# WARM UP LESSONS – BARE BASICS

## CONTENTS

## COMMON PRIMITIVE "DATA TYPES" FOR VARIABLES

```
int  float  char  string (C++)
```

Samples for variables declaration:

```
char  grade;        // declaring the variable symbol "age" as integer type
int   age;          // declaring the variable symbol "age" as integer type
int   x, y, z;      // declaring multiple of them with the same data type
float amount;       // float means allowing the data to contain decimal places.
Bool  isValid;      // Boolean variable to contain either true or false

// you can do declaration and initialization together in a single expression. E.g.

char grade = 'A';
int  age = 90;
int x=10, y=50, z;
float amount = 10.94;
bool isValid = true;
```

To stay within scope of this workshop, these primitive data types will be sufficient. Later, you may learn how to create your own data types.

## ABOUT STANDARD INPUT / OUTPUT

| C | C++ |
|---|---|
| scanf, getc, gets, etc. | cin , cin.getline(), etc. |
| Printf | cout |
| Review the content from the C book | Look into Basic C++ I/O Stream document here. |

Sample 1:

In C standard Output stream version:

```
#include <stdio.h>
int main()
{
  printf( "I am alive!  Beware.\n" );
  return 0;
}
```

In C++ standard Output stream version:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   cout << "I am alive!  Beware.\n";
   return 0;
}
```

Note: the difference in the header include expression and cout vs printf

## MORE ON STANDARD OUTPUT IN C STANDARD

- use *printf*
- from the example above :
  `printf ( "The area of this rectangle is : %d metres.\n",  area ) ;`
- Other display format strings

  **`%d`** – print the value of a decimal (based 10) integer

  **`%5d`** - print the decimal integer in five character positions, right justified.

  **`%-5d`** - left justified.

  **`%c`** - print a single character.

  %s - print a sequence of character. (more on characters in later days)

  **`%f`** - print the value of a signed, decimal, floating point number.

  **`%6.2f`** - print the decimal floating point number in six character positions, right justified and to an accuracy of two decimal places

Sample 2: Try the following and see they all look like: IMPORTANT: Do NOT just use the keyboard Copy & Paste. You must "type" them all in yourself.

In 'C' syntax:

```
// don't forget the header files

void main()
{
   int abc = 97;
   printf("1(%5d)\n", abc);
   printf("2(%-5d)\n", abc);
   printf("3(%10c)\n", 'A');
   printf("4(%-10c)\n", 'A');
   printf("5(%20s)\n", "hello!");
   printf("6(%-20s)\n", "hello!");
   printf("7(%*d)\n", 10, abc);
   printf("8(%*c)\n", 10, abc);
    return;
}
```

In 'C++' syntax:

```
// don't forget the header files

void main()
{
   int abc = 97;
   cout << "1(" << setw(5) << abc << ')' <<  '\n';
   cout << "2(" << setiosflags(ios::left) << setw(5) << abc << ')' << '\n';
   cout << "3(" << setiosflags(ios::right) << setw(10) << 'A' << ')' << '\n';
   cout << "4(" << left << setiosflags(ios::left) << setw(10) <<  'A' << ')' << '\n';
   cout << "5(" << right << setw(20) << "hello!" << ')' << '\n';
   cout << "6(" << left << setiosflags(ios::left) << setw(20) << "hello!" << ')' << '\n';
   cout << "7(" << right<< setw(10) << abc << ')' << '\n';
   cout << "8(" << setw(10) << (char) abc << ')' << '\n';
    return;
}
```

Sample 3:

```
int main()
{
  int n0, n1, n2, n3;
  char term = ',';

//C:   printf(" Enter four digits: e.g. 6,10,49,3 \n");
//C:   scanf("%d, %d, %d, %d/n", &n0, &n1, &n2, &n3);

  cout << " Enter four digits: e.g. 6,10,49,3 \n";
  cin >> n0 >> term >> n1 >> term >> n2 >> term >> n3;

