IS1201: Programming & Problem Solving

3. Data Types - 2

Viraj Welgama

Format Specifiers

- in C, format specifiers are used in functions like printf(), scanf() and similar functions to specify the type of data to be read or written.
- Following are some such specifiers;
  - Integer Types
    - %d: Signed decimal integer.
    - %i: Signed decimal integer (same as %d).
    - %u: Unsigned decimal integer.
    - %o: Unsigned octal.
    - %x: Unsigned hexadecimal integer (lowercase).
    - %X: Unsigned hexadecimal integer (uppercase).

Format Specifiers

- Floating-Point Types
  - %f: Decimal floating-point.
  - %F: Decimal floating-point (uppercase, same as %f).
  - %e: Scientific notation (lowercase).
  - %E: Scientific notation (uppercase).
  - %g: Use the shorter of %e or %f.
  - %G: Use the shorter of %E or %F.
  - %a: Hexadecimal floating-point (lowercase, C99).
  - %A: Hexadecimal floating-point (uppercase, C99).
  - %Lf: Long double (for printf).
- Character and String Types
  - %c: Character.
  - %s: String of characters.
- Pointers
  - %p: Pointer address.
- Special Types
  - %%: Literal percent sign.

C Integer Input/Output

```c
#include <stdio.h>

int main() {
    int testInt;
    printf("Enter an integer: ");
    scanf("%d", &testInt);
    printf("Number = %d", testInt);
    return 0;
}
```

- The scanf() function reads formatted input from the keyboard. When user enters an integer, it is stored in variable testInt.

- Note the '&' sign before testInt gets the memory address of the testInt variable and the value is stored in that address.

C Floats Input/Output

```c
#include <stdio.h>

int main() {
    float testFloat;
    printf("Enter a number: ");
    scanf("%f", &testFloat); //%f format string is used for float
    printf("Number = %f", testFloat);
    return 0;
}
```

- The format string "%f" is used to read and display formatted in case of floats.
- .2f%?
- lf%?

C Character Input/Output

```c
#include <stdio.h>

int main() {
    char ch;
    printf("Enter a character: ");
    scanf("%c", &ch);
    printf("Charater = %c", ch);
    return 0;
}
```

- Format string %c is used in case of character types.

C ASCII Code

```c
#include <stdio.h>

int main() {
    char ch;
    printf("Enter a character: ");
    scanf("%c", &ch);

    //when %c text format is used, char is displayed
    printf("You entered = %c\n", ch);

    //when %d text format is used, char is displayed
    printf("ASCII value of %c is %d", ch, ch);
    return 0;
}
```

1st rule of Programming:

If it works.... don't touch it!..

Keywords

- Keywords are predefined, reserved words used in programming that have special meanings to the compiler.
- Keywords are part of the syntax and they cannot be used as an identifier.

- example:

```
int money;
```

  - Here, int is a keyword that indicates 'money' is a variable of type integer.

Keywords

- As C is a case sensitive language, all its 32 keywords must be written in lowercase.

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

Identifiers

- Identifier refers to name given to entities such as variables, functions, structures etc.
- Identifier must be unique. They are created to give unique name to a entity to identify it during the execution of the program.
- For example:

```
int money;
double accountBalance;
```

  - Here, money and accountBalance are identifiers.
- Identifier names must be different from keywords. You cannot use int as an identifier because int is a keyword.

