Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Fall 2022

Homework 6 – BitRepresentation

Due: 26/12/2022, Monday, 11:55 PM

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You HAVE TO write down the code on your own.
You CAN NOT HELP any friend while coding.
Plagiarism will not be tolerated!!

Submission rules (PLEASE READ, IMPORTANT)

Your submission will consist of only TWO code files:

1. `BitRepresentation.h`
2. `BitRepresentation.cpp`

Please don't change the names of these files.

Don't upload an XCode project or a Visual Studio project. Just the two code files above.

Place these two code files AND ONLY THESE TWO FILES inside a folder named with the following convention:

```
SUCourseUserName_YourLastname_YourName_HWnumber
```

Your SUCourse user name is your SUNet username that is used for your sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the folder name. For example, if your SUCourse username is cago, your first name is Taha Çağlayan, and your last name is Özbugsızkodyazaroğlu, then the folder name must be:

```
cago_Ozbugsizkodyazaroglu_TahaCaglayan_1
```

Compress this folder using a compression program such as WinZip or WinRAR. Please use "zip" compression. "rar", "7z" or any other compression mechanisms are NOT allowed. Our homework processing system only works with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains all your code files. You will receive no credits if your compressed folder does not expand or it does not contain the correct files. The name of the zip file follows the same convention. The zip file for the homework submission by the student mentioned above would be:

```
cago_Ozbugsizkodyazaroglu_TahaCaglayan_1
```

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.). Successful submission is one of the requirements of the homework. If for some reason you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Overview

You will write a templated class BitRepresentation.

```
template <typename T>
class BitRepresentation{
  //...
};
```

Each instance of this class will store an integer number (e.g., short, char, unsigned short, long long, etc.). The object will allow users to manipulate the bits of this number and get its binary and hexadecimal representations.

The class will have one data member, called “data” with the type T (templated type). “data” is the number stored inside the BitRepresentation object.

The following are the member functions you need to implement:

BitRepresentation(T number)

This constructor will create a BitRepresentation object setting the data member to “number”.

Example usage

```
int a = 10;
BitRepresntation<int> x(a);
short s = 100;
BitRepresentation<short> y(s);
```

BitRepresentation(vector<bool>)

This constructor will take a vector of booleans and will translate it into a binary number, with “true” being a 1, and “false” being a 0.

The last element (rightmost element) in the vector corresponds to the least-significant bit

(LSB), and the element before it is the next bit, and so on until you go through the entire vector or you assign all the bits of “data”.

If there are fewer elements in the vector than there are bits in the member “data”, the remaining bits are assumed to be 0. If there are more elements in the vector than there are bits in “data”, the extra bits are discarded.

Example usage

```
#include "BitRepresentation.h"
#include <vector>
using namespace std;

int main() {
   vector<bool> v1 = {true, false, true, false}; // == 0b1010
   BitRepresentation<char> s1(v1); // s1.data = 0b00001010

   vector<bool> v2 = {true, true, true, true, false, false, false, false};
   BitRepresentation<char> s2(v2); // s2.data = 0b11110000

   vector<bool> v3 = {true, false, false, true, true, true, true, true, true,
false, false, false, false};
   BitRepresentation<char> s3(v3); // s3.data = 0b11110000
   // since the char data type has only 8 bits, only the last 8
   // booleans in v3 were used to initialize s3.data.
   // The rest (i.e. the first 5 elements of v3) were discarded.

   return 0;
}
```

string getBinary()

Returns the binary representation of “data” as a string. The string must contain all the bits of the number. So, if “data” is of type char, the string will have 8 characters. if “data” is of type short, the string will have 16 characters.

Example usage

```
#include "BitRepresentation.h"
#include <iostream>
using namespace std;

int main() {
   BitRepresentation<char> x(0);
   cout << x.getBinary() << endl;
   BitRepresentation<char> y(3);
   cout << y.getBinary() << endl;
   return 0;
}
```

Output:

```
00000000
00000011
```

string getHexadecimal()

Returns the hexadecimal representation of “data” as a string (the letters should be capital). The string must contain all the bits of the number. So, if “data” is of type char, the string will have 2 characters. if “data” is of type short, the string will have 4 characters.