//C:   printf("You have entered %d, %d, %d, %d\n", n0, n1, n2, n3);
//C:   printf(" %d is the smallest\n", n);
  cout << "You have entered " << n0 << " " << n1 << " " << n2 << " " << n3 << endl;
  return 0;
}
```

From now on, we'll just use C++ I/O stream method.

Sample 4:

```
// don't forget the basic standard IO header files
#include <math.h>

void main()
{
       int  base;
       int height;
       char ch;

       cout << "Enter Base, height: ";
       cin  >> base >> ch >> height;
       cout << endl << "base = " << base << ", height = " << height
              << ", hypotenuse = " << sqrt((base * base) + (height * height)) << endl;
       return;
}
```

## PRACTICE EXERCISES:

1. Rewrite this sample 2 with C++ standard Output.
2. Create a program to ask user to enter a the base and height, then display the result of hypotenuse.
3. Create a program to ask user to enter a number as Fahrenheit. Your program will convert this to Celsius.
4. Then, create another program to ask user to enter Celsius. Your program will convert this to Fahrenheit.

# ABOUT MATH EXPRESSIONS / OPERANDS / OPERATORS

Expressions can be as simple as a single value and as complex as a large calculation or even some conditional expressions (will be covered a bit later), etc. They are made up of two elements, operators and operands.

Operators:

Arithmetic operators :

- Binary : + - / * %
  - e.g. remainder = num1 % num2 ;
- unary : ++ or – or + & - by itself
  - e.g. ++x;
  - --x;
- prefix vs posfix unary : --x vs x—
  - e.g. int x = 10, y;
  - y = --x;
  - y = x--; // what is the final value of x & y

Relational operators : (will be further discussed on Day 2)

- == > < >= <= !

Logical operators: (will be further discussed on Day 2)

- && ||

LValue rule / Assignment

```
y = x +  10;    // correct
x + 10 = y      // invalid

int x = 10;
int y = 20, z = 30;
bool what;

x =  (y  = (z = 40) );
what  = ( y= (z==30) );  //   what is the value of x after assignment
```

CASTING

- force the compiler to regard a value being of a different type by the use of casting

```
int i = 3, j = 2 ;
float fraction ;
fraction = (float) i / (float) j ;
cout << "fraction :" << fraction << endl ;
```

Try this one. Compile and run.
Look at the answer.

```
int i = 3, j = 2 ;
float fraction ;
fraction = i /  j ;
cout << "ans:" << fraction << endl ;
```

After you have tried the left one, try this one. Compile and run. See what happen to the answer.

```
int i = 3, j = 2 ;
float fraction ;
fraction = (float) i / (float) j ;
cout << "ans :" << fraction << endl ;
```

## PRACTICE EXERCISE

1) Write a program to:
- ask user to provide the radius of a circle
- calculate the circumference and area.
- display all values

2) Plotting the "*" triangle. Write a function that outputs a right triangle of height and width *n*, so *n = 6* would look like the image on the right. You can assume there are only 6 levels. Then, you should allow user to decide the number of levels.

```
     *
    **
   ***
  ****
 *****
******
```

Tip: If you have difficulty in figuring out the loop structure, hardcode the layout of 6 levels, find the pattern, and convert to loop structure.

3) You borrow $1000. Your annual interest rate is 12%. You pay back $100 a month. If it is based on compound interest rate, how much do you still own after 4 months? The display should be something like this:

.e.g. interest you own at 1st month = $1000 * (12%/12) = $10
You pay off $100. Thus you still owe $910 at the end of 1st month
i.e. $1000 + $10 - $100.

interest you own at 2nd month = $910 * (12%/12) = $9.10
You pay off $100. Thus you still owe $910 at the end of 1st month
i.e. $910 + $9.10 - $100.

Output should look like this:

