Rajarata University of Sri Lanka
Department of Physical Sciences

COM1407
Computer Programming

LECTURE 03

VARIABLES AND DATA TYPES

T.C IRUGALBANDARA

Objectives

- At the end of this lecture students should be able to;
  - Define Keywords / Reserve Words in C programming language.
  - Define Identifiers, Variable, Data Types, Constants and statements in C Programming language.
  - Justify the internal process with respect to the variable declaration and initialization.
  - Apply Variable Declaration and Variable initialization statement.
  - Assigning values to variables.
  - Apply taught concepts for writing programs.

Keywords / Reserve Words

- Keywords are predefined, reserved words used in programming that have special meanings to the compiler.
- As C is a case sensitive language, all keywords must be written in lowercase.

Keywords / Reserve Words (Cont…)

_Bool	default	if	sizeof	while
_Complex	do	inline	static	
_Imaginary	double	int	struct	
auto	else	long	switch	
break	enum	register	typedef	
case	extern	restrict	union	
char	float	return	unsigned	
const	for	short	void	
continue	goto	signed	volatile	

Identifiers

- An *identifier* in C consists of a sequence of letters, digits, or underscore characters.
- Identifiers are set when you declare variables, arrays, enumerations, structures, union, typedef and functions.
- Identifier must be unique.
- They are created to give unique name to a entity to identify it during the execution of the program.
- You can choose any name for an identifier (excluding keywords).
- However, if you give meaningful name to an identifier, it will be easy to understand and work on for you and your fellow programmers.

Identifiers (Cont…)

- **Rules for writing an identifier**
  - A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.
  - The first letter of an identifier should be either a letter or an underscore.
  - However, it is discouraged to start an identifier name with an underscore.
  - There is no rule on length of an identifier.
  - However, the first 31 characters of identifiers are discriminated by the compiler.

Variable

- Programming language enable you to assign symbolic names, known as *variable names*, for storing program computations and results.
- A variable name can be chosen by you in a meaningful way to reflect the type of value that is to be stored in that variable.
- Variables can be used to store integers, floating-point numbers, characters, strings and even pointers to locations inside the computer's memory.

Variable (Cont…)

- The rules for forming variable names:
  - They must begin with a letter or underscore (_) and can be followed by any combination of letters (upper- or lowercase), underscores, or the digits 0–9.
  - The following is a list of valid variable names.
    - sum
    - pieceFlag
    - i
    - J5x7
    - Number_of_moves
    - _sysflag

Variable (Cont…)

- Examples for invalid Variable Names
  - sum$value - $ is not a valid character.
  - piece flag - Embedded spaces are not permitted.
  - 3Spencer - Variable names cannot start with a number.
  - int - int is a reserved word.
- You should always remember that upper- and lowercase letters are distinct in C.
  - Therefore, the variable names sum, Sum, and SUM each refer to a different variable.

Variable (Cont…)

- It's typically not practical to use variable names that are too long—just because of all the extra typing you have to do.
  - For example,
    - although the following line is valid

```
theAmountOfMoneyWeMadeThisYear =
theAmountOfMoneyLeftAttheEndOfTheYear –
theAmountOfMoneyAtTheStartOfTheYear;
```

