**# 2\_Basic Instructions for Microsoft Visual Studio C++**

**Created by : Yogendra Patil**

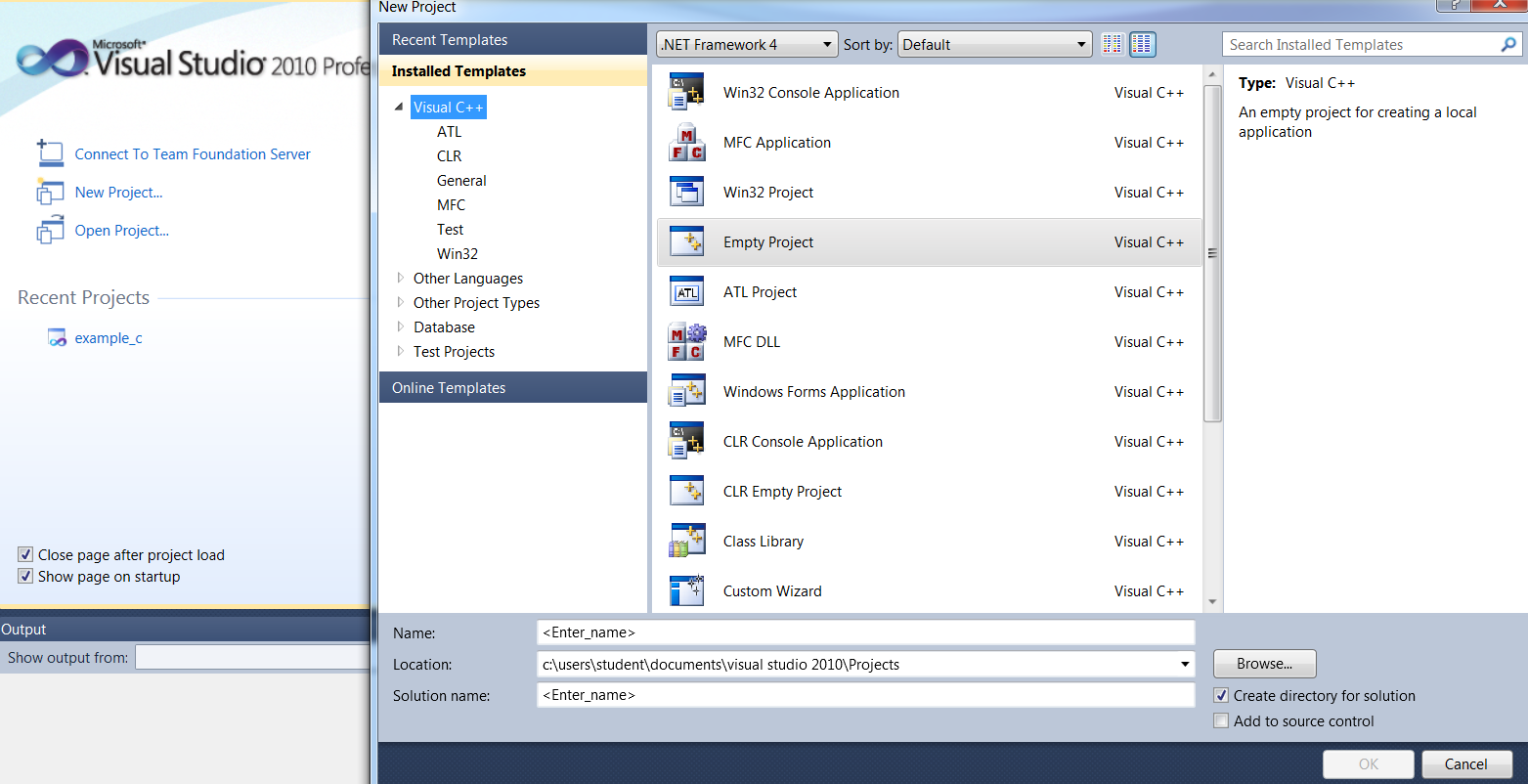
**PhD, University of Alabama**

\* LHS – Left Hand Side, M – Middle, RHS – Right Hand Side.

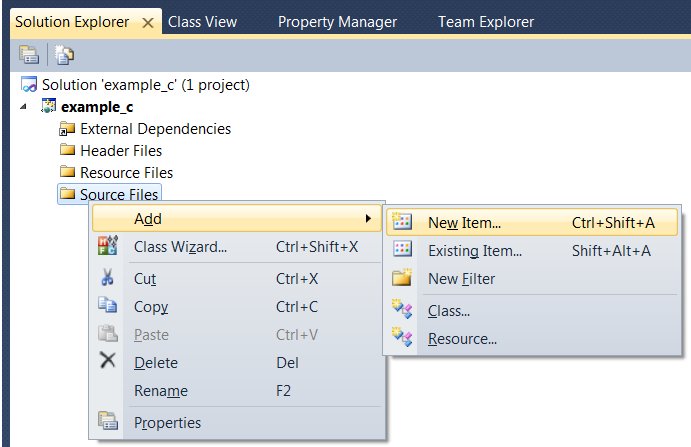
\* T – Top, C – Center, B – Bottom.

Start -> All Programs -> Microsoft Visual Studio 2010 -> Microsoft Visual Studio 2010.

