C++

Introduction to Expressions, User Defined Functions, Arrays, Enumerators and Structures

Expressions

Valid combination of values, variables, functions, and operators results in a VALUE

Values: 2, -636, 0.00427, 23.947, false, 'A'

Variables: declared/defined and initialized

Functions: sin(?), pow(?,?), toupper(?), exit()

Operators: +, -, *, /, %
==, !=, <, <=, >, >=
&&, ||, !

Functions

Declaration

```
r_type fname(parameter list);
```

Definition header should be same

```
r_type fname(parameter list)
{
        body of function
        composed of statements
        return r_expr;
}
```

rtype: return type of function

fname: name of function

parameter list: comma separated list of variable declarations

return: reserve word

(it ends execution of functions body to caller function)

rexpr: expression to return to caller

rtype can be void, in case return line may be omitted

or rexpr should not be provided

Functions

```
// to compute and return average
// of three integer parameters
float average(int a, int b, int c)
{
    float avg;
    int sum = a+b+c;
    avg = sum/3;
    return avg;
}
```

Arrays

#include <array>

array<type, size> var *= {values}* ;

type is a data type, either built-in or user defined

var is array name and size is size of the array

= {values} is assignment operator followed by list of comma separated initial values to be assigned to it (var) at time of declaration. It is optional.

; is statement terminator

Arrays

#include <array>

array<int, 10> a; declares an array of 10 ints

array<char, 50> nam; declares array of 50 chars

array<bool, 4> avail; declares an array of 4 bools

array<double, 5> nums = {38, 54.3, 34, 98.2, 83.3};
declares array of 5 doubles

In C++ indices of arrays are **ZERO** based, i.e., indices of a are 0,1,2,3,4,5,6,7,8,9 and that of nums are 0,1,2,3,4

Example

```
// to compute and return average of
// whole integer array with N values
float average(array<int,5> a, int N)
{
        int sum = 0;

        int j = 0;
        while (j < N)
        {
                sum = sum+a[j];
                j = j+1;
        }

        float avg = sum/N;
        return avg;
}
```

```
#include <iostream>
#include <array>
using namespace std;
int main()
{
        array<int,5> v = {3,5,4,7,4};
        cout << average(v,5);
        return 0;
}
```

User defined types (UDTs)

Enumerators

A list of values having unique type, e.g., PucitDegree, RGBColor, Gender, Coins, etc

```
enum Degree {SE, CS, IT, GIS};

enum Coin {penny=1, dime=10,
quarter=25, dollar=100};
```

Better to use enum classes

Structures

A composite data type to store more than one value, accessible by component name rather index as in array.

```
struct Student
{
        int rollno;
        string name;
        float cgpa;
};
```

Enumerators and Structures

```
enum Gender {male, female};


int main()
{
    Gender g;
    g = male; // or female
    if (g==male)
    {
        cout << "Male";
    }
    else
    {
        cout << "Female";
    }
    return 0;
}
```

```
struct Box
{
    int height, width, depth;
};

int main()
{
    Box b;
    b.height = 3;
    b.width = 5;
    b.depth = 2;

    int v;
    v = b.height * b.width * b.depth;

    cout << "Volume " << v << endl;

    return 0;
}
```

Functionalities for UDTs

The UDTs, being user (i.e., programmer) defined were not known to C++ language makers, so almost NO functionality is available in C++ for UDT, including input, output, comparison, arithmetic operator and functions if applicable.

So, being user defined, programmers has to build that required functionality through creation of several functions.

Program Composition

Code

Statements or instructions like

- Declarations,
- Assignment operation,
- Output,
- Input,
- Function call
- if, if-else
- while
- etc

Data

- Values used by statements.
- Variables/Arrays of
  - Built-in type
  - Enumeration type
  - Structures type
  - Mixture of above
  - Other data types that may not discuss in PF

- Simple variables
- Simple Arrays
- Simple Structures and enumerator
- Their combinations
  - Arrays of structures
  - Array as a structure component
  - Structure as structure component
  - Etc, etc

Pick the PACE

I am not asking you to get READY, you are supposed to be READY.

I am not going to kick you, you have to do it

- By yourself
- For yourself

- Do ‘die hard’ work with honesty and dedication,
- Put ego aside and ask questions
- Avoid friends seated nearby (or possibly tape your mouth :-)