Introduction

1. Simple Program

General Form of a Program:

```
directives

int main(void)
{
  statements
}
```

Example:

```c
/*
* Name: Bismillah.c
* Purpose: Basic program to simulate fundamentals of C
* Author: Kalim Amzad
* Date: 24-03-2018
*/

#include <stdio.h>
int  main(void) /* Beginning of main program */
{
    printf("Bismillahir Rahmanir Rahim\n"); // Simple printf

    return 0;
}
```

Explain directives and main function:

Compiling and Linking:

- ***Preprocessing.*** The program is first given to a ***preprocessor***, which obeys commands that begin with # (known as ***directives***). A preprocessor is a bit like an editor; it can add things to the program and make modifications.
- ***Compiling.*** The modified program now goes to a ***compiler***, which translates it into machine instructions (***object code***). The program isn't quite ready to run yet, however.
- ***Linking.*** In the final step, a ***linker*** combines the object code produced by the compiler with any additional code needed to yield a complete executable program. This additional code includes library functions (like `printf`) that are used in the program.

If there's no `return` statement at the end of the `main` function, the program will still terminate. However, many compilers will produce a warning message (because the function was supposed to return an integer but failed to).

Changing printf statement:

```
printf("Bismillahir");
printf("Rahmanir");
printf("Rahim");

printf("Bismillahir\nRahmanir\nRahim");

printf("Bismillahir\n");
printf("Rahmanir\n");
printf("Rahim\n");
```

Explaining Comment:

2. Keywords

Keywords are predefined, reserved words used in programming that have special meanings to the compiler. For example:

```
int money;
```

Here, `int` is a keyword that indicates `'money'` is a variable of type integer.
As C is a case sensitive language, all keywords must be written in lowercase. Here is a list of all keywords allowed in ANSI C.

Keywords in C Language

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

Along with these keywords, C supports other numerous keywords depending upon the compiler.

Identifiers

Identifier refers to name given to entities such as variables, functions, structures etc.

```
int money;
float accountBalance ;
```

Here, `money` and `accountBalance` are identifiers.

Also remember, identifier names must be different from keywords. You cannot use `int` as an identifier because `int` is a keyword.

Rules for writing an identifier

1. A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.
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, the first 31 characters of identifiers are discriminated by the compiler.

Valid identifiers:

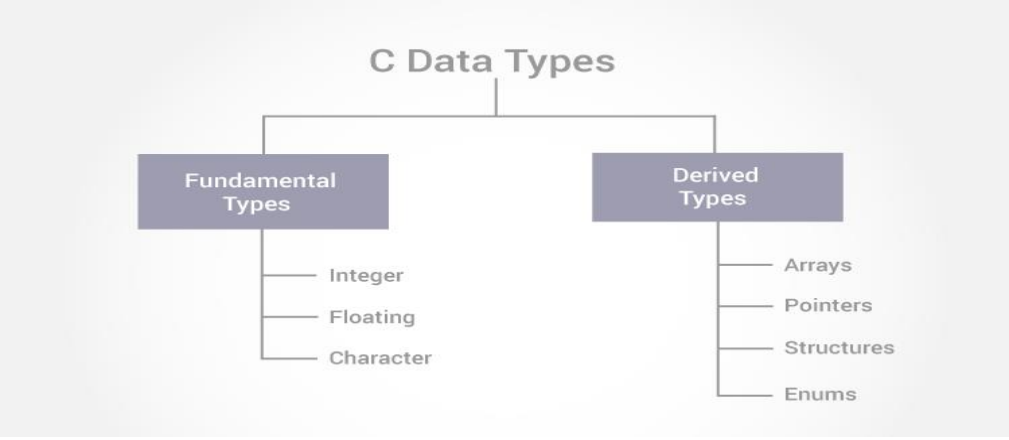
X Y12 sum-1 _temperature

Names area tax-rate TABLE

Invalid identifiers for the reasons stated:

4th	The first character must be a letter.
“x “	Illegal characters (“).
order-no	Illegal character (-).
error flag	Illegal character (blank space).

3. Data Type:



Integer Types

The following table provides the details of standard integer types with their storage sizes and value ranges –

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

To get the exact size of a type or a variable on a particular platform, you can use the **sizeof** operator. The expressions *sizeof(type)* yields the storage size of the object or type in bytes. Given below is an example to get the size of int type on any machine –

```
printf("Storage size for int : %d \n", sizeof(int));
```

Floating-Point Types

The following table provide the details of standard floating-point types with storage sizes and value ranges and their precision −

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

4. 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 playerScore = 95;
```

Here, `playerScore` is a variable of integer type. The variable is assigned value: 95.
The value of a variable can be changed, hence the name 'variable'.
In C programming, you have to declare a variable before you can use it.

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 variable name with an underscore. It is because variable name that starts with an underscore can conflict with system name and may cause error.
3. There is no rule on how long a variable can be. However, only the first 31 characters of a variable are checked by the compiler. So, the first 31 letters of two variables in a program should be different.

C is a strongly typed language. What this means it that, the type of a variable cannot be changed.

Constants

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.

The following rules apply to all numeric-type constants.
1. Commas and blank spaces cannot be included within the constant.
2. The constant can be preceded by a minus (-) sign if desired. (Actually the minus sign is an ***operator*** that changes the sign of a positive constant, though it can be thought of as a part of the constant itself.)

```
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.
Below are the different types of constants you can use in C.