```
Month              total interest paid          still owe
    1                    $ 10.00                $910.00
    2                    $ 9.10                 $819.10
    3 ……..etc.
```

4) Write a program to perform factorial of 2 to 6. Display the result backward as shown on the right.

(note: your program must do the calculation, not hardcode the result.

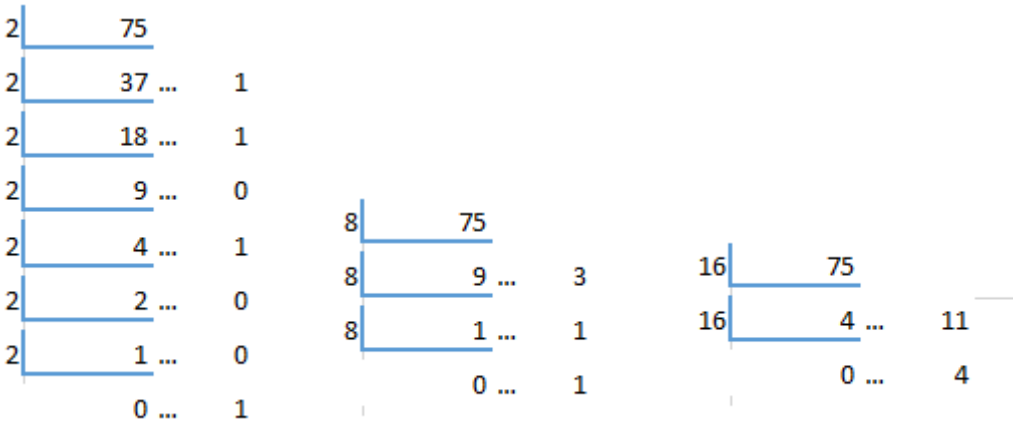
```
6 != 720
5 != 120
4 != 24
3 != 6
2 != 2
```

# BASE CONVERSION

Common Bases :

- Decimal == base-10. Range: 0,…, 9
- Hexadecimal == base-16. Range : 0 … A, B, C, D, E, F
- Octal == base-8, Range: 0, …, 7
- Binary == base-2 . Range: 0, 1

Convert base-10 to base-2 base-10 to base-8 base-10 to base-16




So $75_{10}$ == $100\ 1011_2$ == $113_8$ == $4B_{16}$

## CONVERT OTHER BASES TO BASE-10

| Binary to Decimal | Octal to Decimal | Hex to Decimal |
|---|---|---|
| 1 1 1 1 | 1 1 2 1 | 1 2 1 1 |
| $2^3$ $2^2$ $2^1$ $2^0$ = $15_{10}$ | $8^3$ $8^2$ $8^1$ $8^0$ = $593_{10}$ | $16^3$ $16^2$ $16^1$ $16^0$ = $4625_{10}$ |

**Tips:**

- A single hex digits == 4 bits
- When converting binary to hex, first break the bits into 4 bits chunk.

e.g. Take $10111001010_2$ , break it into 10 1100 $1010_2$.

Dec value for each chunk …… 2 12 10

Hex value = 2 C A $_{16}$

$10111001010_2$ == $2CA_{16}$

## PRACTICE EXERCISES

Decimal (base-10) to Hexadecimal (base-16) and Binary (base-2)

| Decimal (base-10) | Hexadecimal (base-16) | Binary (base-2) |
|---|---|---|
| 12 | | |
| 14 | | |
| 13 | | |
| 15 | | |
| 16 | | |
| 127 | | |
| 255 | | |
| 256 | | |
| 1023 | | |
| 1024 | | |
| 2047 | | |
| 2048 | | |
| 65535 | | |
| 65536 | | |

Binary (base-2) to Hexadecimal (base-16) and Decimal (base-10)

| Binary (base-2) | Hexadecimal (base-16) | Decimal (base-10) |
|---|---|---|
| 00001 | | |
| 00011 | | |
| 00111 | | |
| 01001 | | |
| 110011 | | |
| 1001110 | | |
| 00001001 | | |
| 10001001 | | |
| 01001001 | | |
| 10101000 | | |

Bits Operation

Another examples of using bits and bits shifting

```
#define N 1<<3
#define E 1<<2
#define S 1<<1
#define W 1
char goodDir = N | E;
```