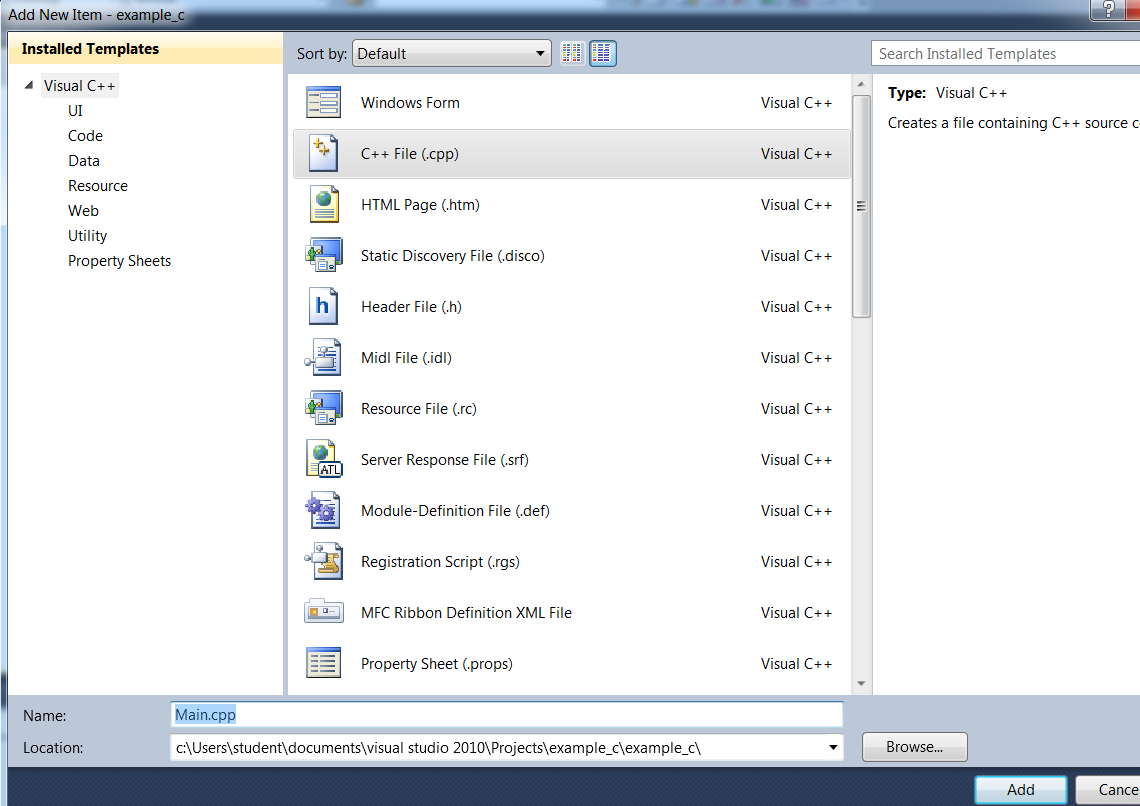
Click -> New Project… -> Visual C++ -> Empty Project Visual C++ -> (Enter the file/script name) <Enter\_name> -> OK



You will get the following window. In order to get the editor file, right click on “Source Files” -> Add -> New Item…



After that you will get a window, select “Visual C++”(LHS) -> C++ File(M) -> (Enter Name) “Main.cpp” (B) -> Add



**Code 1**:

#include <iostream> // important

using namespace std; // important

int main() {

cout << "Hello World!"<< endl; //If you don’t declare namespace std->std::cout << “Hello World!” // endl -> takes to next line

//system("pause"); // This is usually not used, So project->properties->Config Props(LHS)->Linker(LHS)->System(LHS)->SubSystem(RHS)->Console

//Press Ctrl + F5 and not F5

return 0;

}

**Code2**: “Comments, Variables and Data Types”

#include <iostream>

#include <string>

using namespace std;

int main() {

// Data Type nameOfVariable = value

//Data Types

//Integers ==> int, unsigned int, short

//Floating Types ==> double, long double, float

//Strings ==> string, we use "Hi" // include<string>

//Character ==> char, we use 'A'

//Boolean ==> bool true/false

int sample\_variable1 = 5;

double sample\_variable2 = 12.7;

string sample\_variable3 = "Hello World!";

char sample\_variable4 = 'H' ;

bool sample\_variable5 = true;

int input;

cout << "Output:" << sample\_variable5 << endl;

// Get Info from User

cout << "Please Type in a number:";

cin >> input;

cout << endl << "You Typed in :" << input << endl;

return 0;

}

**Code 3**:”Arithematic and Assignment Operators”

#include <iostream>

#include <string>

using namespace std;

int main() {

//Assignment Operators ==> =, +=, -=, \*=, /=, %=

//Arithematic (Math) Opertors ==> + - \* / % ++ --

// int (4bytes) -> 3 for #'s, 1 for sign (+,-)

//unsigned -> 4 bytes for #'s only

//Data Type Conversion:(1)Static\_Cast -> static\_cast<Datatype>(variable)

double myVariable = 1337;

int x = 7;

int y = 2;

double y1 = 2;

int x2 = 3658;

int y2 = 1000;

int x3 = 13;

int x4 = 13;

cout << "Total:" << myVariable << endl;

cout << "Total:" << (y - x) << endl; // -5

cout << "Total:" << (x/y1) << endl; // 3.5 because we use double instead of int y1

cout << "Total:" << static\_cast<double>(x)/y << endl; // Convert x to double using static\_cast temporary

cout << "Total:" << (x2 % y2) << endl; // remainder

x3 += 6;

cout << "Total:" << x3 << endl;

cout << "Total:" << x4++ << endl; // It will print 13 here, bcoz incrementing after printing

cout << "Total:" << x4 << endl; // It will print 14 here

return 0;

}

**Code4**: Math, Algorithm, Cast Conversion

#include <iostream>

#include <string>

#include <cmath> //http://www.cplusplus.com/reference/cmath/

#include <algorithm> //http://www.cplusplus.com/reference/algorithm/

using namespace std;

int main() {

double x = 8;

double y = 2;

double x1 = 3.216;

cout << "Total:" << pow(x,y) << endl;

cout << "Total:" << pow(2.0,8) << endl; // use 2.0 to indicate its double

cout << "Total:" << sqrt(x) << endl; // x should be double

cout << "Total:" << ceil(x1) << endl; // Math Function, all above

cout << "Total:" << min(x,y) << endl; // Algorithm function

// Cast

int time = 10;

int numofBottles;

cout << "How many water bottles did you recycle?";

cin >> numofBottles;

cout << "Rate:" << (static\_cast<double>(numofBottles) / time) << "per minute" << endl;

return 0;

}

**Code 5**: Strings, Getline, Concatenation, String Functions

#include <iostream>

#include <string> // include string class library to use string functions

using namespace std;

int main() {

//getline(cin,variable)

//concatenation: + +=

//Functions: .length() ==> returns number of characters in string

//.size() ==> same as length, .at(int) ==> return char at specified index # 0,1,2,...

//.append(string) ==> Concatenate string at the end of the string

//.insert(index,string, start,length)

/\*

string input;

string input1;

/\* Part I

cout << "Enter your first name: ";

cin >> input;

cout << "Hello," << input << " welcome to our tutorial"<<endl;

\*/

/\*

// Part II getline(cin,variable)

cout << "Enter your full name: ";

getline(cin,input1);

cout << "Hello," << input1 << " welcome to our tutorial"<< endl;

// Part III concatenation

// (a)

string first,last;

cout << "Enter your First Name:";

cin >> first;

cout << "Enter your Last Name:";

cin >> last;

//cout<< "Hello " << first << " " << last << ",welcome to this tutorial." << endl;

// (b)

cout<< "Hello " << first + last << ",welcome to this tutorial." << endl; // But no space between first and last, so...

