Formatted string", variables);

The Formatted string is a string that contains formatting directives. The formatting directives tell the scanf() function how to format the input. The Variables are the variables that will be assigned the input values.

For example, the following code will read an integer from the standard input stream and store it in the variable x:

```
scanf("%d", &x);
```

The %d in the format string tells the scanf() function to read an integer from the input stream. The &x tells the scanf() function to store the input value in the variable x.

The scanf() function can be used to read a variety of data types, including integers, floats, doubles, and strings. For more information on the scanf() function, you can refer to the C programming language documentation.

Simple C Program - Description

What is the basic structure of a C program?

A C program is divided into six sections:

Documentation

This section contains comments that explain what the program does. It is not required, but it is a good practice to include documentation in your programs.

Preprocessor

This section contains instructions for the preprocessor, which is a program that runs before the compiler. The preprocessor is used to perform tasks such as including header files and defining macros.

Definition

This section contains the definitions of variables, functions, and other data structures.

Global Declaration

This section contains declarations of variables and functions that are accessible from outside the main function.

Main Function

The main function is the starting point of every C program. It is responsible for initializing the program, calling other functions, and processing user input.

Subprograms

Subprograms are functions that are called from the main function. They can be used to perform complex tasks or to group related code together.

Here is an example of a simple C program:

```
/* This program prints "Hello, world!" */

#include <stdio.h>

int main() {

  printf("Hello, world!\n");

  return 0;

}
```

This program has the following structure:

- The documentation section contains a comment that explains what the program does.
- The preprocessor section includes the #include directive, which tells the compiler to include the contents of the stdio.h header file.
- The definition section contains the definition of the main function.
- The global declaration section contains the declaration of the printf function.
- The main function prints the message "Hello, world!" to the console.
- The program returns 0 to indicate that it terminated successfully.

Structures in C

How does the structure works in C programs?

In C programming, a structure (also called struct) is a way to group several related variables into one place. Each variable in the structure is known as a member of the structure. Unlike an array, a structure can contain many different data types (int, float, char, etc.).

To declare a structure, you use the struct keyword followed by the name of the structure and a list of the members. For example, the following code declares a structure called Person with two members: a name and an age:

```
struct Person {

  char *name;

  int age;

};
```

To create a variable of a structure type, you use the struct keyword followed by the name of the structure and the variable name. For example, the following code creates a variable called person of type Person:

```
struct Person person;
```

To access the members of a structure variable, you use the dot (.) operator. For example, the following code sets the name of the person variable to "John Smith":

```
person.name = "John Smith";
```

The following code sets the age of the person variable to 30:

```
person.age = 30;
```

Structures are a powerful tool that can be used to organize data in C programs. They can make your code more readable and maintainable.

Here are some examples of how structures can be used in C programs:

- To store information about a person, such as their name, age, and address.
- To store information about a product, such as its name, price, and quantity in stock.
- To store information about a game, such as the number of players, the size of the board, and the rules of the game.