Creating, Compiling and Executing

Creating program

```
#include<stdio.h>
int main() {
     printf("       AA\n");
     printf("      A  A\n");
     printf("     A    A\n");
     printf("    A      A\n");
     printf("   AAAAAAAAAA\n");
     printf("  A          A\n");
     printf(" A            A\n");
     printf("A              A\n");
}
```

Creating, Compiling and Executing

Compiling and Executing

- Name the above program as `bigA.c`
- Compile the program by `gcc -o bigA bigA.c`
- It produces a file named `bigA` in your directory.
- Now execute it as `./bigA` in your directory.
- If `-o` switch is not used, i.e., you use `gcc bigA` default object file `a.out` will be produced.

Creating, Compiling and Executing

Example 1

```
#include<stdio.h>
int main() {
   // Adds two numbers, declare before use.
   int first, second;

   /* We are yet to learn about variables and types.  *
    * But use two int variables first and second here.*/

   printf("Enter two integers: ");
   scanf("%d %d", &first, &second):
   printf("The two numbers are: %d  %d\n",
          first, second);
   printf("Their sum: %d\n", first + second);
}
```

Creating, Compiling and Executing

Compiling and Executing

- Name the above program as `sumTwoNumbers.c`
- Compile the program by `gcc -o sumTwoNumbers sumTwoNumbers.c`
- It produces a file named `sumTwoNumbers` in your directory.
- Now execute it as `./sumTwoNumbers` in your directory.
- Default object file `a.out` gets produced if `-o` not used.
- We talk more about other simple compiler switches when needed.

Basic Features

Comment & Preprocessor

- A C program is preprocessed before compilation.
- Preprocessor commands are directives which start with a # character.
- Eg., C's standard I/O library has number of headers like **<stdio.h>**.
- Comments are essential for annotating programs.
- A comment in a single line: **// Comment**
- A multiline comment: **/* Winged comments */**

Basic Features

Functions

- Functions form building blocks of C programs.
- Function is block of statements provided with a name.
- A function computes a value, and **returns** it.
- C program can consists of several functions but `main()` is mandatory.
- `main()` returns a value (status code) for OS.
- Program still terminates even if there is no return in `main()`.
- Two types of functions:
    - Library functions,
    - User defined functions

Basic Features

Comments & Printing Strings

```
#include<stdio.h>
int main() {
    printf("Hello ");        /* Forgot to close comment ....
    printf("Everybody ");
    printf("in L15 \n");     /* Did not forget to close */
    printf("World\n");

}
```

Basic Features

Variables

- Most programs will need a sequence of computations.
- Storing of temporary result is important.
- Storage locations are given symbolic names and known as **variable**
- Each variable has a **type** (defining kind of data) associated with.

Basic Features

Types

- Some native types: `int, float, double, char`
- The largest value `int` can store is 2,147,483,647.
- The smallest value `int` can store is -2,147,483,647.
- `float` can store larger than an `int` variable.
- Sometime `float` variable gives an approximation to stored number, eg., 0.1 can be output as 0.09999999999999987.

Basic Features

Declaration of Variable

- Declaration of a variable should precede its use.
- A valid identifier: a sequence of one or more letters, digits or _, but not a reserved word.

$$\begin{aligned}
\text{letter} &= \text{a} \ldots \text{z} \mid \text{A} \ldots \text{Z} \\
\text{digit} &= 0 \ldots 9 \\
\text{Identifier} &= [\_]\text{letter}^{+}\text{digit}^{*} \\
\text{Integers} &= \text{digit}^{+} \\
\text{Real number} &= \text{digit}^{*}.\text{digit}^{+}
\end{aligned}$$

Basic Features

Reserved Words: Declaration

```
auto, char, double, extern, float, int, long, register,
short, signed, static, struct, typedef, union, unsigned,
void, volatile
```

Reserved Words: Others

```
break, case, continue, default, do, else, for, goto, if,
return, sizeof, switch, while
```

Basic Features

Assignments

A value in a variable stored through **assignment**

```
#include<stdio.h>                  |   #include<stdio.h>
int main() {                       |   int main() {
     int height;                   |      height = 8;
     int length, breadth;          |      int height;   // WRONG
     height = 8;                   |      int length, bredth;
}                                  |   }
```

Mixing `float` with `int` in assignment is possible but not safe. Type conversion will be shortly.

```
int i;
float f;
i = 72.95; // stores 72 in i
f = 135;   // stores 135.0 in f
```

Writing Simple Programs

Output

- Use `printf` for display of current value of variable.
- `printf("Height = %d\n", height);`
- `%d` is place holder for value to be filled by decimal.
- For `float` use `%f` which display number with 6 digits after decimal point.
- To force `n` digits after decimal point use `%.nf`

Writing Simple Programs

Example 2

```
#include<stdio.h>

int main() {
    int height = 8;
    int length = 32;
    int breadth = 16;
    int volume, weight;

    // 166 = allowable cubic inches per pound
    volume = height * length * breadth;
    weight = volume/166;

    printf("weight of the box = %d kg\n ", weight);
}
```