//(c)

string name = first +" " + last;

cout<< "Hello " << name << ",welcome to this tutorial." << endl;

//(d), another way to avoid using new variable

first += " ";

first += last; // first has all variables

cout<< "Hello " << first << ",welcome to this tutorial." << endl;

\*/

//Part III length of the string

string var1;

cout << "Enter your name: ";

getline(cin,var1);

string var2;

cout << "Enter your last name:";

cin >> var2;

string insert = "not so much";

cout << var1.length() << endl;

cout << var1.size() << endl;

cout << var1.at(3) << endl;

cout << var1.append(var2) << endl;

cout << var1.insert(4,insert,4,2) << endl;//insert(after 4 characters, insert the variable insert, its 4th char and, (2-1) = 1 chars)

return 0;

}

**Code 6**: If statements, Comparison, Logical Operators

#include <iostream>

#include <string>

using namespace std;

int main() {

//comparison Operators: == != > < >= <=

//logical Operators: && || ^ --> XOR table(only one condition is true then execute)

// unary Operator: ! (not)

int score;

cout << "Please type in your test score:";

cin >> score;

if((score >= 90) && !(score == 100)){

cout << "You got an A, but not perfect 100!"<< endl;

}

else if(score >= 80){

cout << "You got a B!"<< endl;

}

else if(score >= 70){

cout << "You got a C!"<< endl;

}

else{

cout << "You got a F!"<< endl;

}

return 0;

}

**Code 7**: Constants, Formatting Decimal

#include <iostream>

#include <iomanip> // allows to manipulate

using namespace std;

//Constants always outside the function, dont change their format

// Typically CAPS LETTER

const double TAX\_FEDERAL = 0.15;

int main() {

/\*

// PART I -> Constants

int income;

cout << "What is your annual salary: ";

cin >> income;

cout << "You will make: " << (income - (income \* TAX\_FEDERAL)) << "this year!" << endl;

\*/

/\*

// PART II -> Formatting Decimals

cout.setf(ios::fixed); // fixed the decimal point places, cannot be changed

cout.setf(ios::showpoint);

cout.precision(2);

cout << "Type in a number:";

double input;

cin >> input;

cout << endl << input << endl;

\*/

// PART III -> Manipulate "First include the iomanip library"

cout << fixed << showpoint << setprecision(2);

cout << "Type in a number:";

double input;

cin >> input;

cout << endl << input << endl;

return 0;

}

**Code 8**: Switch statements

#include <iostream>

using namespace std;

int main() {

/\*

// PART I -> Integer Type

int score;

cout << "Type in your AP score:";

cin >> score;

switch(score){

case 5:

cout << endl << "You are very ready for college"<< endl;

break; // To reduce the time taken by the app. It wont go through all cases!! Use "break"

case 4:

cout << endl << "You are very ready for college"<< endl;

break;

case 3:

cout << endl << "You are probably ready for college"<< endl;

break;

default:

cout << endl << "You are not ready for college"<< endl;

\*/

// PART II -> Character Type

char grade;

cout << "Type in your AP score:";

cin >> grade;

switch(grade){

case 'A':

case 'a':

cout << endl << "You got a A"<< endl;

break; // To reduce the time taken by the app. It wont go through all cases!! Use "break"

case 'B':

case 'b':

cout << endl << "You got a B"<< endl;

break;

case 'C':

case 'c':

cout << endl << "You got a C"<< endl;

break;

default:

cout << endl << "You are not ready for college"<< endl;

}

return 0;

}

**Code 9**: Loops

#include <iostream>

#include <string>

using namespace std;

int main() {

string input;

cout << "Type something in: ";

cin >> input;

int i = 0;

//Loops ==> While; Do While (atleast execute once regardless of what the expression is)

//Loops ==> For

/\*

while(i < 10){

cout << input << endl;

i++;

}

\*/

/\*

do{

cout << input << endl;

i+=3;

}while(i<10); // Use semi-colon at end!!!

\*/

cout << "How many times to print? ";

int number;

cin >> number;

for(int i = 1; i < number; i+=2){

cout << i << " " << input << endl;

}

return 0;

}

**Code 10**: Nesting, Existing and Continuing Loops

#include <iostream>

#include <string>

using namespace std;

int main() {

// How to use BACKSLASH

/\*

string output = "The weather says, \"It's going to rain tomorrow!\"";

cout << output << endl;

\*/

//int first; // For if nested

string first; // For while nested

cout << "Type Something:";

cin >> first;

string second;

cout << endl << "Type Something:";

cin >> second;

/\*

// Nesting if statements

if(first > 5){

cout << endl << "Before the nested if statement" << endl;

if(first < 10){

cout << endl << "Perfect" << endl;

}

else{

cout << endl << "Too High!!" << endl;

}

cout << endl << "After the nested if statement" << endl;

}

\*/

// While , nested for

int i = 0;

while(i < 5){

cout << i << " " << first << endl;

for (int j = 0;j < 5;j++){

if(j == 2){

//continue; // skip to next iteration

break;// exists loops or code permanently

}

cout << j << " " << second << endl;

}

i++;

}

return 0;

}

**Code 11**: Functions

#include <iostream>

#include <string>

using namespace std;

//Function: DataType name\_func(){}

//Function Declarations

void welcomeMessage();//Prints a welcoming message

int calNum();//multiplies integer @return int - product of two integers

int main() {

welcomeMessage();

int product = calNum();

cout << product<<endl;

cout << calNum()<<endl;

return 0;

}

//Fucntion Definitions

void welcomeMessage(){

cout << "Welcome to our tutorial"<<endl;

}

int calNum(){

int x = 5, y = 6;

return(x \* y);

}

**Code 12**: Variable Scope and Parameters

#include <iostream>

#include <string>

using namespace std;

//Function: DataType name\_func(){}

//Function Declarations

void printMessage();//Prints a welcoming message

void printResult(int); // Print as a what is passed as a Para - num to be printer

void addNums(int,int);// add two integers, @ 1stpara - first num to added

// @ 2ndpara - second num to added

void subNums(int,int);

int main() {

printMessage();

int num1,num2;

cout << "Type in two integers: ";

cin >> num1 >> num2; // enter two numbers - two values

addNums(num1,num2);

subNums(num1,num2);

return 0;

}

//Function Definitions

void printMessage(){

// Local Scope

cout << "Math Operations" << endl;

}

void addNums(int x,int y){

printResult(x + y);

}

void subNums(int x,int y){

printResult(x - y);

}

void printResult(int total){

cout << "Total: " << total << endl;

}

**Code 13 PART I**: Passing by Value & Passing by Reference

#include <iostream>

#include <string>

using namespace std;

//Function Declarations

void value(int); // passes paramter by value

void reference(int&);//@param int& - num to be incrmented

int main() {

int num1 = 5;

value(num1);

cout << num1 << endl;

reference(num1);// dont put & when calling by reference

cout << num1 << endl;

return 0;

}

//Function Definitions

void value(int num1){

num1 += 5;

