# Structured Programming: Basics

Compiled by - Nazmus Sakib Akash

# Challenge: Do you understand this code?

```
.MODEL SMALL
.STACK
.DATA
AAA DB 10, 13, ‘Hello$’
.CODE
MAIN PROC
MOV AX, @DATA
MOV DS, AX
LEA DX, AAA
MOV AH, 09H
INT 21H
MOV AX, 4C00H
INT 21H
MAIN ENDP
END MAIN
```

It prints Hello

# This one much easier, isn't it?

```
#include<stdio.h>
int main() {
    printf("Hello\n");
    return 0;
}
```

# Structured Programming Language

**Enforces a logical structure on the program being written to make it more efficient and easier to understand and modify.**

# Levels of programming language

High level language – Java, PHP, C#

Mid level language – C, C++

Low level language – Assembly

Machine language

# BASICS OF C

## Structured Programming

# Variable

- A memory location that stores some value temporarily
- Has a symbolic name
  - So that you know what the variable contains
- Has a specific data type
- Must be declared before using
  - In C, variables are declared at the beginning of (main) function

# Variable naming

**Which of the following are valid names of variables?**

- foo
- 55
- foo5
- 5foo
- foo_5
- 5_foo
- _foo5
- _5foo
- foobar
- foo_bar
- fooBar
- foo bar

# Data Types:

| Data type | Sign | Length | Size (byte) | Keyword | Format specifier |
|---|---|---|---|---|---|
| Character | Unsigned | - | 1 | `unsigned char` | `%c` |
| | Signed | | 1 | `(signed) char` | |
| Integer | Unsigned | Regular | 4 | `unsigned int` | `%u` |
| | | Short | 2 | `unsigned short (int)` | `%hu` |
| | | Long | 4 | `unsigned long (int)` | `%lu` |
| | | Long long | 8 | `unsigned long long (int)` | `%llu` |
| | Signed | Regular | 4 | `(signed) int` | `%d` |
| | | Short | 2 | `(signed) short (int)` | `%hi` |
| | | Long | 4 | `(signed) long (int)` | `%ld` |
| | | Long long | 8 | `(signed) long long (int)` | `%lld` |
| Floating-point | Signed | Single | 4 | `float` | `%f` |
| | | Double | 8 | `double` | `%lf` |
| | | Long double | 16 | `long double` | `%LF` |

# Arithmetic operators

| Operator | Operation | Precedence |
|---|---|---|
| % | Remainder | 1 |
| / | Divide | |
| * | Multiply | |
| + | Add | 2 |
| - | Subtract | |
| = | Assign | 3 |

# Self-assignment operators

| Operator | Task | Example |
|---|---|---|
| += | Add a value | `a += 5;` |
| ++ | Add one | `a++; ++a;` |
| -= | Deduct a value | `a -= 5;` |
| -- | Deduct one | `a--; --a;` |
| *= | Multiply a value | `a *= 5;` |
| /= | Divide a value | `a /= 5;` |

# Relational operators

| Operator | Task |
|---|---|
| == | Equal |
| != | Not equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |

# Logical operators

| Operator | Task | Example |
|---|---|---|
| && | Logical and | `(a < b) && (a > c)` |
| \|\| | Logical or | `(a < b) || (a > c)` |
| ! | Negate | `!(a > b)` |

# Correct the following C program

```
#include<stdio.h>
int main() {
  int a, b;
  scanf("%d%d%d", &a, &b, &c);
  if(a > b) {
    printf(a);
  } else {
    printf(b);
  }
  return 0;
}
```

```
#include<stdio.h>
int main() {
  int a, b;
  scanf("%d%d%d", &a, &b, &c);
  if(a > b) {
    printf("%d\n", a);
  } else {
    printf("%d\n", b);
  }
  return 0;
}
```

# Correct the following C program

```
#include<stdio.h>
Int Main() {
  int a, b, c;
  scanf("%d%d%d", a, b, c);
  if(a > 0) printf("Positive\n");
  if(b > 0) printf("Positive\n");
  if(c > 0) printf("Positive\n");
  return 0;
}
```

```
#include<stdio.h>
int main() {
  int a, b, c;
  scanf("%d%d%d", &a, &b, &c);
  if(a > 0) printf("%d\n", a);
  if(b > 0) printf("%d\n", b);
  if(c > 0) printf("%d\n", c);
  return 0;
}
```