Şafak AKINCI
152120141003

12.12.2016

Experiment 5
Classes and Objects-II

Objectives

To write simple computer programs using classes and objects.

Prelab Activities

Programming Output

1- **6:22:9 PM**

2- **3:04:05 AM**
03:4:5 AM
03:4:5 PM

3- **61**

4- **22**
52

5- **444**

CorrectTheCode

6-	4	:	`Time(int, int =0, int =0)`	:	compilation error
7-	4	:	`Q (int)`	:	compilation error
8-	9	:	`int qData;`	:	compilation error
9-	1	:	`void Q::setQ( int input)`	:	compilation error
10-	1	:	`void Time::setHour( int hh)`	:	logic error
	5	:	`return hour;`	:	logic error
11-	4	:	`(*clockPtr).printUniversal();`	:	compilation error
12-	17	:	`Increment::Increment( int c, int i)`	:	compilation error
13-	7	:	`void setTime(int, int, int)`	:	logic and compilation error
	9	:	`void setHour(int)`	:	logic and compilation error
	12	:	`void setMinute(int)`	:	logic and compilation error
	15	:	`void setSecond(int)`	:	logic and compilation error
14-	19	:	`void newDay();`	:	compilation error
15-	5	:	`Time( int =0, int =0, int =0)`	:	compilation error
16-	3	:	`return count;`	:	compilation error

