Large Integer Data Type Implementation in C++ Programming Language

Mehmet Sahin

Mentor: Anna Salvati
CIS Department
Fall 2018

Motivation

- Primitive data types in C++:
  - char, short, int, bool, float and double
- Create a new data type to make it possible to operate on very large integers.

Background

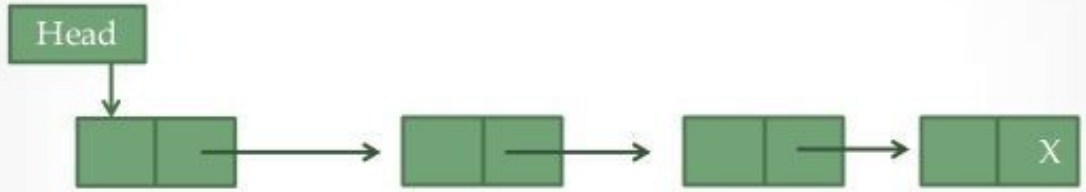
1) Double Linked List

2) Operator Overloading

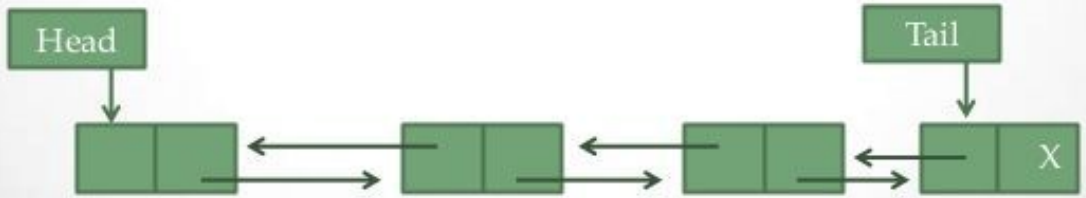
Double Linked List

Linked Lists

- Single Linked List



- Double Linked List



Operator Overloading

ASSIGNMENT OPERATOR	ARITHMETIC OPERATOR	RELATIONAL OPERATORS	LOGICAL OPERATORS	BITWISE OPERATORS	INCREMENT & DECREMENT OPERATOR	SPECIAL OPERATOR
=	+ - * / % += -+ *= /= %=	< <= > >= == !=	! && ||	& | ~ ^ << >> <<= >>= &= = ^+	++ --	, [ ] ( ) -> ->* new new [ ] delete delete[ ]

Approach

- I created a new data type called LargeInt
  - Store data in doubly linked list
  - Manipulate data using operators (+, - etc.)
  - Test each of new functionalities within the C++ to make sure it is running without error.
  - Save the data type on GitHub so that it can be accessed by my Mentor and anyone else who is interested in.

SOFTWARE/PLATFORM

Visual Studio Code

GitHub

Equals Operator Overloading

```
27  void LargeInt:: operator= (const LargeInt& other)
28  {
29      UDList<int>::iterator head = list.begin();
30      if (head){
31          list.makeEmpty();
32      }
33      list = other.list;
34      this->negative = other.negative;
35  }
```

Demonstration

```
43  int main(int argc, const char * argv[]) {
44
45      // 50 digits of Pi number
46      LargeInt l1("31415926535897932384626433832795028841971693993751O");
47      LargeInt l2(0);
48
49      cout <<  "LargeInt l1 = " << l1 << endl;
50      cout <<  "LargeInt l2 = " << l2 << endl;
51
52      l2 = l1; // l2 is now has the same integers as l1
53
54      cout <<  "LargeInt l1 = " << l1 << endl;
55      cout <<  "LargeInt l2 = " << l2 << endl;
56
57      return 0;
58  }
59
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
~/Downloads/LargeInt-master » ./a.out
LargeInt l1 = 31415926535897932384626433832795028841971693993751O
LargeInt l2 = 0
LargeInt l1 = 31415926535897932384626433832795028841971693993751O
LargeInt l2 = 31415926535897932384626433832795028841971693993751O
```

Multiplication Operator Overloading

```
252  LargeInt LargeInt:: operator* (const LargeInt& other)
253  {
254      LargeInt temp( abs(*this) );
255      LargeInt temp2 = abs(other); // this will be a counter to keep track of how many times to add
256      LargeInt subsObject(1);
257      UDList<int>::iterator ithis = list.begin();
258      UDList<int>::iterator itemp2 = temp2.list.begin();
259
260      if (ithis && itemp2) {
261
262          if ( (ithis->info != 0) && (itemp2->info != 0) ) {
263              while ( !(temp2 == subsObject) ) {
264                  temp = add(temp, (*this)); // add function does not care about sign
265                  temp2 = (temp2 - subsObject);
266              }
267          }
268
269          if ( !bothNegative(other) && !bothPositive(other) )
270              temp.negative = true;
271      }
272
273      return temp;
274  }
```

Demonstration

```
43  int main(int argc, const char * argv[]) {
44
45      // First 50 digits of Pi number
46      LargeInt l1
        ("31415926535897932384626433832795028841971693993751058209
        74944592307816406286208998628034825342117067982148086 51");
47      // First 110 digits of Pi number
48      LargeInt l2
        ("31415926535897932384626433832795028841971693993751058209
        74944592307816406286208998628034825342117067982148086513328
        23066470938446095505822317253594081284811174502841027019 38
        521105559");
49      LargeInt result;
50      result = l1 + l2;
51
52      cout <<  "LargeInt l1 = " << l1 << endl;
53      cout <<  "LargeInt l2 = " << l2 << endl;
54      cout << endl;
55      cout <<  "LargeInt resut (l1 + l2) = " << result << endl;
56
57      return 0;
```

PROBLEMS TERMINAL ••• 1: zsh

Cmd + click to follow link

```
~/Downloads/LargeInt-master » ./a.out
LargeInt l1 = 31415926535897932384626433832795028841971693993751058209749445923
07816406286208998628034825342117067982148086 51
LargeInt l2 = 31415926535897932384626433832795028841971693993751058209749445923
07816406286208998628034825342117067982148086513282306647093844609550582231725 35
94081284811174502841027019385211055 59

LargeInt resut (l1 + l2) = 31415926535897932384626433832795028841971693993751058
20974944592307816720445474357607358671606455395932436506230222244157675942104 0
09813013365988029028343920932818314408736735914210
```

Future work:

- Add new functionalities:
  - Except Handling
  - Iteration Classes

Thank you.

Any questions ?