}

void reference(int& z){

z += 5;

}

**Code 13 PART II**: Passing by Value & Passing by Reference

#include <iostream>

#include <string>

using namespace std;

//Function Declarations

void askUser(int&,string&); // passing by reference

void calOutcome(int,string);//passing by value

int main() {

int score;

string name;

askUser(score,name);

calOutcome(score,name);

return 0;

}

//Function Definitions

void askUser(int& newScore,string& newName){

cout << "Please type in your name: ";

cin >> newName;

cout << "Please type in your score: ";

cin >> newScore;

}

void calOutcome(int newScore,string newName){

if(newScore > 50){

cout << "You did not fail!" << endl;

}

else{

cout << "You fail!" << endl;

}

}

**Code 14**: Functions as Parameters

#include <iostream>

#include <string>

using namespace std;

//Function Declarations

void askUserName(string&); // passing by reference

int askUserScore();

void print(string,int);

int main() {

string name;

askUserName(name);

print(name,askUserScore());//Using function as a Paramter

return 0;

}

//Function Definitions

void askUserName(string& newName){

cout << "Please type in your name: ";

cin >> newName;

cout << endl;

}

int askUserScore(){

int score;

cout << "Please enter credit score: ";

cin >> score;

cout << endl;

return score;

}

void print(string newName,int newScore){

if(newScore >= 740){

cout << endl << newName << " is in good standing!" << endl;

}

else if(newScore >= 600){

cout << endl << newName << " is in questionable standing." << endl;

}

else{

cout << endl << newName << "is not in good standing" << endl;

}

}

**Code 15**: Arrays

#include <iostream>

#include <string>

using namespace std;

//Array ==> dataType nameOfArray[];

//Constants

const int CAPACITY = 5;

int main() {

//string colors[] = {"red","orange","yellow","green","blue","purple"};

//cout << "Color:" << colors[0] << endl;

string colors[CAPACITY];

int i = 0;

int numOfElements = 0;//Keep count of elements in Array

string input;

cout << "Please type in colors you want (-1 to stop):";

cin >> input;//multiple strings can be inputed

while((input != "-1") && (i < CAPACITY)){

numOfElements++;

colors[i] = input;

i++;

cin >> input;

}

for(int j =0; j < numOfElements; j++){

cout << "Color#" <<(j+1)<< " " << colors[j] << endl;

}

return 0;

}

**Code 16**: Arrays as Parameters (Debug Has Some Error!!!)

#include <iostream>

using namespace std;

//Array ==> dataType nameOfArray[];

//Constants

const int CAPACITY = 39;

void fillArray(int[], int&);//(1)scores of student,(2)# of students,change by reference

//Arrays are always passed by reference, so dont use int[]&

void printArray(const int[], int);//just a safe way to use constant(cannot be modified)

int main() {

int scores[CAPACITY];

int numOfElements = 0; //Keep count of elements in Array

fillArray(scores,numOfElements);//Dont put brackets[], as arrays are passed as parameters, and declaration tells this already

printArray(scores,numOfElements);

return 0;

}

void fillArray(int newScores[],int& numOfElements){ //by DEFAULT Arrays are always passed by reference, so no need for &

int i = 0;

int score;

cout << "Type in scores of students(-1 to stop): ";

cin >> score;

while((score != -1) && (i < CAPACITY))

numOfElements++;

newScores[i] = score;

i++;

cin >> score;

}

void printArray(const int newScores[], int numOfElements){

cout << "Scores:";

for(int i =0; i < numOfElements; i++){

cout << newScores[i] << " " ;

}

}

**Code 17**: Multi-Dimensional Arrays

#include <iostream>

using namespace std;

//Constants

const int ROWS = 4;

const int COLS = 3;

//Functions Declarations

void fillScores(int[ROWS][COLS]);

void printScores(int[ROWS][COLS]);

int main() {

//PART I

//You have to put # of COLS, when you know the data,

/\*int scores[][COLS] = {{80,90,90},{60,70,82},{56,89,45},{78,89,65}}; //Each {} is the rows, So there are 4 {}'s

cout << scores[0][2] << endl;

\*/

//PART II

int scores[ROWS][COLS];

fillScores(scores);

printScores(scores);

return 0;

}

//Function Definitions

void fillScores(int newScores[ROWS][COLS]){

int score;

for(int i = 0;i < ROWS; i++){

cout << "Enter Score for Test#" << (i+1) << ":";

for (int j = 0;j < COLS;j++){

cin >> score;

newScores[i][j] = score;

}

}

}

void printScores(int newScores[ROWS][COLS]){

for(int i = 0; i < COLS; i++) {

cout << endl << "Student" << (i+1) << "Scores:";

for (int j = 0;j < ROWS; j ++){

cout << newScores[j][i]<<" ";

}

}

}

**Code 18**: Vectors and Vector Functions

#include <iostream>

#include <vector>

using namespace std;

//Constants

//Functions Declarations

int main() {

//Format: vector<DataType> nameofVector

//myVector.push\_back(value)==> adds an element to the end of teh vector, also resizes it

//myVector.at(index) or myVector(index) ==> returns element with given index

//myVector.size()==> returns an unsigned int equal to the number of elements

//myVector.begin()==>reads elements from first element (i.e index = 0)

//myVector.insert(myVector.begin()+integer, new value)==> add elements AFTER specified index

//myVector.erase(myVector.begin()+integer) ==> Removed elements AT specified index

//myVector.clear() ==> removes all elements in vector

//myVector.empty() ==> returns boolean value if vector is empty

vector<int> myVector;

myVector.push\_back(3);

myVector.push\_back(7);

myVector.push\_back(4);

myVector.push\_back(12);

myVector.push\_back(9);

cout << "Vector: ";

// as you use vector.size, its give a'unsigned int value', and if 'i' is not unsigned, there will be an error

for(unsigned int i = 0;i<myVector.size();i++){

cout << myVector[i] << " ";

}

myVector.insert(myVector.begin()+3,5);

cout <<endl << "Vector: ";

for(unsigned int i = 0;i<myVector.size();i++){

cout << myVector[i] << " ";

}

myVector.erase(myVector.begin()+4);

cout <<endl << "Vector: ";

for(unsigned int i = 0;i<myVector.size();i++){

cout << myVector[i] << " ";

}

if(myVector.empty()){

cout << endl<<"Is empty:";

}

else{

cout << endl<<"Is not empty:";

}

myVector.clear();

if(myVector.empty()){

cout << endl<<"Is empty:";

}

else{

cout << endl<<"Is not empty:";

}

cout << endl;

return 0;

}

**Code 19**: Vectors as Parameters

#include <iostream>

#include <vector>

using namespace std;