  - this line

```
moneyMadeThisYear = moneyAtEnd –
moneyAtStart;
```

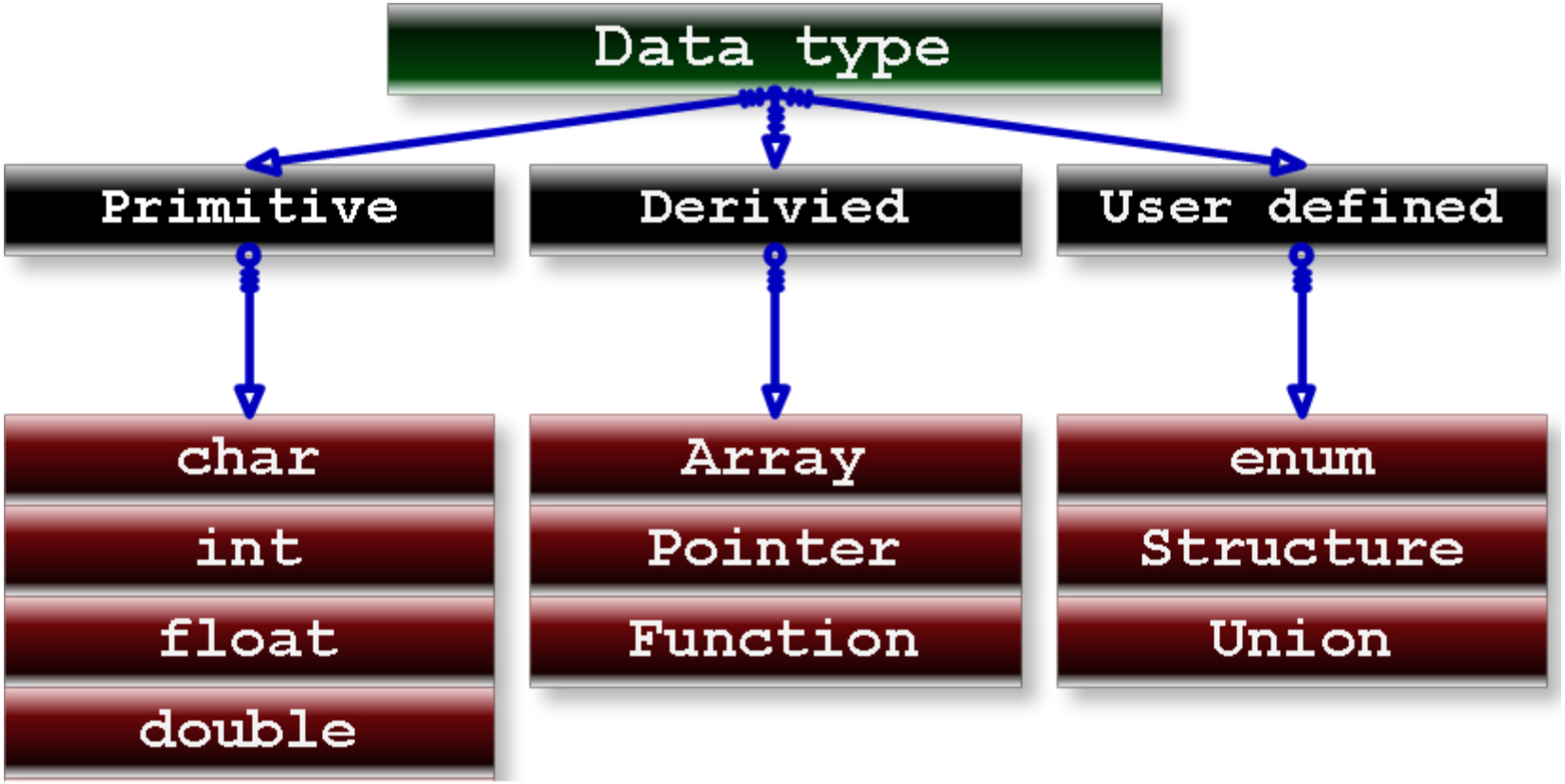
conveys almost as much information in much less space.

** Always remember to pick names that reflect the intended use of the variable

Data Types

- Now we know that, variables are placeholders used to store values.
- The data type of a variable determines how the bits representing those values are stored in the computer's memory.
- When you declare a variable, you can also supply a data type for it.
- All variables have a data type that determines what kind of data they can store.

Data Types (Cont...)



Data Types (Cont…)

- The C programming language provides int, float and char as basic data types.
- Additionally there are double and _Bool types available.
  - int – for storing integral numbers.
  - float – for storing floating-point numbers.
    - double – for storing double precision floating point numbers.
  - char – for storing a single character.
  - _Bool - for indicating an on/off, yes/no, or true/false situation. For storing value 1 or 0.

