C++ Coursae

Monday, March 24, 2025 10:03 AM

Namespaces:
"::" is scope resolution operator: tells from which namespare is the command belonging to reduce naming conflict

Eg: std::cout

second:
Using namespace (xyz);
Third:
Qualified using namespace variants:
Eg
Using std::cout
Using std:: in
Using std::endl

Input and output streams:

Cout = output stream from console(<<)
Cin = input stream (>>)

The insersion (<<) and extraction (>>) operators can be used in chain

Eg std::cin var1;>> Var1>>Var2;
std::cout<<"entry1"<<Var1<<"second entry"std::endl

endl and " \n" are used for new line

Variable and Constants
Initialization :
Int a = 1;
Int x (1);
Int x {1};

char xyz = ' ... '

Fundamental data types:
Character char xyz = ' ';
Integer (signed and uznsigned)
Floating-Point types (float)
Boolean type(bool)

#include <climits> : size and precision of the datatype

No of unique values a datatype can have is 2^nbits

Constants:
1) Litral Constants:
   Const double pi = 3,141592653589793 ;
2) #define pi 3.14159
   This will always replace any occurance of "pi" with tbe number

---

Array and Vectors
1) Array: compound /structure datatype
   Collection of elements----each element could be accessed directly
   INITIALIZATION
   Int Array [4] = {1,2,3,4}//definer elements, if lesser inputs given the other elements are set to 0 by default
   Int Array []= {1,2,3,4} //elements are counted automatically

   Multi dimensional arrays:
   Syntax:

```
int movie_rating [rows][columns]
{
        {1,2,3,4},
        {5,6,7,8},
        {9,10,11,12}
};
```

1	2	3	4
5	6	7	8
9	10	11	12

Vectors:
Syntax:

#include <vector>
Using namespace std;
Vector<datatype>name (no. of elements);// here all the elements will be initialized to 0.
Vector<datatype>name (no. of elements, value); // all the elements shall be set to the "value"
Vector<datatype>name {'a', 'b' , 'c', 'd', 'e'};// declaring char individually
Vector<datatype>name {0,1,2,3,4};// declaring the values individually

Accessing elements in an array is similar to Arrays:
Vector<int>Counter{0,1,2,3,4,5};
Cout<<counter[6];// no bounce checking
Output : 5

Method 2: vector.at(element index);//same as array to fetch a value in vector
vector.push_back(element);// dynamically adds another element to the end of the vector

Vector fetures:
Vector_name.at(index);//fetches the element at that index
Vecto_name.push_back(element);// adds the element to the back of the vector
Vector_name.size()// tells the current size of the vector

2D vectors
Initialization: a 2D vector is Vector of Vectors

Vector<vector<datatype>>Vector_Name

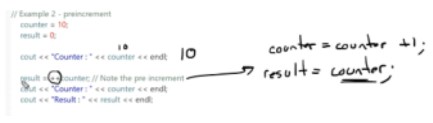
Expression, Statement and Operators

Expression: fundamental block of programming
Statement: functional line of code usually ending with a ;
Usually contain expression

Arthematic operators:
+ addition (overloaded)
- subtraction
* multiplication
/ devision
% modulo (remander)----inly works with integers and gives remander 10%3 = 1

If i divide 100 by 200 and both var are defined as int, i would get the result as int i e. not 0.5 but 0. to get 0.5, i must use double or float.

Increm,ent and Decrement operator (can be usewd to move pointers)
'DONT OVERUSE IT
#NEVER USE TWICE FOR THE SAME VARIABLE IN THE SAME STATEMENT



```
// Example 2 - preincrement
   counter = 10;
   result = 0;

   cout << "Counter : " << counter << endl;    10

   result = ++counter; // Note the pre increment
   cout << "Counter : " << counter << endl;
   cout << "Result : " << result << endl;
```

counter = counter +1;
result = counter;

Result = counter ++; (post increment)
Result will save the value of counter before incrementing it anf then increase
result = ++counter (pre increment)
Result will save the value of incremented counter as the counter is incremented before it is ssaved in the result

Mixed Type Expression:

Higher to lower order: Long double, double, float, unsigned long, long, unsigned int, int
^this is in decreasing order
Coercion: conversion of one operend to another datatype
 PROMOTION: CONVERSION FROM LOWER TO HIGHER DATATYPE
Demotion. Conversion from higher to lower datatyype

Promotion and demotion using static cast: you can static_case the value assigned to the same type variable.
eg: double average
Int total
Int count
Average = static_cast<double>(total)/count

#this will promote total to double.