1. Integer constants

An integer constant is a numeric constant (associated with number) without any fractional or exponential part. There are three types of integer constants in C programming:

- **Decimal constant (base 10)**

  Valid:

  0 1 743 5280 32767 9999

  Invalid:

12,245	illegal character (,)
36.0	illegal character (.)
10 20 30	illegal character (blank space)
123-45-6789	illegal character (-)
0900	the first digit cannot be a zero

- **Octal constant (base 8):** octal constant must start with a 0

  Valid:

  0 01 0743 077777

  Invalid:

  743 Does not begin with 0.

  05280 Illegal digit (8).

  0777.777 Illegal character (.)

- **hexadecimal constant (base 16):** A *hexadecimal* integer constant must begin with either Ox or OX

  Valid:

  OX OX1 OX7FFF Oxabcd

  Invalid:

  OX12.34 Illegal character (.).

  OBE38 Does not begin with Ox or OX.

  Ox. 4bff Illegal character (.).

  OXDEFG Illegal character (G)

2. Floating-point constants

A floating point constant is a numeric constant that has either a fractional form or an exponent form.

Valid:

0.	**1.**	**0. 2**	**827.602**
50000.	**0.000743**	**12.3**	**315.0066**
2E-8	**0.006e-3**	**1.6667E+8**	**.12121212e1**

Invalid:

1	Either a decimal point or an exponent must be present.
1,000.0	Illegal character (,).
2E+10.2	The exponent must be an integer quantity (it cannot contain a decimal point).
3E 10	Illegal character (blank space) in the exponent.

3. Character constants

A character constant is a constant which uses single quotation around characters. For example: 'a', 'l', 'm', 'F'

4. Escape Sequences

Sometimes, it is necessary to use characters which cannot be typed or has special meaning in C programming. For example: newline(enter), tab, question mark etc. In order to use these characters, escape sequence is used.

For example: `\n` is used for newline. The backslash `( \ )` causes "escape" from the normal way the characters are interpreted by the compiler.

Character	***Escape Sequence***	***ASCII Value***
bell (alert)	\ a	**007**
backspace	\ b	**008**
horizontal tab	**\t**	009
vertical tab	\ v	**01 1**
newline (line feed)	\ n	**01*0***
form feed	\ f	**012**
carriage return	\r	**013**

quotation mark (")	\"	034	
apostrophe (')	\'	039	
question mark (?)	\?	063	
backslash 0)	\\	092	
null	\0	000	

5. String constants

String constants are the constants which are enclosed in a pair of double-quote marks. For example:

"green" "Washington, D.C. 20005" “270-32-3456"

"$19.95" "THE CORRECT ANSWER IS:” “2* (I+3)/J "

“ “ " L i n e l \ n L i n e 2\nLine 3" “”

Variable Declaration and Assignment:

Variables must be declared before they can be used. To declare a variable we first specify the type of the variable, then its name.

Example:

```
int height;
float profit;

int height, length, width, volume;
float profit, loss;
```

- If several variables have the same type their declarations can be combined.
- A variable can be given a value by means of assignment.

Example:

```
height = 8;
length = 12;
width = 10;

// Before assigning a variable it must be declared first
//Right
int height;
height = 8;

// Wrong: First assigned before declared
height = 8;
int height;
```

Exercises:

1. **Computing Dimensional weight of a Box:**

Computing the Dimensional Weight of a Box

Shipping companies don't especially like boxes that are large but very light, since they take up valuable space in a truck or airplane. In fact, companies often charge extra for such a box, basing the fee on its volume instead of its weight. In the United States, the usual method is to divide the volume by 166 (the allowable number of cubic inches per pound). If this number—the box's "dimensional" or "volumetric" weight—exceeds its actual weight, the shipping fee is based on the dimensional weight. (The 166 divisor is for international shipments; the dimensional weight of a domestic shipment is typically calculated using 194 instead.)

Let's say that you've been hired by a shipping company to write a program that computes the dimensional weight of a box. Since you're new to C, you decide to start off by writing a program that calculates the dimensional weight of a particular box that's 12″ × 10″ × 8″. Division is represented by / in C, so the obvious way to compute the dimensional weight would be

```
weight = volume / 166;
```

where `weight` and `volume` are integer variables representing the box's weight and volume. Unfortunately, this formula isn't quite what we need. In C, when one integer is divided by another, the answer is "truncated": all digits after the decimal point are lost. The volume of a 12″ × 10″ × 8″ box will be 960 cubic inches. Dividing by 166 gives the answer 5 instead of 5.783, so we have in effect rounded *down* to the next lowest pound; the shipping company expects us to round *up*. One solution is to add 165 to the volume before dividing by 166:

```
weight = (volume + 165) / 166;
```

A volume of 166 would give a weight of 331/166, or 1, while a volume of 167 would yield 332/166, or 2. Calculating the weight in this fashion gives us the following program.

Solve Hints:

1. Declare variable and assign const value(Show different type of initialization)
2. Calculate volume
3. Calculate weight(type casting int/int = int)
4. Print the result

Modified version:

1. Instead assigning const value take it from user.
2. Use #define to declare const that divide volume to find weight.

2. **Converting Fahrenheit to Celsius**
   Formula: C = ((F-32)*5)/9 [C= Celsius value, F = Fahrenheit value]
3. **C Program to Calculate Area and Circumference of Circle**

Given, r = radius
Pi = value of pi

Formula: Area = pi * r2
Circumference = 2 * pi * r

4. **C Program to Calculate Area of Equilateral Triangle**
5. **C Program to Calculate Area of Right angle Triangle**
6. **C Program to Calculate Area of Rectangle**
7. **C Program to Calculate Area of Square**
8. **C Program to Add Two Integers Entered by User**
9. **C Program to Multiply two Floating Point Numbers**
10. **C Program to Find ASCII Value of Character Entered by User**