Q.3

As time passes, technology evolves in order to meet the ever-growing and rapidly changing demands of the customer. This evolution demands that the backbone behind the technology evolves as well! So, we see that programming languages like C++ and Java have evolved over the years, continually adding new features that will allow programmers to cater to the business requirements efficiently. However, there comes a point where a new language brings about a totally different approach to the problem and solves the problem more efficiently. Now at an organizational level, business owners would require their employees to work on the technology that caters to the business need most efficiently. To work on a technology that was previously being used, that now has a lower efficiency is not ideal. For instance, with Blockchain, although we may be tempted to develop these applications in our favorite languages, Subject Matter Experts would confirm that Blockchain development needs to be done in the corresponding languages in order to utilize all the necessary functions that will result in a better end- product. At the end of the day, the objective is to build the best product, and if new languages and approaches get us that result, then we must learn to adapt!

Q.4.1

Programming Design is the set of procedures that a developer/programmer must follow before they can start developing the system. Basically, the programmer provides the design of the system and other such documentations. These documents are very important for software systems as they will help other developers to maintain these systems in the future. This ensures that the functioning of a software system is not entirely dependent on the main developer. For instance, one way to go about a program design would be; Firstly, a common architectural design is developed that provides the basic architecture with the major parts of the software system and their functionalities. A detailed design is then produced which provides the exact details of each of the parts of the system. In a way, this is a modified version of the first design.

Q.4.2

While making designs in C++ two fundamental design rules that I would follow are:

1. Designing a base class that would describe the general properties of the system. And the specific classes can inherit from this one base class. For instance, if we are developing an application that involves various car models, defining one base car class with the basic properties which can then be inherited by the objects of the specific car classes(ex : Hyundai).
2. Data Abstraction is another important design rule, this allows the re-usability just as before, while at the same time, improves the privacy of a particular function/program as private data members cannot be changed/modified by functions outside the class.

Q.6

Code:

#include<iostream>

#include<string>

using namespace std;

string  change(string a, string b, string c)

{

   size\_t pos = 0;

   while((pos = a.find(b, pos))   != string::npos)

   {

       a.replace(pos,b.length(),c);

       pos += c.length();

   }

   return a;

}

int main()

{

   string first, find, replace;

   string second;

   cout<< "Enter the original string:";

   getline(cin, first);

   cout<< "Enter the string to be found:";

   getline(cin, find);

   cout << "Enter the string to replace with:";

   getline(cin, replace);

   second = change(first,find,replace);

   cout << " Original String is :" << first << endl ;

   cout << "String to be replaced is :" << find << endl;

   cout << "String to replace with is:" << replace << endl;

   cout << " Replaced String is :" << second << endl;

    return 0;

}

Output :

Text

Description automatically generated

Q.7

Code:

1. CustomStringClass.h file

#include<iostream>

#include<cstring>

using namespace std;

class CustomString{

    private:

    char\* str;

    public:

    CustomString()

    {

        str = new char[1];

        str[0] = '\0';

    }

    CustomString(char\* s)

    {

      cout <<"One Argument Constructor Called:" << endl;

      if(s == nullptr)

      {

        str = new char;

        str ='\0';

      }

      else

      {

        str = new char[strlen(s) + 1];

        strcpy(str,s);

      }

    }

    CustomString(const CustomString &s)

    {

        cout << "Copy Constructor Called:" << endl;

        str = new char[strlen(s.str) + 1];

        strcpy(str, s.str);

        str[strlen(s.str)] = '\0';

    }

    CustomString(CustomString&& s )

    {

        cout << "Move Constructor Called:" << endl;

        str = s.str;

        s.str = nullptr;

    }

    CustomString operator+(const CustomString &s)

    {

        CustomString s1;

        int k =0;

        int i =0, j=0;

        while( i < strlen(str) && j < strlen(s.str))

        {

          s1.str[k++] = str[i++];

          s1.str[k++] = s.str[j++];

        }

        if(strlen(str) > strlen(s.str))

        {

          while( i < strlen(str))

          {

            s1.str[k++] = str[i++];

          }

          s1.str[k] = '\0';

        }

        else

        {

          while(j < strlen(s.str))

          {

            s1.str[k++] = s.str[j++];

          }

          s1.str[k] = '\0';

        }

        return s1;

    }

    void GetString()

    {

      cout << "The string is :" << str << endl;

    }

    friend bool compare(CustomString& s1, CustomString& s2)

    {

        int n1 = strlen(s1.str);

        int n2 = strlen(s2.str);

        int c = n1 > n2 ? n1 : n2;

        for(int i =0; i < c ; i++)

        {

          if(c == n1 && i == n2 )

           return false;

          else if(c == n2 && i == n1)

           return false;

          else if(s1.str[i] != s2.str[i])

           return false;

        }

        return true;

    }

    ~CustomString()

    {

      delete str;

    }

};

1. Q7.cpp file

#include<iostream>

#include<string>

#include"CustomStringClass.h"

using namespace std;

int main()

{

    char\* a = new char[50];

    cout<< "Enter the first string:" << endl;

    cin >> a;

    CustomString s(a); // Calling one argument constructor

    s.GetString();

    char\*b = new char[50];

    cout << "Enter the second string:" << endl;

    cin >> b;

    CustomString s1(b); // Calling one argument constructor

    s1.GetString();

    cout << "Comparison :" << compare(s,s1) << endl; // Calling Compare function

    CustomString s2 = s + s1; // Calling overload function

    s2.GetString();

    CustomString s3(s); // Calling copy constructor with first object

    s3.GetString();

    CustomString s4 = move(s3); // Calling move constructor with previously created object

    return 0;

}

Output:

Text

Description automatically generated

Text

Description automatically generated

Q.8

Code:

#include<iostream>

#include<string>

using namespace std;

class Exception

{

private:

string m;

public:

Exception(const string& msg)

{

    m = msg;

}

 string getMessage()

 {

     return m;

 }

};

void t()

{

    throw(Exception("ArrayIndexoutofBoundsException"));

}

int main()

{

   int n, b;

   int\* a;

   cout << " Enter the no. of elements in the array:" << endl;

   cin >> n;

   a = new int[n];

   for(int i =0; i< n; i++)

   {

     cout << "Enter the element:" << endl;

     cin >> a[i];

   }

   try

   {

     cout<< "Enter the index to access:";

     cin >> b;

     if(b < 0 || b >= n)

      t();

    else

      cout << "Element is:" << a[b] << endl;

   }

   catch(Exception& e)

   {

       cout << e.getMessage() << endl;

   }

    return 0;

}

Output:

Text

Description automatically generated