

# WARM UP LESSONS – BARE BASICS

## CONTENTS

Common primitive “data types” for variables .....	2
About standard input / output .....	2
More on standard output in C standard .....	3
Practice Exercise .....	6
About Math Expressions / Operands / Operators .....	7
Casting .....	7
Practice Exercise .....	8

## 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	<a href="#">Look into Basic C++ I/O Stream document here.</a>

### 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: (do NOT forget the header files)

**C version:**

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

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

    // the following will generate an error, fix it
    printf("You have entered %d, %d, %d, %d\n", n, n1, n2, n3);
    return 0;
}
```

**// C++ version:**

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

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

    // the following will generate an error, fix it
    cout << "You have entered " << n << " " << n1 << " " << n2 << " " << n3 << endl;
    return 0;
}
```

Sample 4 ( in C++ version)

```
// don't forget to enter the basic standard IO header files.
// See samples at the beginning of this doc.
// you will need include a math.h in order to use the function "sqrt(...)"

#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 EXERCISE

1. Rewrite this sample 4 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.

$$\text{Fahrenheit to Celsius : } ^\circ\text{C} = (^\circ\text{F} - 32) \times 5/9$$

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 postfix 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 ;
```

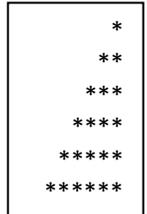
After you have tried the left one, try this  
one. Compile and run. See what happen to the  
answer.

```
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

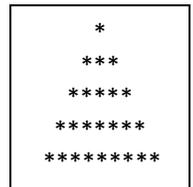
- 2) Plotting the "\*" triangle. Write a function that outputs a right triangle of height and width  $n$ , so  $n = 6$  would look like the image on the right.



- a. You can assume there are only 6 levels if you are not familiar with "loop" structure. Otherwise, you should do (b) only.
- b. 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) Like (2), except the triangle is an isosceles triangle – see the image on the right.



- 4) 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 1<sup>st</sup> month =  $\$1000 * (12\%/12) = \$10$   
 You pay off \$100. Thus you still owe \$910 at the end of 1<sup>st</sup> month  
 i.e.  $\$1000 + \$10 - \$100$ .

interest you own at 2<sup>nd</sup> month =  $\$910 * (12\%/12) = \$9.10$   
 You pay off \$100. Thus you still owe \$910 at the end of 1<sup>st</sup> 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.	

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

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

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