Data Types (Cont...)

Type	Meaning
`int`	Integer value; that is, a value that contains no decimal point; guaranteed to contain at least 16 bits of precision.
`short int`	Integer value of reduced precision; takes half as much memory as an `int` on some machines; guaranteed to contain at least 16 bits of precision.
`long int`	Integer value of extended precision; guaranteed to contain at least 32 bits of precision.
`long long int`	Integer value of extraextended precision; guaranteed to contain at least 64 bits of precision.
`unsigned int`	Positive integer value; can store positive values up to twice as large as an `int`; guaranteed to contain at least 16 bits of precision.
`float`	Floating-point value; that is, a value that can contain decimal places; guaranteed to contain at least six digits of precision.
`double`	Extended accuracy floating-point value; guaranteed to contain at least 10 digits of precision.
`long double`	Extraextended accuracy floating-point value; guaranteed to contain at least 10 digits of precision.

Data Types (Cont…)

`char`	Single character value; on some systems, sign extension might occur when used in an expression.
`unsigned char`	Same as `char`, except ensures that sign extension does not occur as a result of integral promotion.
`signed char`	Same as `char`, except ensures that sign extension does occur as a result of integral promotion.
`_Bool`	Boolean type; large enough to store the values `0` or `1`.
`float _Complex`	Complex number.
`double _Complex`	Extended accuracy complex number.
`long double _Complex`	Extraextended accuracy complex number.
`void`	No type; used to ensure that a function that does not return a value is not used as if it does return one, or to explicitly "discard" the results of an expression. Also used as a generic pointer type (`void *`).

Data Types (Cont…)

- **Storage size and Ranges**
  - Every value, whether it's a character, integer, or floating-point number, has a *range* of values associated with it.
  - This range has to do with the amount of storage that is allocated to store a particular type of data.
  - In general, that amount is not defined in the language.
  - It typically depends on the computer you're running, and is, therefore, called *implementation-* or *machine-*dependent.
  - For example, an integer might take up 32 bits on your computer, or perhaps it might be stored in 64.

Data Types (Cont…)

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

Data Types (Cont…)

Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

Statements

- A program statement is any valid expression (usually an assignment or function call) that is immediately followed by a semicolon, or it is one of the special statements like compound, statement, break, continue, do, for , go to, if, return, switch, while and null statement.

Variable Declaration

- When defining a particular variable, structure, union, enumerated data type, or typedef, the compiler does not automatically reserve any storage.
- The definition merely tells the compiler about the particular data type and (optionally) associates a name with it.
- After the definition has been made, variables can be declared to be of that particular data type.
- A variable that is declared to be of *any* data type *does* have storage reserved for it.

Variable Declaration (Cont…)

- Declaration is done by simply listing the variables before the terminating semicolon of the definition.
  - E.g.

    data_type variable;

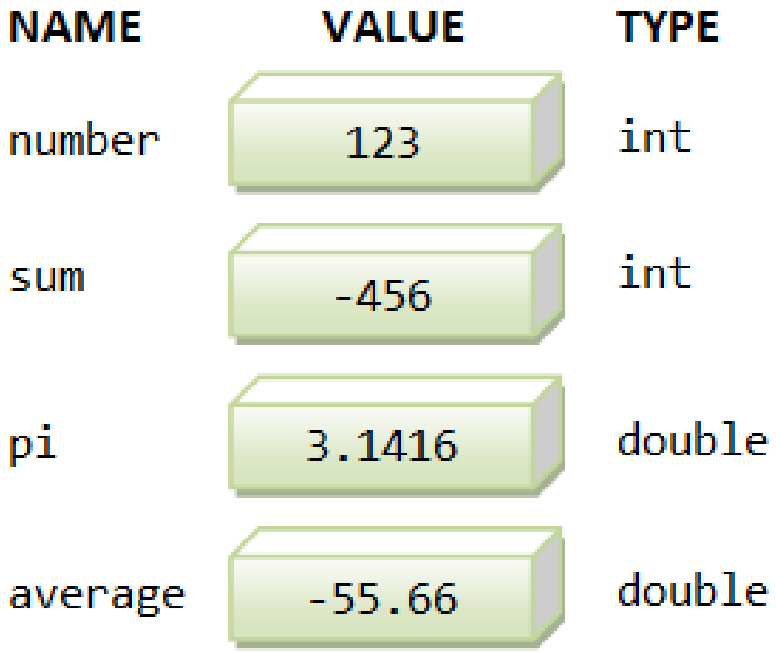
    int age;
- The language also enables storage to be allocated at the same time that a particular variable data type is defined.
- You can declare multiple variables in same type using a single statement as follows;