Using bool:
Bool equal_result {false};

Initialized to false.
 equal_result = (num1==num2);
Cout<<equal_result;

Output:
1 = 1
Equal_number = 1;
That is true.

The output is 1 = 1 true
0 = false
#IF THE STORED INTEGER IS OF A NUMERIC TYPE
In case of char initialized variable: the output would be true and false
// Exercise on operators
#include <iostream>
using namespace std;

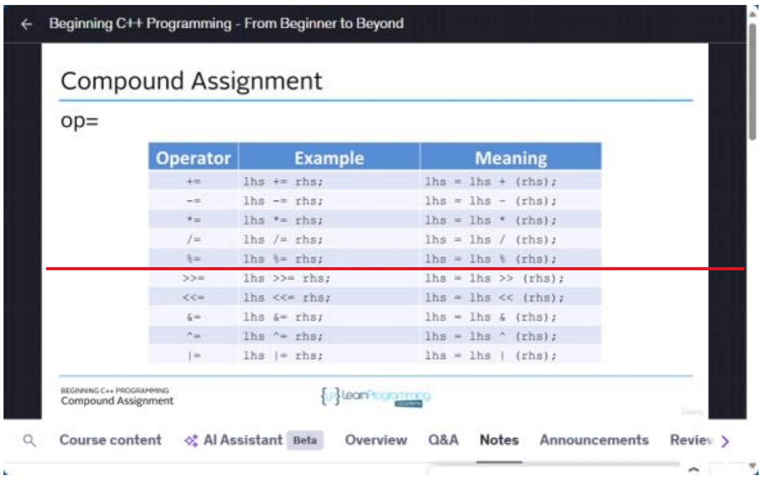
Using boolalpha

```
int main(){


    int number1 {0};
    int number2 {0};
    number1 = 10;
    number2 = 20;
    cout<<boolalpha;
    cout<< (number1<number2);
    cout<<noboolalpha;
    cout<< (number1<number2);
}
```
#keep the comparison statement in bracket to print true or false

Logical Operatorsd
! NOT
&& and
|| OR
Precedence in increasing order
! > && > ||

Compound operators:
General rule: lhs (operator)= rhs
Read as: lhs = lhs (operator) rhs


Beginning C++ Programming - From Beginner to Beyond

Compound Assignment

op=

Operator	Example	Meaning
+=	lhs += rhs;	lhs = lhs + (rhs);
-=	lhs -= rhs;	lhs = lhs - (rhs);
*=	lhs *= rhs;	lhs = lhs * (rhs);
/=	lhs /= rhs;	lhs = lhs / (rhs);
%=	lhs %= rhs;	lhs = lhs % (rhs);
>>=	lhs >>= rhs;	lhs = lhs >> (rhs);
<<=	lhs <<= rhs;	lhs = lhs << (rhs);
&=	lhs &= rhs;	lhs = lhs & (rhs);
^=	lhs ^= rhs;	lhs = lhs ^ (rhs);
|=	lhs |= rhs;	lhs = lhs | (rhs);

BEGINNING C++ PROGRAMMING
Compound Assignment

#check for operator precedance list

Flow Control: mQrdering statement sequentially------>making Decition------> Looping and repeating

1) If
2) If else
3) If….else if…else
4) Nested if statement
5) Switch

General Syntax: include break and default statement
If break not included then the code performs all the statements without checking the case.

```
switch(expression) {
  case x:
    // code block
  break;
 case y:
   // code block
  break;
 default:
   // code block
}
```
6) Conditional Operator: ? :
(if Condition) ? Expression in case true : expression in case false ;
Just like if else statement.

Input output manupulators:
#include <iomanip>

Cout<<fixed<<setprecision(1); //this sets the decimal precision to 1 decimal

Special datatype for vectors:

ⓘ Why Use size_t?

- It's guaranteed to be big enough to hold the size of any object in memory.
- It avoids signed/unsigned comparison warnings when using .size(), .length(), etc.
- It's portable across platforms (e.g., 32-bit vs 64-bit).

Looping:
Use cases:
1) Specific number of time
2) For each element in a collection
3) While the specific condition remains true
4) Until a specific condition becomes false
5) Unti we reach the end of the input
6) Forever
7) ++++

1) For Loop: used for itirating for a specific number of time

Syntax: for(initialization ; condition range ; increment)
        {*………*;}

        #note: you can initialize and increment multiple variables with comma
        For(int i {0},j{0}; i <=5; i++,j++)
            {
                Jdkavbökjdv;
            }

1) Range-based for loop: one iteration for every element in range or collection
   #1 definee an array or vector of collections

   The increment variable (num) is initialized to the value of vector or arrays

   Int arr[] {100, 90, 70};

   For (int num : arr)
   {
        Cout<< num;
   }
   Result: 100 90 70

   Instead of "int" we can uswe "auto" for auto deduction of variable type wrt collection.
   For (auto num : arr)

   Similarly for vectors
2) While loop: iterates while codition remains true
   While loop is a pre test loop therefore the test in done in the beginning of the loop--if fails then the loop is never entered.