Example usage

```
#include "BitRepresentation.h"
#include <iostream>
using namespace std;

int main() {
   BitRepresentation<short> x(0);
   cout << x.getHexadecimal() << endl;
   BitRepresentation<short> y(12351);
   cout << y.getHexadecimal() << endl;
   return 0;
}
```

Output

```
0000
303F
```

void setBit(unsigned int location)

Sets the bit at location “location” to 1. Location starts counting at the LSB. So location = 0 means set the least significant bit. If the number has fewer bits than “location”, the function will do nothing.

```
#include "BitRepresentation.h"
#include <iostream>
using namespace std;

int main() {
   BitRepresentation<char> x(0);
   x.setBit(0);
   cout << x.getBinary() << endl;
   x.setBit(1);
   cout << x.getBinary() << endl;
   x.setBit(6);
   cout << x.getBinary() << endl;
```

```
    x.setBit(8); // function will do nothing in this case
    cout << x.getBinary() << endl;
    return 0;
}
```

Output

```
00000001
00000011
01000011
01000011
```

void unsetBit(unsigned int location)

Sets the bit at location “location” to 0. Location starts counting at the LSB. So location = 0 means unset the least significant bit. If the number has fewer bits than “location”, the function will do nothing.

Example usage

```
#include "BitRepresentation.h"
#include <iostream>
using namespace std;

int main() {
    BitRepresentation<char> x(255);
    cout << x.getBinary() << endl;
    x.unsetBit(0);
    cout << x.getBinary() << endl;
    x.unsetBit(1);
    cout << x.getBinary() << endl;
    x.unsetBit(6);
    cout << x.getBinary() << endl;
    x.unsetBit(8); // function will do nothing in this case
    cout << x.getBinary() << endl;
    return 0;
}
```

Output

```
11111111
11111110
11111100
10111100
10111100
```

bool isBitSet(unsigned int location)

Returns true if the bit at location “location” in “data” is 1. Returns false otherwise. If there are fewer bits than “location”, returns false.

Example usage

```
#include "BitRepresentation.h"
#include <iostream>
using namespace std;

int main() {
   BitRepresentation<char> x(255);
   x.unsetBit(0);
   x.unsetBit(1);
   x.unsetBit(6);
   cout << x.getBinary() << endl;
   cout << x.isBitSet(0) << endl;
   cout << x.isBitSet(2) << endl;
   cout << x.isBitSet(6) << endl;
   cout << x.isBitSet(11) << endl;
   return 0;
}
```

Output:

```
10111100
0
1
0
0
```

T getNumber()

Returns the data member “data”.

Example usage

```
#include "BitRepresentation.h"
#include <iostream>
using namespace std;

int main() {
   BitRepresentation<short> s(0);
   s.setBit(0);
   s.setBit(1);
   s.setBit(3);
   short number = s.getNumber();
   cout << number << endl;
   return 0;
}
```

Output

```
11
```

Reminder about sizeof

Remember that you can get the number of bytes in a data type using the sizeof() function. Similarly, you can get the size of a templated data type using the same function. See the following example:

```
#include "BitRepresentation.h"
#include <iostream>
using namespace std;

template <typename T>
void get_size(){
   cout << "Size of T " << sizeof(T) << endl;
}

int main() {
   get_size<int>();
   get_size<char>();
   get_size<short>();
   get_size<unsigned long>();
   return 0;
}
```

Output

```
Size of T 4
Size of T 1
Size of T 2
Size of T 4
```

Also, remember that each byte is made up of 8 bits.

Rules and assumptions

1. When implementing your templated class, include BitRepresentation.cpp at the bottom of BitRepresentation.h, inside the header guard. In other words, follow Solution 2 of the Fundamental Dilemma (For more information, check slide set “5.1-Templated.ppt” slides 22-23).
2. You may assume that the data types used as template arguments in BitRepresentation are always going to be integer data types.

3. You may add extra data members and member functions to the BitRepresentation class.
4. You may add extra free functions to your solution as long as you declare them in the BitRepresentation.h file, and define (implement) them in the BitRepresentation.cpp file.

Good Luck!

Amro