Lab Exercises

Lab Exercise 1 – StringBuilder

```
/*****************************************
 * StringBuilder.h                       *
 *****************************************
 * IDE : Xcode                           *
 * Author : Şafak AKINCI                 *
 * Experiment 5: Classes And Objects-II  *
 *****************************************/

#include <sstream>                  //  To create an object from sstream class.
#include <vector>                   //  To create an object from vector class.
using namespace std;                //  To use cin and cout functions under the std namespace.
```

```
//Performs string appending, removing operations for some primitive data types (int, float, double, char)
class StringBuilder
{
    stringstream stream;        //stream objects is created from stringstream class.

public:
    //All append functions are used to perform string appending operation.
    StringBuilder& Append (const string& data);
    StringBuilder& Append (int data);
    StringBuilder& Append (float data);
    StringBuilder& Append (double data);
    StringBuilder& Append (char data);
    StringBuilder& AppendNewLineCharacter ();

    //Removes the content of the stream object.
    StringBuilder& Clear ();

    //Checks the stream object contents and returns true if given string is located in the stream, otherwise false.
    bool Contains (const string& str);

    //Removes the character at the given index.
    StringBuilder& RemoveAt (int index);

    //Returns the content of the stream object
    string GetData ();

    //Removes all characters which is equal to the given character in the stream objects.
    StringBuilder& Remove (char removeChar);

    //Removes characters starting from the given index until given charCount removed.
    StringBuilder& Remove (int startIndex, int charCount);

    //Removes all occurrences of the given string from the stream.
    StringBuilder& Remove (const string& removeString);

    //Replaces all occurrences of the given string by the newstring.
    StringBuilder& Replace (const string& replacedString, const string& newString);

    //Splits the
    vector<string> Split(const string& character);

    //Adds the given string to stringVector.
    void Join (const vector<string>& stringVector, const string& mergeCharacter);

    StringBuilder();
    ~StringBuilder();
};
```

```cpp
/*****************************************
 * StringBuilder.cpp                     *
 *****************************************
 * IDE : Xcode                           *
 * Author : Şafak AKINCI                 *
 * Experiment 5: Classes And Objects-II  *
 *****************************************/

#include "stringBuilder.h"          //Include header file to know function prototypes.
#include <iostream>                 //To use standart input output functions like cin and cout.

//Constructor function of StringBuilder class.
StringBuilder::StringBuilder()
{

}//StringBuilder()

//Destructor function of StringBuilder class.
StringBuilder::~StringBuilder()
{

}//~StringBuilder()

//Appends given string called data to the "stream" that is created stringstream class.
StringBuilder& StringBuilder::Append (const string& data)
{
    stream<<data;

    //Function's return type is StringBuilder& (Th reference of the object that is created by StringBuilder class).
    //This means that function will return the object(not its copy) that called this function.
    //Returning object enables using cascade functions.

    return (*this);
}

//Appends given int number called data to the "stream" that is created stringstream class.
StringBuilder& StringBuilder::Append (int data)
{
    stream<<data;

    //Function's return type is StringBuilder& (Th reference of the object that is created by StringBuilder class).
    //This means that function will return the object(not its copy) that called this function.
    //Returning object enables using cascade functions.

    return (*this);
}
```

```cpp
//Appends given float number called data to the "stream" that is created stringstream class.
StringBuilder& StringBuilder::Append (float data)
{
    stream<<data;

    //Function's return type is StringBuilder& (Th reference of the object that is created by StringBuilder class).
    //This means that function will return the object(not its copy) that called this function.
    //Returning object enables using cascade functions.

    return (*this);
}

//Appends given double number called data to the "stream" that is created stringstream class.
StringBuilder& StringBuilder::Append (double data)
{
    stream<<data;

    //Function's return type is StringBuilder& (Th reference of the object that is created by StringBuilder class).
    //This means that function will return the object(not its copy) that called this function.
    //Returning object enables using cascade functions.

    return (*this);
}
```

```cpp
//Appends given char called data to the "stream" that is created stringstream class.
StringBuilder& StringBuilder::Append (char data)
{
    stream<<data;

    //Function's return type is StringBuilder& (Th reference of the object that is created by StringBuilder class).
    //This means that function will return the object(not its copy) that called this function.
    //Returning object enables using cascade functions.

    return (*this);
}

//Appends a new line character to the "stream" that is created stringstream class.
StringBuilder& StringBuilder::AppendNewLineCharacter ()
{
    stream<<endl;               //stream<<"\n";

    //Function's return type is StringBuilder& (Th reference of the object that is created by StringBuilder class).
    //This means that function will return the object(not its copy) that called this function.
    //Returning object enables using cascade functions.

    return (*this);
```

```
}

//Erases the contents of the string, which becomes an empty string (with a length of 0 characters).
StringBuilder& StringBuilder::Clear ()
{
    //The str function of stream takes string parameter and assigns it to stream.
    stream.str("");             //string s1;        stream.str(s1);

    //Function's return type is StringBuilder& (Th reference of the object that is created by StringBuilder class).
    //This means that function will return the object(not its copy) that called this function.
    //Returning object enables using cascade functions.

    return (*this);
}

//Checks if the string contains the given string, if it is returns true.
bool StringBuilder::Contains (const string& str)
{