//Constants

//Functions Declarations

void fillVector(vector<int>&);//@params vector<int>&:very large in size in value, so you always pass by reference, unlike ARRAYS

void printVector(const vector<int>&);//const-because you are just printing and dont want to change its values

void reverse(const vector<int>&);

void printEvens(const vector<int>&);

void replace(vector<int>&);

int main() {

vector<int> myVector;

fillVector(myVector);// Remember here its not written as pass by reference

printVector(myVector);//

reverse(myVector);

printEvens(myVector);

replace(myVector);

return 0;

}

//Function Definitions

void fillVector(vector<int>& newmyVector){

cout << "Type in a list of numbers(-1 to stop): ";

int input;

cin >> input;

while(input != -1){

newmyVector.push\_back(input);

cin >> input;

}

cout << endl;

}

void printVector(const vector<int>& newmyVector){

cout << "Vector: ";

for (unsigned int i = 0;i < newmyVector.size(); i++){

cout << newmyVector[i] << " ";

}

cout << endl;

}

void reverse(const vector<int>& newmyVector){

cout << "Vector Reversed: ";

for(unsigned int i = (newmyVector.size());i > 0; i--){

cout << newmyVector[i - 1] << " ";//Make it 'i-1' in order to get an element one below

}

cout << endl;

}

void printEvens(const vector<int>& newmyVector){

cout << "Evens: ";

for(unsigned int i = 0;i < newmyVector.size(); i++){

if((newmyVector[i] % 2) == 0){

cout << newmyVector[i] << " ";//Make it 'i-1' in order to get an element one below

}

}

cout << endl;

}

void replace(vector<int>& newmyVector){

int old, replace;

cout << endl<<"Type a number to be replaced with a other number: ";

cin >> old >> replace;

for (unsigned int i = 0;i < newmyVector.size(); i++){

if(newmyVector[i] == old){

newmyVector[i] = replace;

}

}

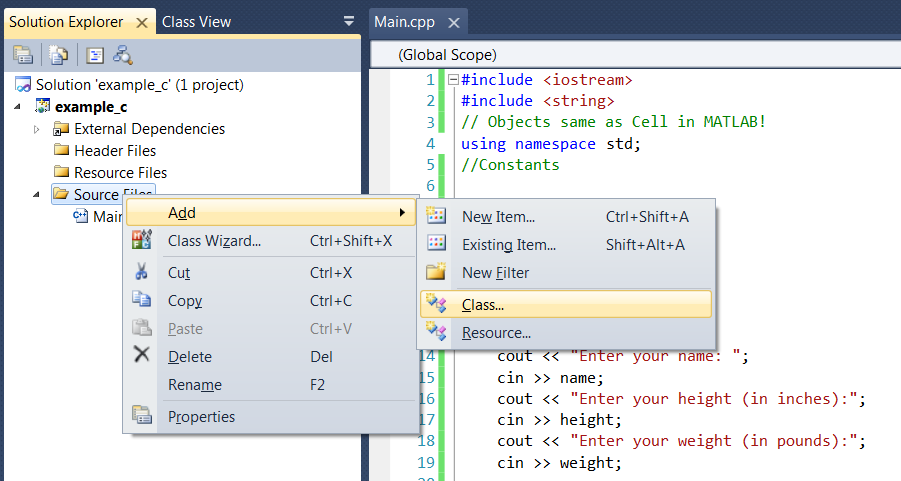
printVector(newmyVector);

cout << endl;

}

**Code 20**: Classes and Objects 1

Right click -> Source Files -> Add > Class (AUTOMATICALLY)



To add MANUALLY -> Right Click -> Source Files -> New Item… -> Then click on “C++File(.cpp)” (M) and at the bottom enter “NAME: BMI.cpp“

Now add an header file by MANUALLY -> Right Click -> Header Files -> New Item… -> Then click on “Header File(.h)” (M) and at the bottom enter “NAME: BMI.h“

//\*\*\*\*\*\*\*\*\*Header ==> Function Declarations\*\*\*\*\*\*\*\*//BMI.h

#include <iostream>

#include <string>

using namespace std;

#ifndef BMI\_H //If (header) Not DEFined

#define BMI\_H

//Declare member variable inside the class

class BMI{

public:

// Default Constructor

BMI();

// Overload Constructor

BMI(string,int,double);

// Destructor - Destroyed the used one

~BMI();

//Accesor Functions- Use of key word 'get' to access class

string getName() const;//use const Good Prog Skill

//getName(): returns name of patient

int getHeight() const;

double getWeight()const;

//Mutator Functions-Modify one member at a time

void setName(string);

void setHeight(int);

void setWeight(double);

double CalBMI();

private:

//Member variables

string newName;//As compared to other prog. lang.dont need to set string to NULL

int newHeight;// Set it to NULL

double newWeight;// Set it to NULL

};// You Do put ; for Class REMEMBER!!

#endif

//\*\*\*\*\*\*Function Definitions\*\*\*\*\*\*\*\*//BMI.cpp

#include "BMI.h"

// :: scope resolution

BMI::BMI(){

newHeight = 0;//Setting to NULL;

newWeight = 0.0;//No need for string

}

BMI::BMI(string name, int height, double weight){

newName = name;

newHeight = height;

newWeight = weight;

}

BMI::~BMI(){

}

string BMI::getName() const{

return newName;

}

int BMI::getHeight() const{

return newHeight;

}

double BMI::getWeight() const{

return newWeight;

}

void BMI::setName(string name){

newName = name;

}

void BMI::setHeight(int height){

newHeight = height;

}

void BMI::setWeight(double weight){

newWeight = weight;

}

double BMI::CalBMI(){

return ((newWeight \* 703)/(newHeight \* newHeight));

}

//\*\*\*\*\*\*\*\*\* Main cpp \*\*\*\*\*\*\*\*\*//

#include <iostream>

#include <string>

#include "BMI.h" //include the header file, that contains the Declaration of functions

// Objects same as Cell in MATLAB!

using namespace std;

//Constants

//Functions Declarations

int main() {

string name;

int height;

double weight;

cout << "Enter your name: ";

cin >> name;

cout << "Enter your height (in inches):";

cin >> height;

cout << "Enter your weight (in pounds):";

cin >> weight;

//Create Object using Overload Constructor

BMI Student\_1(name, height, weight);

cout << endl << "Patient Name:" << Student\_1.getName()<<endl <<

"Height: " << Student\_1.getHeight() << endl <<

"Weight:" << Student\_1.getWeight() << endl;

cout << endl;

//Create Object using Default Constructor

cout << "Enter your name: ";

cin >> name;

cout << "Enter your height (in inches):";

cin >> height;

cout << "Enter your weight (in pounds):";

cin >> weight;

cout << endl;

