Your First C Program

vi hello.c and then type out (in insert mode) the following, to print “Hello World”

```
1. #include<stdio.h>
2. int main(void)
3. {
4.         printf(“Hello World\n”);
5.         return 0;
6. }
```

Revisit "Hello World" Program

```
1. #include<stdio.h>
2. int main(void)
3. {
4.         printf("Hello World\n");
5.         return 0;
6. }
```

Revisit "Hello World" Program

The first line is #include<stdio.h>

- # is a pre-processing directive
- #include tells the pre-processor to include the header file stdio.h into the program
- The angle brackets tells the preprocessor that the header file stdio.h will be available in the standard directory where all header files are available.

Revisit "Hello World" Program

```
1. #include<stdio.h>
2. int main(void)
3. {
4.         printf("Hello World\n");
5.         return 0;
6. }
```

Revisit "Hello World" Program

The second line is int main(void)

- main() is the name of a function. Execution of all C programs start with the function main.
- The word int and void are keywords
- int tells the compiler that this function returns an integer value.
- void tells the compiler that this function does not take any argument.

Revisit "Hello World" Program

```
1. #include<stdio.h>
2. int main(void)
3. {
4.         printf("Hello World\n");
5.         return 0;
6. }
```

Revisit "Hello World" Program

The third and last line are braces { }

- braces are used to surround the body of a function
- braces are used to group statements together.
- The right and left braces should match.

Revisit "Hello World" Program

```
1. #include<stdio.h>
2. int main(void)
3. {
4.         printf("Hello World\n");
5.         return 0;
6. }
```

Revisit "Hello World" Program

The fourth line is printf("Hello World\n")

- printf is a standard library function.
- Information about printf is included in the header file stdio.h
- The string Hello World is an argument to printf.
- \n represent a single character called newline. It is used to move the cursor to the next line

Revisit "Hello World" Program

```
1. #include<stdio.h>
2. int main(void)
3. {
4.         printf("Hello World\n");
5.         return 0;
6. }
```

Revisit "Hello World" Program

The fifth line is return 0

- It causes the integer value 0 to be returned to the operating system.
- The returned value may or may not be used.

Compile & Run a C program

- gcc hello.c –o hello
- ./hello
- Compiling Options
- gcc –help

- [sourav@gaya]$ hello

  Output: -bash: hello: command not found
- [sourav@gaya]$ PATH=$PATH:.
- [sourav@gaya]$ hello

  Output: Hello World

Convert miles, yards to kms

```
1. #include <stdio.h>
2. /* Convert miles, yards to kms */
3. int main(void)
4. {
5.       int miles, yards; float kms;
6.       miles = 26; yards = 385;
7.       kms = 1.609 * (miles + yards / 1760.0);
8.       printf("\nThe distance in kms is %f.\n\n",
         kms);
9.       return 0;
10.  }
```

Convert miles, yards to kms

```
1. #include <stdio.h>
2. /* Convert miles, yards to kms */
3. int main(void)
4. {
5.       int miles, yards; float kms;
6.       miles = 26; yards = 385;
7.       kms = 1.609 * (miles + yards / 1760.0);
8.       printf("\nThe distance in kms is %f.\n\n",
         kms);
9.       return 0;
10.  }
```

Comments

- Anything written between /* … */ is a comment and is ignored by the compiler

- Lines starting with // are also comments and ignored by the compiler

Convert miles, yards to kms

```
1. #include <stdio.h>
2. /* Convert miles, yards to kms */
3. int main(void)
4. {
5.       int miles, yards; float kms;
6.       miles = 26; yards = 385;
7.       kms = 1.609 * (miles + yards / 1760.0);
8.       printf("\nThe distance in kms is %f.\n\n",
         kms);
9.       return 0;
10.  }
```

Variables Declaration Stmt.

Line 5: int miles, yards;

is a declaration statement

miles and yards are called variables.

The keyword int tells the compiler that these variables are of type integer and take on integer values.

Line 5 float kms;

is again a declarative statement

Tells the compiler that variable kms is of type floating point.

Convert miles, yards to kms

```
1. #include <stdio.h>
2. /* Convert miles, yards to kms */
3. int main(void)
4. {
5.       int miles, yards; float kms;
6.       miles = 26; yards = 385;
7.       kms = 1.609 * (miles + yards / 1760.0);
8.       printf("\nThe distance in kms is %f.\n\n",
         kms);
9.       return 0;
10.  }
```

Assignment Statements

Line 6: miles = 26; yards = 385;

- are assignment statements
- = is an assignment operator. Assigns the value 26 to variable miles and 385 to variable yards.
- Place spaces on either side of a binary operator. This makes the program more readable.