    //npos is a static member constant value with the greatest possible value for an element of type size_t.
    //As a return value, it is usually used to indicate no matches.

//size_t is an alias of one of the fundamental unsigned integer types.
//It is a type able to represent the size of any object in bytes.
    //The meaning of pos is "Position of the first character in the string" to be considered in the search.

    //If the given string is found, find function returns the pos the type of size_t.
    if(stream.str().find(str) != string::npos)
        return true;

    return false;
}

//Removes the character at the given index.
StringBuilder& StringBuilder::RemoveAt (int index)
{

    string s1 = stream.str();

    //Erase function erases the character according to the given position.
//If used like s1.erase(index), it will erase from given index to the end of string.
    s1.erase(s1.begin()+index);
    stream.str(s1);

    //Function's return type is StringBuilder& (Th reference of the object that is created by StringBuilder class).
    //This means that function will return the object(not its copy) that called this function.
    //Returning object enables using cascade functions.
```

```
    return (*this);
}

//Removes all characters which is equal to the given character in the stream objects.
StringBuilder& StringBuilder::Remove (char removeChar)
{
    //Gets data and assigns it to s1.
    string s1 = stream.str();

    //If the given character is found, find function returns the pos the type of size_t.
    size_t pos = s1.find(removeChar);

    while(pos != string::npos)              //If there is given character.
    {
        s1.erase( s1.begin()+pos );         //Erase the character at given position.
        pos = s1.find(removeChar);
    }


    stream.str(s1);

    //Function's return type is StringBuilder& (Th reference of the object that is created by StringBuilder class).
    //This means that function will return the object(not its copy) that called this function.
    //Returning object enables using cascade functions.

    return (*this);
}

//Removes charCount times character starting with startIndex.
StringBuilder& StringBuilder::Remove(int startIndex, int charCount)
{
    //Gets data and assigns it to s1.
    string s1 = stream.str();

    //Erases charCount times characters starting with given position.
    s1.erase(startIndex, charCount);

    //s1 is assigned to data not appended.
    stream.str(s1);

    //Function's return type is StringBuilder& (Th reference of the object that is created by StringBuilder class).
    //This means that function will return the object(not its copy) that called this function.
    //Returning object enables using cascade functions.

    return (*this);
}
```

```
//Removes given string(if there is more than one string, then remove all of them)  from data.
StringBuilder& StringBuilder::Remove (const string& removeString)
{
    //Gets data and assigns it to s1.
    string s1 = stream.str();

    //If the given character is found, find function returns the pos the type of size_t.
    size_t startIndex = s1.find(removeString);

    while(startIndex != string::npos)
    {
        //Erases removeString.length() times character starting with startIndex # s1.find(removestring) #
        s1.erase(startIndex, removeString.length());
        startIndex = s1.find(removeString);
    }

    //s1 is assigned to data not appended.
    stream.str(s1);

    //Function's return type is StringBuilder& (Th reference of the object that is created by StringBuilder class).
    //This means that function will return the object(not its copy) that called this function.
    //Returning object enables using cascade functions.
    return (*this);
}


//Removes replacedString.length() times character starting with startIndex from s1,
//and copy the given string #newString# starting with startIndex.
StringBuilder& StringBuilder::Replace (const string& replacedString, const string& newString)
{
    //Gets data and assigns it to s1.
    string s1 = stream.str();

    //If the given string is found, find function returns the pos the type of size_t.
    size_t startIndex = s1.find(replacedString);

    while(startIndex!= string::npos)
    {
        //Removes replacedString.length() times character starting with startIndex from s1,
        //and copy the given string #newString# starting with startIndex.
        s1.replace(startIndex, replacedString.length(), newString);
        startIndex = s1.find(replacedString);
    }

    //s1 is assigned to data not appended.
```

```
    stream.str(s1);

    //Function's return type is StringBuilder& (Th reference of the object that is created by StringBuilder class).
    //This means that function will return the object(not its copy) that called this function.
    //Returning object enables using cascade functions.

    return (*this);
}

//Returns the string data which is encapculated in stream.
string StringBuilder::GetData ()
{
    return stream.str();
}

//Splits the content of the string builder into parts according to the given split character string.
vector<string> StringBuilder::Split(const string& character)
{
    //Function will split the data as words, and they will store in a matrix, but we don't know how many words our
data have.
    //We can store them into dynamic matrix, and vector class helps us here to store values without knowing the size
of the matrix.
    //We also know the matrix's size using size function ( return_value.size() ).
    vector<string> return_value;

//Gets data and is assigned to content.
string content = stream.str();

//The index of given character at the string.
int idx;

//If find function doesn't find the place of given character, it will return -1.
while( ( idx=content.find(character) ) != -1)
{
    //substr function returns a sub-string of content that starts from 0 and ends with idx(end of the word).
    string word = content.substr(0,idx);

    //After the sub-string is got, erase this word from content.
    content.erase(0,idx+1);

    //If word (the type of string) is full, push it to return_value using vector's push_back function.
    if(!word.empty())
        return_value.push_back(word);
}

//If there is still word in the content, (it is the last word) push it to return_value.
if( ! content.empty() )
```

```
        return_value.push_back(content);

    //Returns a string vector not its reference.
    //Because, return_value is a local variable of this function, after it finishes, local variables will be removed.
    return return_value;
}