BMI Student\_2;//Go to H-file, create the Default Constr, go to cpp file and set members to NULL state;

Student\_2.setName(name);//Go to H-file, then Mutator func, cpp - set member var Name to new name

Student\_2.setHeight(height);

Student\_2.setWeight(weight);

cout << endl << "Patient Name:" << Student\_2.getName()<<endl <<

"Height: " << Student\_2.getHeight() << endl <<

"Weight:" << Student\_2.getWeight() << endl <<

"BMI:" << Student\_2.CalBMI() << endl;

cout << endl << "Student 1 name: "<<Student\_1.getName() << endl;

return 0;

}

**Code 20**: Classes as Vectors

Create Class files using header and cpp as before

//Header ==> Function Declarations

#include <iostream>

#include <string>

#include <vector>

using namespace std;

#ifndef STUDENT\_H

#define STUDENT\_H

class Student{

public:

//Defaul Constructor

Student();

//Overload Constructor

Student(string, char);

//Destructor

~Student();

//Acceassor

string getName() const;

char getGrade() const;

//Mutator

void setName(string);

void setGrade(char);

private:

//Member Variables

string newName;

char newGrade;

};

#endif

#include "Student.h"

Student::Student(){

newGrade = ' ';

}

Student::Student(string name, char grade){

newName = name;

newGrade = grade;

}

Student::~Student(){

}

string Student::getName() const{

return newName;

}

char Student::getGrade() const{

return newGrade;

}

void Student::setName(string name) {

newName = name;

}

void Student::setGrade(char grade){

newGrade = grade;

}

#include <iostream>

#include <string>

#include <vector>

#include "Student.h"

using namespace std;

//Constants

//Functions Declarations

void fillVector(vector<Student>&);//fill in students informat

void printVector(const vector<Student>&);

int main() {

//Vector of the class Student //pass the class similar to int, string

vector<Student> myClass;

fillVector(myClass);

printVector(myClass);

return 0;

}

//Function Definitions

void fillVector(vector<Student>& newMyClass){

string name;

char grade;

cout << "How many students are in your Class?";

int classsize;

cin >> classsize;

for(int i = 0;i < classsize;i++){

cout << "Enter Student Name: ";

cin >> name;

cout << "Enter Student Grade: ";

cin >> grade;

Student newStudent(name,grade);//Overload Constructor

newMyClass.push\_back(newStudent);//Pop-in, Pop-out in vector

cout << endl;

}

cout << endl;

}

void printVector(const vector<Student>& newMyClass){

unsigned int size = newMyClass.size();//unsigned to avoid future error

for(unsigned int i = 0;i < size; i++){

cout << "Student Name : " << newMyClass[i].getName()<<endl;

cout << "Student Grade : " << newMyClass[i].getGrade()<<endl;

cout << endl;

}

}

**Code 21**: Random Numbers and Static Variables

#include <iostream>

#include <ctime>//library to produce random number

using namespace std;

// Constants and Static Variables

static int x = 0;//similar to global variable

//Functions Declarations

void printNum();

int main() {

// PART I Random Variables

/\* srand(time(0));

unsigned int output = rand() % 100 + 1;

for (int i = 0; i < 175; i++){

cout << output << " ";//rand() % 5-> produce rand# between 0 and 5

output = rand() % 100 + 1;//Change the random value of output

} \*/

//PART II Static Variables

printNum();

printNum();

return 0;

}

//Function Definitions

void printNum(){

cout << (x += 5) << endl;

}

**Code 23**: Static Member Variables

//\*\*\*\*\*\*\* Bank.h \*\*\*\*\*\*\*\*\*//

#ifndef BANK\_H

#define BANK\_H

#include <iostream>

#include <string>

using namespace std;

class Bank{

public:

//Default Constructor

Bank();

//Overload

Bank(string,int, double);

//Destructor

~Bank();

//Accessor

string getName() const;

int getAccountNumber() const;

double getBalance() const;

//Mutator

void setName(string);

void setAccountNumber(int);

void setBalance(double);

void withdraw(double);

void deposit(double);

static void printBankInfo();//Prints the total Acc and Total Balance

//we use static because we use static vars 'totalAccounts & bankBalace'

private:

//Member Variables

string name;

int accountNumber;

double balance;

static int totalAccounts;//Donot initialize here, but in Bank.cpp

static double bankBalance;//Donot initialize here, but in Bank.cpp

};

#endif

//\*\*\*\*\*\*\* Bank.cpp \*\*\*\*\*\*\*\*\*//

#include "Bank.h"

//Intialization of Static Variables

int Bank::totalAccounts = 0;//Counter to count total accs in a bank

double Bank::bankBalance = 10000;//Total amount initially in Bank

Bank::Bank(){

accountNumber = 0;

balance = 0.0;

totalAccounts++;//When a new user is initialize, it means a new acc is opened!!

}

Bank::Bank(string newName, int newAccountNumber, double newBalance){

name = newName;

accountNumber = newAccountNumber;

balance = newBalance;

totalAccounts++;

bankBalance += newBalance;

}

Bank::~Bank(){//If a user to leave the bank, then

totalAccounts--;//Reduce the total bank Acc by 1

bankBalance -= balance;//Remove the balance of that particular user

}

string Bank::getName() const{

return name;

}

int Bank::getAccountNumber() const{

return accountNumber;

}

double Bank::getBalance() const{

return balance;

}

void Bank::setName(string newName){

name = newName;

}

void Bank::setAccountNumber(int newAccountNumber){

accountNumber = newAccountNumber;

}

void Bank::setBalance(double newBalance){

bankBalance -= balance;//Subtract the original balance that existed

balance = newBalance;

bankBalance += balance;

}

void Bank::withdraw(double withdraw){

balance -= withdraw;

bankBalance -= withdraw;

}

void Bank::deposit(double deposit){

balance += deposit;

bankBalance += deposit;

}

void Bank::printBankInfo(){

cout << endl << "Number of Accounts: " << totalAccounts << endl;

cout << "Total Balance: " << bankBalance << endl;

}

//\*\*\*\*\*\*\* Main.cpp \*\*\*\*\*\*\*\*\*//

#include <iostream>

#include <string>

#include "Bank.h"

using namespace std;

// Constants and Static Variables

static int x = 0;//similar to global variable

//Functions Declarations

void printBankInfo();

int main() {

Bank::printBankInfo();//Reference a class inside a main function

cout << endl;

cout << "Adam created an account and deposited 500: ";

Bank Adam("Adam",0001,500);//Overload Constructor

Bank Sarah;//Defualt Constructor

//Below the Mutator Constructor

cout << endl << "Sarah created an account and deposited 1000";