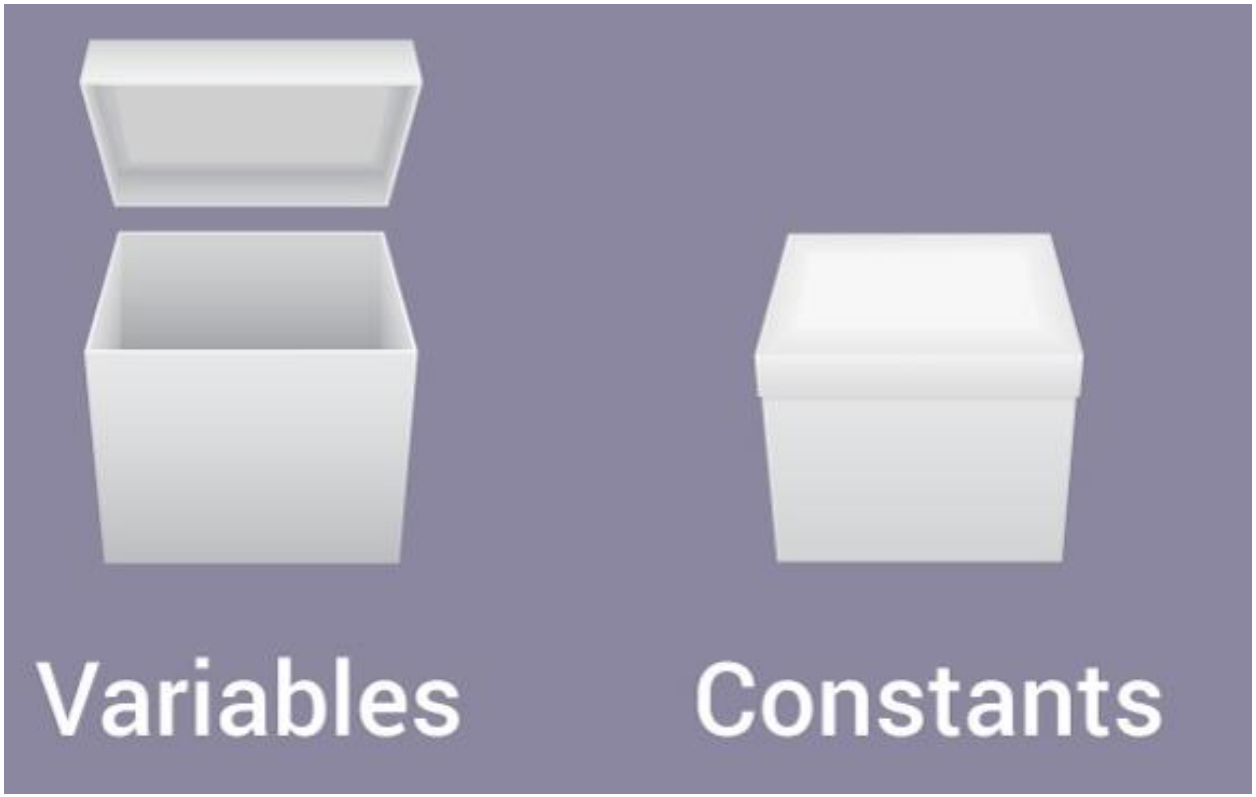
Rules for define an Identifier

1. A valid identifier can have letters (both uppercase and lowercase letters), digits and underscore only.

2. 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.

3. There is no rule on length of an identifier.
   - However, only the first 31 characters of a identifier are checked by the compiler.

Variables & Constants



Variables

- In programming, a variable is a container (storage area) to hold data.
- To indicate the storage area, each variable should be given a unique name (identifier).
- Variable names are just the symbolic representation of a memory location.
- For example:

```
int marks = 73;
```

  - Here, marks is a variable of integer type.
  - The variable is assigned value: 73
- The value of a variable can be changed, hence the name 'variable'.

Rules for naming a variable in C

1. A variable name can have letters (both uppercase and lowercase letters), digits and underscore only.
2. The first letter of a variable should be either a letter or an underscore.
   - However, it is discouraged to start an identifier name with an underscore.
3. There is no rule on length of an identifier.
   - However, only the first 31 characters of a identifier are checked by the compiler.
4. C is a strongly typed language. What this means it that, **the type of a variable cannot be changed.**