Convert miles, yards to kms

```
1. #include <stdio.h>
2. /* Convert miles, yards to kms */
3. int main(void)
4. {
5.       int miles, yards; float kms;
6.       miles = 26; yards = 385;
7.       kms = 1.609 * (miles + yards / 1760.0);
8.       printf("\nThe distance in kms is %f.\n\n",
         kms);
9.       return 0;
10.  }
```

Expressions

The seventh line:

kms = 1.609 * (miles + yards/1760.0);

- is also an assignment statement.
- the value of the expression on the right hand side is computed and assigned to the floating type variable *kms*.

Convert miles, yards to kms

```
1. #include <stdio.h>
2. /* Convert miles, yards to kms */
3. int main(void)
4. {
5.       int miles, yards; float kms;
6.       miles = 26; yards = 385;
7.       kms = 1.609 * (miles + yards / 1760.0);
8.       printf("\nThe distance in kms is %f.\n\n",
         kms);
9.       return 0;
10.  }
```

PRINTF

The eighthth line: printf("\nThe distance in kms is %f.\n\n", kms);

- printf() can take variable number of arguments
- The control string %f is matched with the variable kms.
- It will print the variable kms as a floating point number where the format %f occurs.

Output

- printf("<control string>",other argument);

Control String	How printed
c	as a character
d	as a decimal integer
e	as a floating-point in scientific notation
f	as a floating-point number
g	in e or f format, whichever shorter
s	as string

Input

Eg: scanf("%d", &integer1);

- Obtains a value from the user
- scanf uses standard input (usually keyboard)
- %d - indicates data should be a decimal integer
- &integer1 - location in memory to store variable

- Change the program to convert miles, yards to kms. such that the value of *miles* and *yards* are taken as input from the user

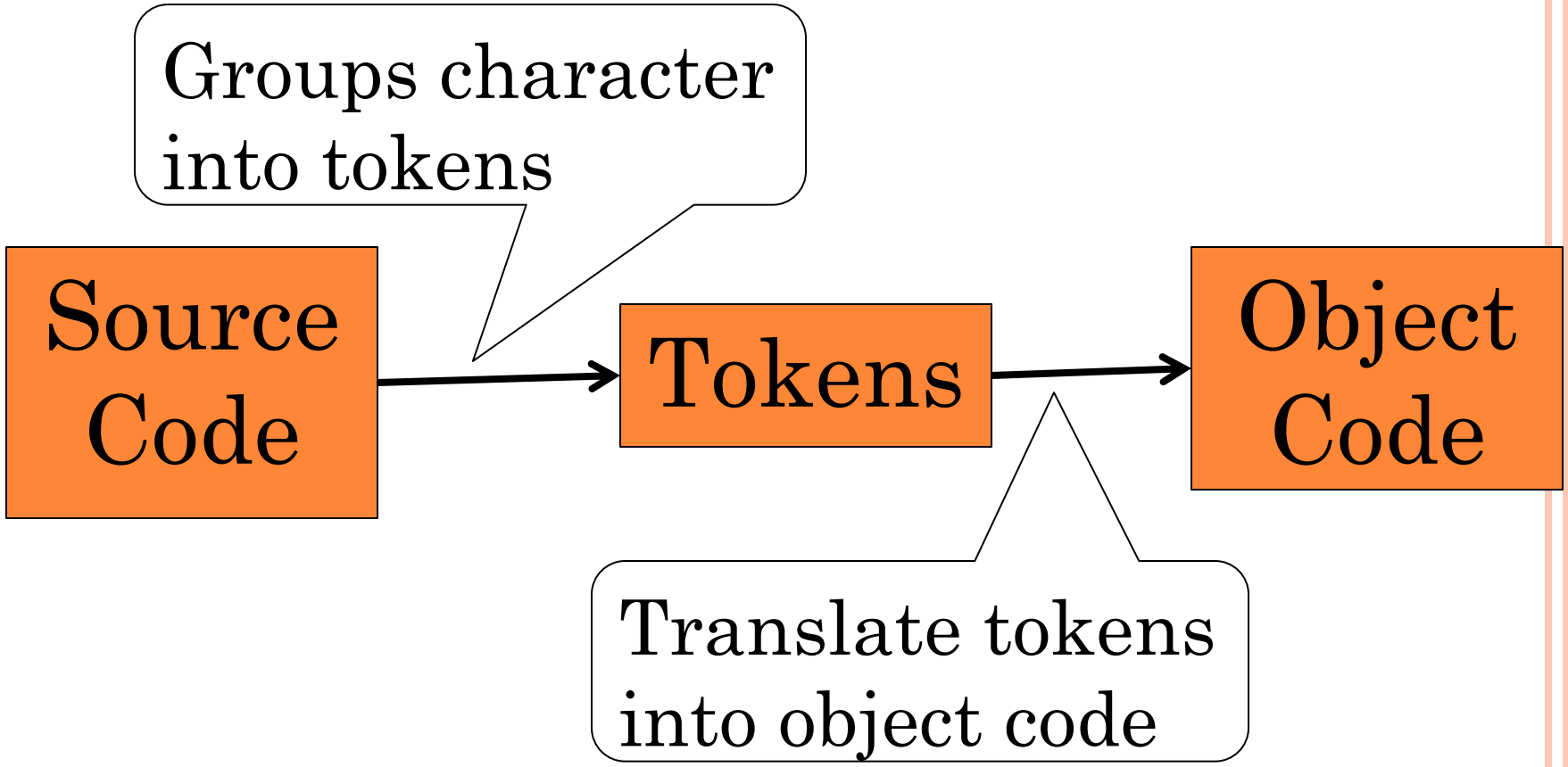
CONVERT MILES, YARDS TO KMS

```
1. #include <stdio.h>
2. /* Convert miles, yards to kms */
3. int main(void)
4. {
5.     int miles, yards; float kms;
6.         printf("Enter integer value of mile and
       yards\n");
7.         scanf("%d%d",&miles,&yards);
8.         kms = 1.609 * (miles + yards / 1760.0);
9.         printf("\nThe distance in kms is %f.\n\n",
           kms);
10.        return 0;
11.    }
```