Sarah.setName("Sarah");

Sarah.setAccountNumber(0002);

Sarah.setBalance(1000);

Bank::printBankInfo();

cout << endl << "Don created and account and deposited 1500";

Bank Don("Don",0003,1500);

Bank::printBankInfo();

cout << endl << "Don set his balance to 1200";

Don.setBalance(1200);

Bank::printBankInfo();

cout << endl << "Sarah deposited 700";

Sarah.deposit(700);

Bank::printBankInfo();

cout << "Adam's account was terminated due to lack of use:";

Adam.~Bank();

Bank::printBankInfo();

return 0;

}

**Code 24**: Inheritance

**Code 25**: Pointers

#include <iostream>

using namespace std;

// Constants and Static Variables

//Functions Declarations

//Pointers - variables that store address of other vars

int main() {

// (\*) ==> dereference opertor

// \*p1 ==> returns value at that address

// \*p1 ==> returns the address of the variable to which the pointer is pointing

// &p1 ==> returns address of pointer

//int \*p1;//delcare a pointer

int num1 = 8,

\*p1 = &num1;

cout << "value: " << \*p1 << endl;//Do not forgot the '\*' sign before p1

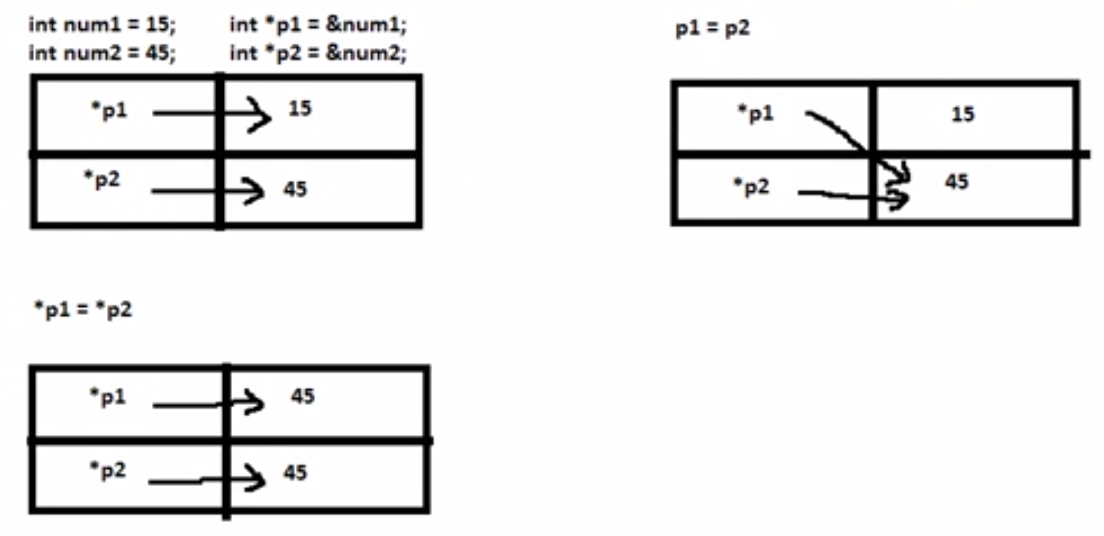
cout << "Address of Var: " << &num1 << endl;

cout << "value: " << p1 << endl;// You get the address of num1

cout << "value: " << &p1 << endl;// You get the address of num1

return 0;

}



#include <iostream>

using namespace std;

// Constants and Static Variables

//Functions Declarations

//Pointers - variables that store address of other vars

int main() {

// (\*) ==> dereference opertor

// \*p1 ==> returns value at that address

// \*p1 ==> returns the address of the variable to which the pointer is pointing

// &p1 ==> returns address of pointer

//Part I: \*p1 = \*p2

/\*int num1 = 15, num2 = 45;

int \* p1 = &num1, \*p2 = &num2;

cout << "Pointer 1: " << \*p1 << endl;

cout << "Pointer 2: " << \*p2 << endl;

\*p1 = \*p2;//\*\*change the value to which p1 is pointing to value to which p2 is pointing!!!!

cout << "Pointer 1: " << \*p1 << endl;

cout << "num1: " << num1 << endl;//\*\*Value of num1 has changed to 45!!!!!!because \*p1 = \*p2;!!!!

cout << "num2: " << num2 << endl;\*/

//Part II: p1 = p2

int num1 = 15, num2 = 45;

int \* p1 = &num1, \*p2 = &num2;

cout << "Pointer 1: " << \*p1 << endl;

cout << "Pointer 2: " << \*p2 << endl;

p1 = p2;//address in p2 i.e. num2 address assign to p1

cout << "Pointer 1: " << \*p1 << endl;

cout << "num1: " << num1 << endl;//Value of num1 is still to 15!!!!!!because p1 = p2;!!!!

cout << "num2: " << num2 << endl;

return 0;

}

**Code 26**: Dynamic Memory Allocation (DMA)

#include <iostream>

using namespace std;

//Part I: Create the variable inside a heap(special memory), to increase the life span

//Part II: Pointers as parameters

void print(int\*);//int\* indicates you are passing a pointer!!

int\* print\_return(int\*);//In order to return a value to calling function!!

int main() {

int \*p1, \*new\_p1;

p1 = new int;//point to an unknown integer,i.e. could return anyting!!This is DMA!!

\*p1 = 45;//whatever address it returns(previous step), that address will have 45 as its value

cout << "Pointer: " << \*p1 << endl;

print(p1);//Dont use the deference operator(\*), or (&) or when passing to function

new\_p1 = print\_return(p1);

cout << "New value of pointer: " << \*new\_p1 << endl;

return 0;

}

void print(int\*p1){

cout << "Function Pointer: " << \*p1 << endl;

}

int\* print\_return(int\*p1){

\*p1 = 90;

return p1;

}

**Code 27**: Dynamic Arrays

#include <iostream>

using namespace std;

const int CAPACITY = 1000;

int main() {

int \*dArray;

dArray = new int[CAPACITY];//Creating DMA arrays!!

//Main advantage is that you can allocate these large arrays in heap and the

//deallocate them as you have done using them, this saves memory space

dArray[0] = 3;

dArray[1] = 7;

dArray[2] = 2;

dArray[3] = 9;

cout << "Array: ";

for(int i = 0;i < 4; i++){

cout << dArray[i] << " ";

}

cout << endl;

delete [] dArray;//After you have done using the DMA array, you can delete it!!