Constants/Literals

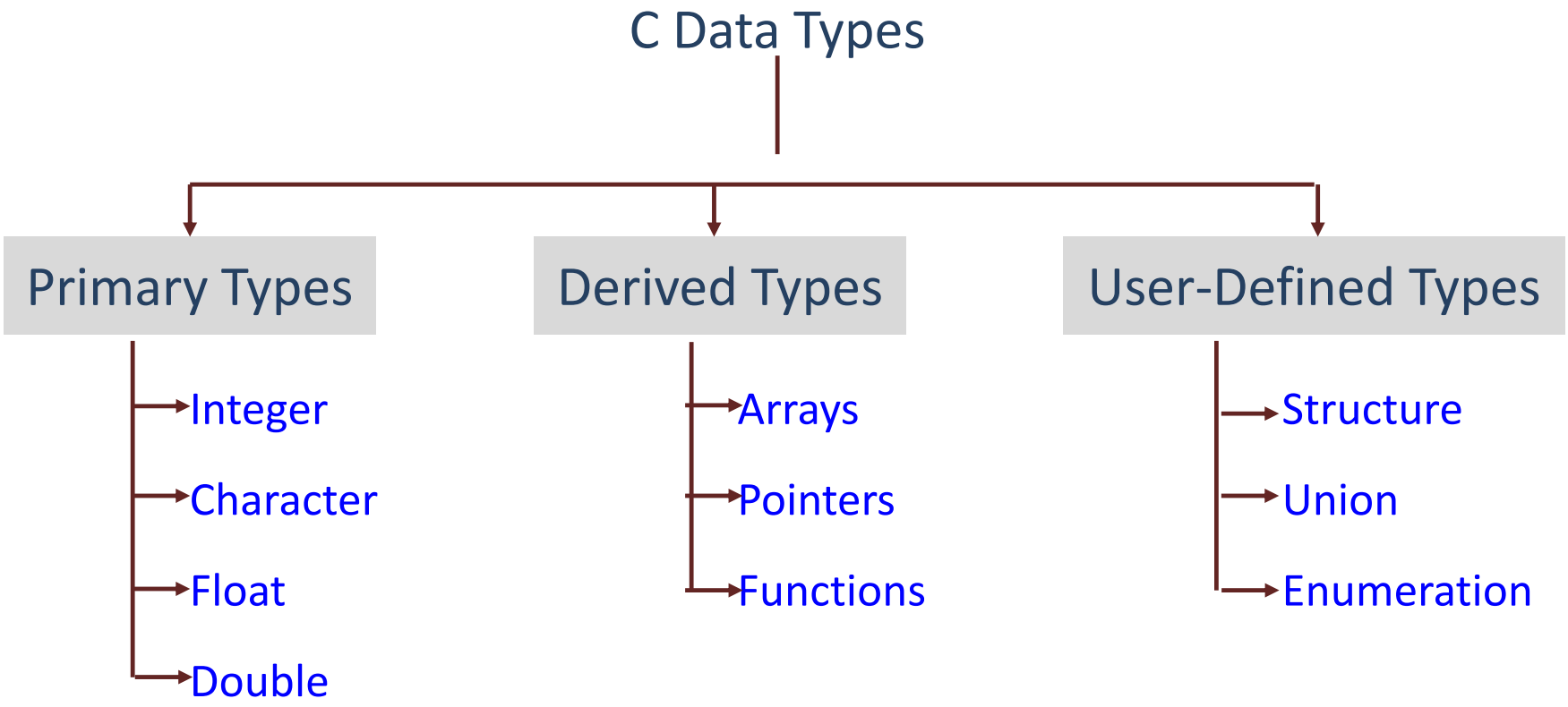
- A constant is a value or an identifier whose value cannot be altered in a program.
  - For example: 1, 2.5, "C programming is easy", etc.
- As mentioned, an identifier also can be defined as a constant.

- For example;

```
const double PI = 3.14;
```

  - Here, PI is a constant.
  - Basically what it means is that, PI and 3.14 is same for this program.

Data Types


C Data Types
Primary Types
Integer
Character
Float
Double
Derived Types
Arrays
Pointers
Functions
User-Defined Types
Structure
Union
Enumeration

Int: Integers

- In C programming, keyword int is used for declaring integer variable.

```
int age = 23;
```

  - Here, age is a variable of type integer.

