

## CS1100 – Introduction to Programming

- Programming.
- Model. History of a computing machine.
- Representation of numbers.
- Started with C-programming.

Lectures  
1,2 and 3

- Data Types in C and representations, range of values for each type.
- Arithmetic operators, and operator precedence.
- Formatting the Input and the Output with various data types.

Lectures  
3, 4 and  
5.

## Weighted sum of 2 numbers

- Suppose  $x$  denotes marks in Maths,  $y$  denotes marks in Physics.
- We wish to calculate weighted total such that Maths marks are given 30% weightage and Physics marks are given 70% weightage.
- $z = \frac{30}{100}x + \frac{70}{100}y$ .

## Weighted sum of 2 numbers

---

```
#include <stdio.h>

/* weighted sum 2 integers */

main() {
    int mathMarks = 98;
    int phyMarks = 99;
    int total;

    total = (30/100)*mathMarks + (70/100)*phyMarks;
    printf("%d\n", total);
}
```

- 
- What is the output of the program?
  - Is the variable `total` still guaranteed to be an integer?

## Weighted sum of 2 numbers

---

```
#include <stdio.h>

/* weighted sum 2 integers */

main() {
    int mathMarks = 98;
    int phyMarks = 99;
    float total;          /* float variable */

    total = (30/100)*mathMarks + (70/100)*phyMarks;
    printf("%f\n", total); /* change here */
}
```

- 
- What is the output of the program?
  - $\frac{30}{100}$  and  $\frac{70}{100}$  evaluate to 0 and therefore total is zero.

## Weighted sum of 2 numbers – a correct program

---

```
#include <stdio.h>

/* weighted sum 2 integers */

main() {
    int mathMarks = 98;
    int phyMarks = 99;
    float total;

    total = (30.0/100)*mathMarks + (70.0/100)*phyMarks;
    printf("%f\n", total);
}
```

- 
- What is the output of the program?

## Learnings so far..

- C allows different kinds of variables to be declared.
- C defines arithmetic operators, like  $+$ ,  $-$ ,  $*$ ,  $/$ ,...
- Have meaningful names for variables  
`mathMarks`, `phyMarks`, `total`

choose variable names to be indicative – good programming practice

avoid reserved words like `int`, `float`, .. as variable names.

- C defines arithmetic operators, like  $+$ ,  $-$ ,  $*$ ,  $/$ ,...
- **Assignment operator** `"="`: used to change contents of a variable.

```
scanf(format-string, &var1, &var2, ... , &var3);
```

---

- scanf is a function which allows us to accept inputs.
- Usually functions take **fixed** number of parameters/arguments.
- scanf takes variable number of arguments.
- Notice the **&** preceeding the variables.

## Exercise: Swap two integers

- Two integers  $x$  and  $y$  contain 10 and 20 respec.
- Need to exchange values in  $x$  and  $y$ .  
`swap two integers`.
- Write a C program to do the same.

## Swap – fill in correct code

```
#include<stdio.h>

main() {
    int x, y;

    printf("Enter x:");
    scanf("%d", &x);
    printf("Enter y:");
    scanf("%d", &y);

    /* Fill in code here */

    printf("x = %d\n", x);
    printf("y = %d\n", y);
}
```

# Variable modification

- A C program is a sequence of commands that modify different variables using different operators.
- Basic operators in C.
  - Operator precedence and associativity.
- Basic data types in C.
  - How much space does a particular data type take?
  - How to input and output variables of a particular type?

# Basic operators in C

- Arithmetic operators:  $+$ ,  $-$ ,  $*$ ,  $/$
  - Modulus operator:  $\%$   
 $x \% y$  : remainder when  $x$  is divided by  $y$ .
  - Assignment operator:  $=$
- 

## Operator precedence:

- **first:** parenthesized sub-expression; inner-most to outer-most.
- **second:**  $*$ ,  $/$ ,  $\%$  ; left to right.
- **third:**  $+$ ,  $-$  ; left to right.
- $\text{total} = 30 / 100 * \text{mathMarks} + 70 / 100 * \text{phyMarks};$   
 $\text{total} = ((30 / 100) * \text{mathMarks}) + ((70 / 100) * \text{phyMarks});$
- $z = a + b * c * d \% e - f / g$   
 $z = a + (((b * c) * d) \% e) - (f / g)$

## Increment / decrement operators

- ++, --
- prefix and post-fix only to a variable.

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

```
main() {
```

```
    int x, y;
```

```
    int n = 10;
```

```
    x = n++;
```

```
    y = ++n;
```

```
    printf(" x =  %d, y = %d\n", x, y);
```

```
}
```

# Assignment operator =

Form: variable-name = expression

- $z = x + y$
- $x + y = z$       **Incorrect form**
- Assignment between different data types.
  - What happens if you assign float to int and vice versa?
- Multiple assignments.
  - $x = y = z = (a + b);$
  - evaluations happen right to left.
- $x = x + 10$  can be written as  $x += 10;$
- instead of  $+$ , we can also have  $-$ ,  $*$ ,  $/$ ,  $\%$

## Exercises

Write a program that reads an integer from the input and prints 0 if the integer is even and 1 if the integer is odd.

---

Write a program that takes as input a 3 digit integer, separates the digits of the integer and prints the individual digits separated by spaces.

For example if the input is 194, then your program must print  
1 9 4