    int height, width;

Assigning Values to Variables

- After declaring a variable with a type, you can add some content or value to that variable.
- This is known as assigning values to variable.
- Some times we have to initialize the variables to its initial or default values and it is based on the problem we are addressing.
- This is known as variable initialization.
- There are two ways to assign value to a variable:
  - Assign the value directly in the program.
  - Ask from user to input a value and then assign that value.

Assigning Values to Variables (Cont…)



A variable has a *name*, stores a *value* of the declared *type*

Assigning Values to Variables (Cont…)

- Assign the value directly in the program.
  - Method 1: varible declaration and the assigning values as two separate statements;

```
data_type variable;

variable = value;

int length;

length = 20;
```

  - Here "=" is denoted as assignment operator.

Assigning Values to Variables (Cont…)

- Method 2: direct method for variable declaration and initialization;

```
data_type variable = value;
```

```
int length = 20;
```

- Variable initialization;

```
int age = 0;
float average = 0.0;
int minimum_marks = 40;
```

Assigning Values to Variables (Cont…)

```
#include <stdio.h>

int main (void)

{

	int integerVar = 100;

	float floatingVar = 331.79;

	double doubleVar = 8.44e+11;

	char charVar = 'W';

	_Bool boolVar = 0;

	printf ("integerVar = %i\n", integerVar);

	printf ("floatingVar = %f\n", floatingVar);

	printf ("doubleVar = %e\n", doubleVar);

	printf ("doubleVar = %f\n", doubleVar);

	printf ("charVar = %c\n", charVar);

	printf ("boolVar = %i\n", boolVar);

	return 0;

}
```

Variable Declaration & Initialization

Assigning Values to Variables (Cont…)

```
#include <stdio.h>
int main (void)
{
    int sum;
    sum = 50 + 25;
    printf ("The sum of 50 and 25 is
%i\n", sum);
    return 0;
}
```

declares the *variable* sum to be of type *integer*.
C requires that all program variables be declared before they are used in a program.

The number 50 is added to the number 25, and the result is stored (as indicated by the *assignment operator*, the equal sign) in the variable sum.

Assigning Values to Variables (Cont…)

```
printf ("The sum of 50 and 25 is %i\n",
sum);
```

Formatting Character:
The character that immediately follows the percent sign specifies what *type* of value is to be displayed at that point. In the preceding program, the
letter i is recognized by the printf routine as signifying that an integer value is to be displayed. Whenever the printf routine finds the %i characters inside a character string, it automatically displays the value of the next argument to the printf routine. Because sum is the next argument to printf, its value is automatically displayed after the characters "The sum of 50 and 25 is" are displayed.

Assigning Values to Variables (Cont…)

```
#include <stdio.h>
int main (void)
{
    int value1, value2, sum;
    value1 = 50;
    value2 = 25;
    sum = value1 + value2;
    printf ("The sum of %i and %i is %i\n",
                        value1, value2, sum);
    return 0;
}
```

Variable Declaration

Variable Initialization

Assigning Values to Variables (Cont…)

- Ask from user to input a value and then assign that value.

```
#include <stdio.h>
int main()
{
    int  number;
    printf("Type in a number : ");
    scanf("%i", &number);
    printf("The number you typed was %i\n", number);
    return 0;
}
```

Assigning Values to Variables (Cont…)

- `scanf("%i", &number);`
  - The *scanf* routine, which accepts the response, has two arguments.
  - The first ("%i") specifies what type of data type is expected (ie char, int, or float).
  - The second argument (&number) specifies the variable into which the typed response will be placed.
  - In this case the response will be placed into the memory location associated with the variable *number*.
  - This explains the special significance of the & character (which means the address of).