- You can declare multiple variable at once in C programming.

```
int marks, age, NoofStudents;
```

float: for decimal numbers

- Floating type variables can hold real numbers such as: 2.34, -9.382, 5.0 etc.
- You can declare a floating point variable in C by using either float or double keyword.
- For example:

```
float bookPrice;
double accountBalance;
```

  - Here, both bookPrice and accountBalance are floating type variables.

- In C, floating values can be represented in exponential form as well.

```
float normalizationFactor = 22.442e2;
```

float vs double

- The size of float (single precision float data type) is 4 bytes.
- And the size of double (double precision float data type) is 8 bytes.
- Floating point variables has a precision of 6 digits whereas the precision of double is 14 digits.

char: for characters

- Keyword char is used for declaring character type variables.

- For example:

```
char ch = 'a';
```

  - Here, test is a character variable.
  - The value of test is ‘a'.

- The size of character variable is 1 byte.

Integer Types

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	8 bytes or (4bytes for 32 bit OS)	-9223372036854775808 to 9223372036854775807
unsigned long	8 bytes	0 to 18446744073709551615

Checking sizes

```c
#include <stdio.h>
#include <limits.h>

int main() {

    printf("CHAR_BIT    :   %d\n", CHAR_BIT);
    printf("CHAR_MAX    :   %d\n", CHAR_MAX);
    printf("CHAR_MIN    :   %d\n", CHAR_MIN);
    printf("INT_MAX     :   %d\n", INT_MAX);
    printf("INT_MIN     :   %d\n", INT_MIN);
    printf("LONG_MAX    :   %ld\n", LONG_MAX);
    printf("LONG_MIN    :   %ld\n", LONG_MIN);
    printf("SCHAR_MAX   :   %d\n", SCHAR_MAX);
    printf("SCHAR_MIN   :   %d\n", SCHAR_MIN);
    printf("SHRT_MAX    :   %d\n", SHRT_MAX);
    printf("SHRT_MIN    :   %d\n", SHRT_MIN);
    printf("UCHAR_MAX   :   %d\n", UCHAR_MAX);
    printf("UINT_MAX    :   %u\n", UINT_MAX);
    printf("ULONG_MAX   :   %lu\n", ULONG_MAX);
    printf("USHRT_MAX   :   %d\n", USHRT_MAX);

    return 0;
}
```

Checking sizes

```c
#include <stdio.h>
#include <limits.h>

int main() {

    printf("CHAR_BIT    :   %d\n", CHAR_BIT);
    printf("CHAR_MAX    :   %d\n", CHAR_MAX);
    printf("CHAR_MIN    :   %d\n", CHAR_MIN);
    printf("INT_MAX     :   %d\n", INT_MAX);
    printf("INT_MIN     :   %d\n", INT_MIN);
    printf("LONG_MAX    :   %ld\n", LONG_MAX);
    printf("LONG_MIN    :   %ld\n", LONG_MIN);
    printf("SCHAR_MAX   :   %d\n", SCHAR_MAX);
    printf("SCHAR_MIN   :   %d\n", SCHAR_MIN);
    printf("SHRT_MAX    :   %d\n", SHRT_MAX);
    printf("SHRT_MIN    :   %d\n", SHRT_MIN);
    printf("UCHAR_MAX   :   %d\n", UCHAR_MAX);
    printf("UINT_MAX    :   %u\n", UINT_MAX);
    printf("ULONG_MAX   :   %lu\n", ULONG_MAX);
    printf("USHRT_MAX   :   %d\n", USHRT_MAX);

    return 0;
}
```

```
CHAR_BIT    :   8
CHAR_MAX    :   127
CHAR_MIN    :   -128
INT_MAX     :   2147483647
INT_MIN     :   -2147483648
LONG_MAX    :   2147483647
LONG_MIN    :   -2147483648
SCHAR_MAX   :   127
SCHAR_MIN   :   -128
SHRT_MAX    :   32767
SHRT_MIN    :   -32768
UCHAR_MAX   :   255
UINT_MAX    :   4294967295
ULONG_MAX   :   4294967295
USHRT_MAX   :   65535
```