   #used to prompt a valid value input. (while statement opposite to the requirement opposite to the requirement)

   'check examples for usage of bool in a while loop
   Also, when using vectors, use the .size() for index increment
   Use .at(index) for the value at that index

   Whilw(expression){
   Statement;
   }

   Stops when condition is false
   Checks the condition at beginning of every iteration
3) Do-while:
   Same as while but, the condition is checked at the end of every iteration

   Executed atleast once
   Do{
        Statement that has to be executed anyway:
        Eg: entering an integer for the condition of while loop or some static calculation which is then asked by while loop (enter length width andcalc area--Do|do you want to calculate the area again----while)

        Declare the condition variable for while loop outside the while loop otherwise compiler error
   }while(condition){

```
Execution statement
}; // DONT FORGET SEMICOLON
```

Examples

Monday, March 24, 2025 3:54 PM

```
#include <iostream>
#include <vector>
using namespace std;

//global variable declaration
const int PriceSmallRoom {25};
const int PriceLargeRoom {35};
const float SalesTaxRate {0.06};
const int Validity {30};

int main(){
   //local variable declaration
   int smallRoom {0};
   int largeRoom {0};
   int rawRoomPrice {0};
   float totalTax {0};

   cout<<"Welcome to Franks Cleaning Service\n";
   cout<<"Please give the No. of small rooms to be cleaned: ";
   cin>> smallRoom;
   cout<<"Please give the No. of large rooms to be cleaned: ";
   cin>> largeRoom;

   //bill structure
   cout<<"Number of small rooms: "<<smallRoom<<endl;
   cout<<"Number of large rooms: "<<largeRoom<<endl;
   cout<<"Price per small room: $"<<PriceSmallRoom<<endl;
   cout<<"Price per large room: $"<<PriceLargeRoom<<endl;
   rawRoomPrice = (smallRoom*PriceSmallRoom)+(largeRoom*PriceLargeRoom);
   cout<<"Cost: $"<<rawRoomPrice<<endl;
   totalTax = rawRoomPrice*SalesTaxRate;
   cout<<"Tax $"<<totalTax<<endl;
   cout<<"===============================\n";
   cout<<"Total Estimate = $"<<(totalTax+rawRoomPrice)<<endl;
   cout<<"This Estimate is valid for "<< Validity<<" days.";

   return 0;
}
```

Vectors example

```
#include <iostream>
#include<vector>
using namespace std;

//Global definition
```

```
int main(){
   vector<int>TestScoreStudents{1,2,3,4,5,6,7,8,9,10};
   int Test_Score_Array[10]={1,2,3,4,5,6,7,8,9,10};
   int i = 0;
   int j = 0;
   int AddValueToMyVector {0};

   for (i=0;i<=9;i++){

      cout<<"The value of student " <<i<< " is "<<Test_Score_Array[i]<<endl;
   }

   cout<<"using vectors and array output method with []. \n";
   cout<<"Please add a value to the student testscore \n";
   cin>>AddValueToMyVector;
   TestScoreStudents.push_back (AddValueToMyVector);
   cout<<TestScoreStudents.size()<<endl;

   for(j=0;j<=((TestScoreStudents.size())-1);j++){
      cout<<"The value of student "<<j<<" is "<<TestScoreStudents[j]<<endl;
   }
   cout<<"using vectors and array output method with vectorname.at() . \n";

   for(j=0;j<=9;j++){
      cout<<"The value of student "<<j<<" is "<<TestScoreStudents.at(j)<<endl;
   }

   return 0;
}
```