Assigning Values to Variables (Cont…)

```
#include <stdio.h>

int main()
{
    int units;
    char letter;
    float unit_price =12.5f;
    float amount = 0.0f;

    printf("Please enter # of units : ");
    scanf("%d", &units );

    printf("Please enter categogy id : ");
    scanf("  %c", &letter );

    amount = unit_price * units;
```

Using Type Specifies

```
#include <stdio.h>
int main (void)
{
    short shortvar = 34;
    printf ("int shortvar = %hi\n", shortvar);

    int variable = 1590;
    printf ("int variable = %i\n", variable);

    long int factorial = 1000001L;
    printf ("long int factorial = %li\n", factorial);

    long long int maxAllowedStorage = 100000011000LL;
    printf ("long long int maxAllowedStorage = %lli\n", maxAllowedStorage);
```

Using Type Specifies (Cont…)

```
    unsigned int counter = 10;
    printf ("unsigned int counter = %u\n", counter);

    unsigned counter2 = 20;
    printf ("unsigned counter2 = %u\n", counter2);

    unsigned char char_counter = 'a';
    printf ("unsigned char = %u\n", char_counter);

    unsigned char_counter2 = 'A';
    printf ("unsigned char_counter2 = %u\n", char_counter2);

    return 0;
}
```

Format Modifiers

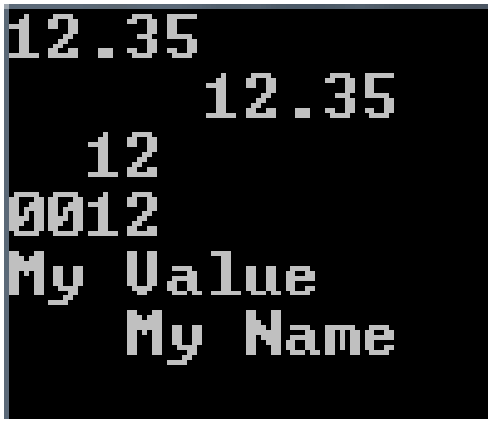
- Can be used to specify the required width of decimal integers and text strings

Modifier	Description
%d , %i	Print as decimal integer
%6d	Print as decimal integer, at least six characters wide.
%f	Print as floating point
%6f	Print as floating point, at least six characters wide.
%.2f	Print as floating point, 2 characters after decimal point.
%6.2f	Print as floating point, at least 6 wide and 2 characters after decimal point.
%-4s	Print as four character string with left justified
%4s	Print as four character string with right justified.

Format Modifiers (Cont…)

```
#include <stdio.h>
int main()
{
    float value = 12.3456f;
    int number = 12;

    printf ("%.2f\n",value);
    printf ("%10.2f\n",value);
    printf ("%4i\n",number);
    printf ("%04d\n",number);
    printf ("%-20s\n","My Value");
    printf ("%10s\n","My Name");
    return 0;
}
```



Objective Re-cap

- Now you should be able to:
  - Define Keywords / Reserve Words in C programming language.
  - Define Identifiers, Variable, Data Types, Constants and statements in C Programming language.
  - Justify the internal process with respect to the variable declaration and initialization.
  - Apply Variable Declaration and Variable initialization statement.
  - Assigning values to variables.
  - Apply taught concepts for writing programs.

References

- Chapter 04, Appendix A - Programming in C, 3rd Edition, Stephen G. Kochan

Q & A

NEXT: TYPE CASTING, COMMAND LINE ARGUMENTS AND DEFINING CONSTANTS