//Joins the given vector members by the merge character string.
void StringBuilder::Join(const vector<string>& stringVector, const string& mergeCharacter)
{
    //Clear the data.
    (*this).Clear();

    //Appends mergeCharacter to data after the each word that are stored in the stringVector.
    for(int i=0; i<stringVector.size(); i++)
    {
        stream<<stringVector[i];
        stream<<mergeCharacter;
    }
}

/*****************************************
 * StringBuilderTestMain.cpp             *
 *****************************************
 * IDE : Xcode                           *
 * Author : Şafak AKINCI                 *
 * Experiment 5: Classes And Objects-II  *
 *****************************************/

#include "stringBuilder.h"          //Include header file to know function prototypes.
#include <iostream>                 //To use standart input output functions like cin and cout.
using namespace std;                //To use cin and cout functions under the std namespace.

void TEST_Append(StringBuilder& sb)
{
    cout << "+-------------+" << endl
         << "| APPEND TEST |" << endl
         << "+-------------+" << endl;

    //Appends given data to stream using cascade functions.
    sb.Append("TestName1").Append(" ").Append("TestSurname1").Append(" ").Append(25).Append("
").Append(45.2).AppendNewLineCharacter();

    //Appends given data to stream using cascade functions.
    sb.Append("TestName2").Append(" ").Append("TestSurname2").Append(" ").Append(24).Append("
").Append(47.45).AppendNewLineCharacter();
```

```
    //Prints the data to console.
    cout << sb.GetData() << endl;

    //Clear the data.
    sb.Clear();
}

void TEST_Clear(StringBuilder& sb)
{
    cout << "+-------------+" << endl
         << "| CLEAR TEST  |" << endl
         << "+-------------+" << endl;

    //Clear the data.
    sb.Clear();

    //Prints the data to console.
    cout << sb.GetData() << endl;
}

void TEST_Contains(StringBuilder& sb)
{
    cout << "+---------------+" << endl
         << "| CONTAINS TEST  |" << endl
         << "+---------------+" << endl;

//Appends given string to data.
sb.Append("ESOGU");

//If the given string is found, contains function will return true.
if (sb.Contains("OGU"))
{
    cout << "StringBuilder contains the string 'OGU'" << endl;
}
else
{
    cout << "StringBuilder does not contain the string 'OGU'" << endl;
}

//If the given string is found, contains function will return true.
if (sb.Contains("Bilgisayar"))
{
    cout << "StringBuilder contains the string 'Bilgisayar'" << endl;
}
else
{
    cout << "StringBuilder does not contain the string 'Bilgisayar'" <<endl;
```

```
    }

    //Clear the data.
    sb.Clear();
}

void TEST_RemoveAt(StringBuilder& sb)
{
    cout<< "+-----------------+" << endl
        << "| REMOVE AT TEST  |" << endl
        << "+-----------------+" << endl;

    //Appends the given string to the data.
    sb.Append("ESOGU Bilgisayar");

    //Remove the character that is placed at given index.
    sb.RemoveAt(1);

    //Prints the data to console.
    cout << sb.GetData() << endl;

    //Clear the data.
    sb.Clear();
}

void TEST_RemoveChar(StringBuilder& sb)
{
    cout << "+----------------------+" << endl
         << "| REMOVE CHARACTER TEST  |" << endl
         << "+----------------------+" << endl;

    //Appends given string to the data.
    sb.Append("ESOGU Bilgisayar");

    //Removes the given character from data.
    sb.Remove('a');

    //Prints data to console.
    cout << sb.GetData() << endl;

    //Clear the data.
    sb.Clear();
}

void TEST_RemoveRange(StringBuilder& sb)
{
    cout<< "+-------------------+" << endl
```

```
        << "| REMOVE RANGE TEST  |" << endl
        << "+--------------------+" << endl;
    //Appends given string to the data.
    sb.Append("ESOGU BilgisayarBilgisayar");

    //Removes the 10 characters starting with 16 index.
    sb.Remove(16, 10);

    //Prints the data to console.
    cout << sb.GetData() << endl;

    //Clear the data.
    sb.Clear();
}

void TEST_RemoveString(StringBuilder& sb)
{
    cout <<"+--------------------+" << endl
        << "| REMOVE STRING TEST  |" << endl
        << "+--------------------+" << endl;

    //Appends given strings to data.
    sb.Append("ESOGU Bilgisayar, ESOGU Bilgisayar, ESOGU Bilgisayar");
    cout << "Before Remove : " << sb.GetData() << endl;

    //Removes given string from data.
    sb.Remove("Bilgisayar");
    cout << "After Remove : "  << sb.GetData() << endl;

    //Clear the data.
    sb.Clear();
}

void TEST_Replace(StringBuilder& sb)
{
    cout<< "+----------------------+" << endl
        << "| REPLACE STRING TEST  |" << endl
        << "+----------------------+" << endl;

    //Clear the data.
    sb.Clear();

    //Appends given string to data.
    sb.Append("ESOGU BILGI, ESOGU BILGI, ESOGU BILGI");
    cout << "Before Replace : " << sb.GetData() << endl;

    //Replaces given two strings.
```

```
    sb.Replace("BILGI", "Bilgisayar");
    cout << "After Replace : " <<sb.GetData() << endl;

    //Clear the data.
    sb.Clear();
}

void TEST_QUIZ(StringBuilder& sb)
{
    cout<<"+----------------------+"<<endl
        <<"| Test Sentence Content |"<<endl
        <<"+----------------------+"<<endl;

    //Appends given string to data.
    sb.Append("This   is a   test sentence for     the execution   that    ");

    //Split contents according to space, and function returns string vector.
    vector<string> rv = sb.Split(" ");
    for(int i=0; i<rv.size(); i++)
        cout<<i<<". word: "<<rv[i]<<endl;

    cout<<"+-----------+"<<endl
        <<"| Join Test |"<<endl
        <<"+-----------+"<<endl;

    //Merge , after the each word of the data.
    sb.Join(rv, ",");

    //Prints data to console.
    cout<<sb.GetData();
}

int main()
{
    StringBuilder strBuilder;

    TEST_Append(strBuilder);
    TEST_Clear(strBuilder);
    TEST_Contains(strBuilder);
    TEST_RemoveAt(strBuilder);
    TEST_RemoveChar(strBuilder);
    TEST_RemoveRange(strBuilder);
    TEST_RemoveString(strBuilder);
    TEST_Replace(strBuilder);
    TEST_QUIZ(strBuilder);

    return 0;
```

```
}//end main ()
```

Conclusion

→ Used an object that is created by another class(stringstream) as a member data of class.
→ Learnt stringstream class and its member functions like, str(), erase(), find() …
→ Learnt function overloading, compiler decides according to the given parameters.
→ Vector class is also used, it is useful to get any type of dynamic matrix when we don’t know the size of it.
→ Vector class’s functions are also used, like size(), push_back(), popBack()