//However the pointer still exits!! So use dArray = NULL;

cout << dArray[0] << endl;

dArray = NULL;

return 0;

}

**Code 28**: Recursion

// PART I: Recursive functions: Different styles of displaying output

#include <iostream>

using namespace std;

//Recursion:alternative form of loop/iteration (calling a function within that function)

void counter(int);//recursive function that counts down, int - number to increment/decrement

int main() {

int num;

cout << "Type in an integer: ";

cin >> num;

counter(num);

return 0;

}

void counter(int count){

if(count >= 1){

//cout << "Number: " << count << endl;//Part I : Ans:7 6 5 4 3 2 1 if input was 7

counter(count - 1);//dont use count--

cout << "Number: " << count << endl;//Part II : Ans:1 2 3 4 5 6 7 if input was 7

//Different style of executing in C++

}

}

// PART II: Recursive functions: Factorial

#include <iostream>

using namespace std;

//Recursion:alternative form of loop/iteration (calling a function within that function)

int counter(int);//recursive function that counts down, int - number to increment/decrement

int main() {

int num;

cout << "Type in an integer: ";

cin >> num;

cout << counter(num) << endl;

return 0;

}

int counter(int count){

if(count <= 1){

return 1;

}

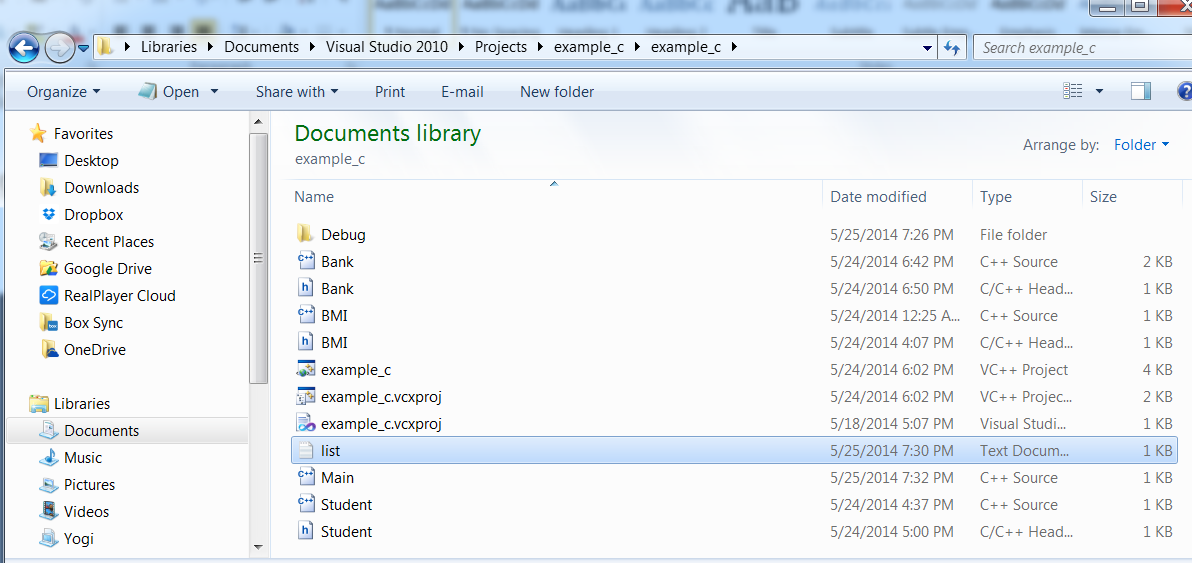
else{

return(count \* counter(count -1 ));

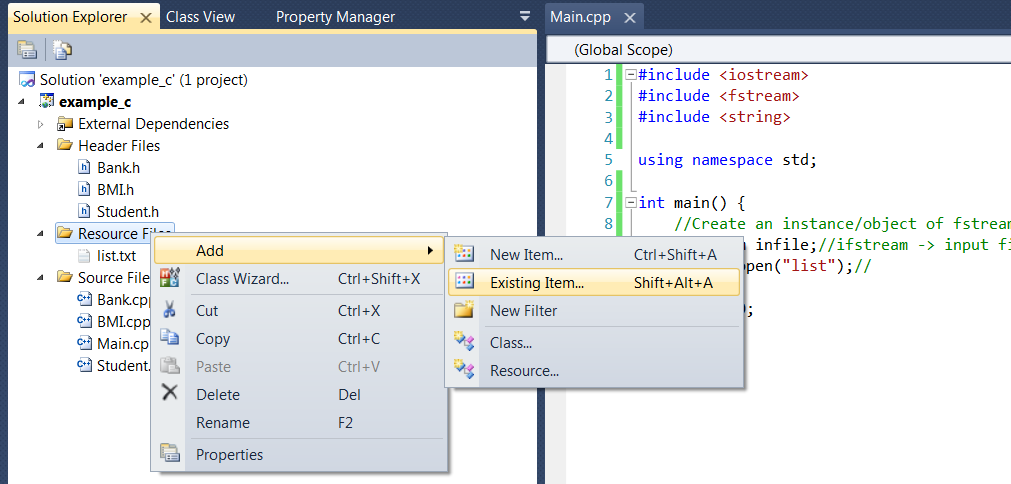
}

}

**Code 29**: Reading from a file



Create a text file called as list.



Before opening file at it to your library manually by Resource Files -> Add > Existing Item… (then select your text file). Or type in the path in Main.cpp

PART I: Print Number in the given file. A text file name “numbers.txt” was created of which first line has number 4 and second has 5.

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

//Create an instance/object of fstream class

ifstream infile;//ifstream -> input file stream, ofstream for output

infile.open("list.txt");//or you can write the path

//Check for error

if(infile.fail()){

cerr << "Error in opening file" << endl;

exit(1);

}

int x,y;

infile >> x >> y;//>> next line operator in this case

cout << "Num1: " << x << endl;

cout << "Num2: " << y << endl;

infile.close();

return 0;

}

PART II: a. Count the total items, b. Count instances of a particular item

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

//Create an instance/object of fstream class

ifstream infile;//ifstream -> input file stream, ofstream for output

infile.open("list.txt");//or you can write the path

//Check for error

if(infile.fail()){

cerr << "Error in opening file" << endl;

exit(1);

}

string item;

int count = 0;

//Read a file untill you reach the end

while(!infile.eof()){

infile >> item;

if(item=="Orange"){//Part II b

count++;

}

//count++; //Part II a

}

//cout << count << "Item Found!" << endl;//Part II b

cout << count << " Instances of Orange Found!" << endl;

infile.close();

return 0;

}

PART III: Create a file “sample.txt” using visual C++ and write to the “First number: 5”

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

//Create an instance/object of fstream class

ofstream outfile;//ofstream (predefine class) -> output file stream, ofstream for output

outfile.open("sample.txt");//Wil create automatically

outfile << "First Number: " << 5 << endl;

outfile.close();

return 0;

}