Syntax and Semantics

- Syntax refers to the grammar structure and semantics to the meaning
- A program may be syntactically correct but semantically wrong.
- Compiler checks for syntax errors.

Compilation Process



Tokens

- Syntactic units of the language
- Six kind of tokens
  - Keywords
  - Identifiers
  - Constants
  - String constants
  - Operators
  - Punctuators

Keywords

- Reserved words have a strict meaning
- Cannot be redefined.

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

- **if, else, switch, case, default –** Used for decision control programming structure.
- **break –** Used with any loop OR switch case.
- **int, float, char, double, long –** These are the data types and used during variable declaration.
- **for, while, do** – types of loop structures in C.
- **void –** One of the return type.
- **goto –** Used for redirecting the flow of execution.
- **auto, signed, const, extern, register, unsigned –** defines a variable.

- **return** – This keyword is used for returning a value.
- **continue** – It is generally used with for, while and dowhile loops, when compiler encounters this statement it performs the next iteration of the loop, skipping rest of the statements of current iteration.
- **enum** – Set of constants.
- **sizeof** – It is used to know the size.
- **struct, typedef** – Both of these keywords used in structures (Grouping of data types in a single record).
- **union** – It is a collection of variables, which shares the same memory location and memory storage.
- **volatile -** The Volatile Keyword is used for making volatile objects. Volatile objects can only be altered by hardware and not a program.

Identifiers

- Give unique names to objects in a program.
- Composed of sequence of letters, digits and the special character _
- Letter or _ should be the first character.
- Examples
  - k, principal, i123, _id (Allowed)
  - not#me, 1iam,-plus (Not allowed)

Programming Tip

- Choose name of the identifiers that are meaningful to enhance readability and documentation of the program.

Constants

- Integer Constant: Finite strings of decimal digits, eg. 0, 77, etc
- Floating point constant: 345.0
- Character constants: ‘a’, ‘b’
- Octal Integer: 0123
- Hexadecimal Integer: 0x123

String Constants

- Sequence of characters enclosed in a pair of double quotes.
- Example
  - “” (Null String)
  - “A String Constant”
  - “a = b + c”
  - “abc” “def” equivalent to “abcdef”
  - “The boy said \”Hello\””