Challenge exercise: Vectors

```
#include <iostream>
#include <vector>
//namespace declarations
using namespace std;

//global declarations

int main(){

   //local declarations
   vector<int>vector1;
   vector<int>vector2;

   //adding 10 and 20 to vector1 dynamically
   vector1.push_back(10);
   vector1.push_back(20);

   //displaying vector1 elements using .at()
   cout<<"vector1 element 1 is: "<<vector1.at(0)<<".\n";
```

```
    cout<<"vector1 element 2 is: "<<vector1.at(1)<<".\n";
    //displaying vurrent size of vector 1:
    cout<<"The current size of vector1 is: "<<vector1.size()<<".\n";

    //adding 100 and 200 to vector2 dynamically
    vector2.push_back(100);
    vector2.push_back(200);

    //displaying vector2 elements using .at()
    cout<<"vector2 element 1 is: "<<vector2.at(0)<<".\n";
    cout<<"vector2 element 2 is: "<<vector2.at(1)<<".\n";
    //displaying vurrent size of vector 2:
    cout<<"The current size of vector2 is: "<<vector2.size()<<".\n";

    //declaring 2D vector
    vector<vector<int>>vector_2d;
    //adding vector1 to 2D vector dynamically
    vector_2d.push_back(vector1);
    //adding vector2 to 2D  vector dynamically
    vector_2d.push_back(vector2);

    //display elements in 2d vector usinf .at()
    cout<<"vector_2d row 1 elements are: "<<vector_2d.at(0).at(0)<<" "<<vector_2d.at(0).at(1)<<".\n";
    cout<<"vector_2d row 2 elements are: "<<vector_2d.at(1).at(0)<<" "<<vector_2d.at(1).at(1)<<".\n";

    //changing vector1 element (0) to 1000 using .at()
    vector1.at(0) = 1000;

    //display elements in 2d vector usinf .at()
    cout<<"vector_2d row 1 elements are: "<<vector_2d.at(0).at(0)<<" "<<vector_2d.at(0).at(1)<<".\n";
    cout<<"vector_2d row 2 elements are: "<<vector_2d.at(1).at(0)<<" "<<vector_2d.at(1).at(1)<<".\n";

    //displaying vector1 elements using .at()
    cout<<"vector1 element 1 is: "<<vector1.at(0)<<".\n";
    cout<<"vector1 element 2 is: "<<vector1.at(1)<<".\n";

    return 0;
```

```
// Change Calculator
#include <iostream>
using namespace std;

//global variable declaration
```

```
// constant denomination
/******************************************
 * 1 dollar = 100c
 * 1 quarter = 25c
 * 1 dime = 10c
 * 1 nickel is 5 cents
 * 1 penny is 1 cent
 * *****************************************/

const int kDollar {100};
const int kQuarter {25};
const int kDime {10};
const int kNickel{5};
const int kPenny {1};

int main (){
   // local declarations
   int AmountInCents {}, Balance {}, Dollar{}, Quarter{}, Dime {}, Nickel {}, Penny {};


   //enter the conversion amount in cents
   cout<< "Enter change amount in Cents: \n";
   cin>> AmountInCents;

    //dollar is no of c divided by 100
    //balance is no of c - (doller*100)
    Dollar = AmountInCents / kDollar;
    Balance = AmountInCents - (Dollar*kDollar);
    //quarter = balance divided by 25c
    //balance = balance - (quarters * 25c)
    Quarter = Balance/kQuarter;
    Balance -= Quarter*kQuarter;
    //dime = balance divided by 10
    //balance = balance - (quarter*10)
    Dime = Balance/kDime;
    Balance -= Dime*kDime;
    //nickel = balance divided by 25
    //balance = balance - (balance* nickel)
    Nickel = Balance/kNickel;
    Balance -= Nickel*kNickel;
    //penny = balance
    Penny = Balance;

    //Cout statement
    cout<< "Dollar: "<<Dollar<<endl;
    cout<< "Quarter: "<<Quarter<<endl;
    cout<< "Dime: "<<Dime<<endl;
    cout<< "Nickel: "<<Nickel<<endl;
    cout<< "Penny: "<<Penny<<endl;


   return 0;
}
```

While loop with bool control

```
//using while and do while loops
#include <iostream>
using namespace std;
//global declarations

int main(){
   //Task: Enter an integer between 1 to 5.
   //local declarations
   bool result {false};
   int numberEntered {0};

   while (!result){
      cout<<"enter an integer between 1 to 5: ";

      cin>> numberEntered;
      cout<<endl;

      if(numberEntered>=1 && numberEntered<=5){
         cout<<"if entered"<<endl;
         result = true;
      }else{
         cout<<"else entered"<<endl;
         result = false;
         cout<<"Please enter an integer value between 1 to 5!!\n";
      }
   }
   cout<<"correct value entered.\n";

   //Second way with OR Gate
      while (!result){
      cout<<"enter an integer between 1 to 5: ";

      cin>> numberEntered;
      cout<<endl;

      if(numberEntered<1 || numberEntered>5){
         cout<<"if entered"<<endl;
         cout<<"Please enter an integer value between 1 to 5!!\n";
         result = false;
      }else{
         cout<<"else entered"<<endl;
         result = true;
         cout<<"correct value entered.\n";
      }
   }
```

```
    return 0;
}
```