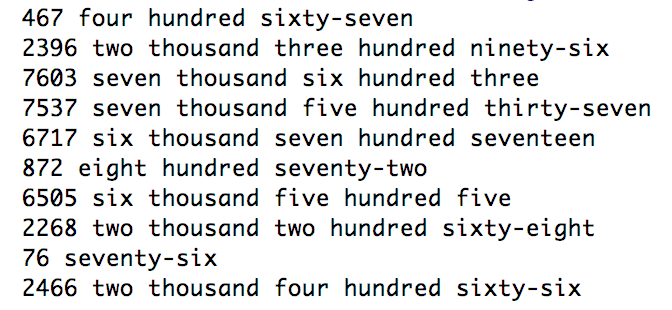
CSC 122 001 Computer Science II

Julius Ranoa

Chapter 11 Programming Challenge 1. Check-Writing.

Write a class that accepts integers (range 0 – 9999, inclusive) and converts that into English descriptions of those integers.

Screenshot of runtime:



Files Included: (1) main.cpp, (2) Numbers.h, (3) Numbers.cpp

**main.cpp**

#include **<iostream>**#include **<random>**#include **"Numbers.h"  
  
int** main() {  
   
 Numbers n(0);  
  
 srand(time(**NULL**));  
 **for** (**int** i = 0; i < 10; i++) {  
 **static int** x;  
 x = rand() % 9999;  
 n.setNumber( x );  
 std::cout << x << **" "**;  
 n.print();  
 std::cout << **"\n"**;  
 }  
  
 **return** 0;  
}

**Numbers.h**

#ifndef **CH11\_PR1\_CHECK\_WRITING\_NUMBERS\_H**#define **CH11\_PR1\_CHECK\_WRITING\_NUMBERS\_H**#include **<string>  
  
class** Numbers {  
  
 *// Min and Max Definition.* **static const int** min;  
 **static const int** max;  
  
 *// Int to Text Conversions. Values in Implementation.* **static** std::string lessThan20[ ];  
 **static** std::string tens[ ];  
 **static** std::string hundred;  
 **static** std::string thousand;  
  
**private**:  
 **int** number;  
 std::string text;  
 std::string stringify(**int**);  
  
**public**:  
 Numbers();  
 Numbers(**int**);  
 **bool** isInRange(**const int**&);  
  
 **void** print();  
 std::string getText();  
 **void** setNumber(**const int**);  
  
};  
  
  
#endif *//CH11\_PR1\_CHECK\_WRITING\_NUMBERS\_H*

**Numbers.cpp**

#include **<iostream>**#include **"Numbers.h"***// Number to Text Conversions - Reference***const int** Numbers::min = 0;  
**const int** Numbers::max = 9999;  
std::string Numbers::lessThan20[ ] = {  
 **"zero"**,  
 **"one"**, **"two"**, **"three"**, **"four"**, **"five"**,  
 **"six"**, **"seven"**, **"eight"**, **"nine"**, **"ten"**,  
 **"eleven"**, **"twelve"**, **"thirteen"**, **"fourteen"**, **"fifteen"**,  
 **"sixteen"**, **"seventeen"**, **"eighteen"**, **"nineteen"**};  
std::string Numbers::tens[ ] = {  
 **""**,  
 **"ten"**, **"twenty"**, **"thirty"**, **"forty"**, **"fifty"**,  
 **"sixty"**, **"seventy"**, **"eighty"**, **"ninety"**};  
std::string Numbers::hundred = **"hundred"**;  
std::string Numbers::thousand = **"thousand"**;  
  
*// Default Constructor*Numbers::Numbers() {  
 number = 0;  
 text = stringify(number);  
}  
  
*// Constructor. Accepts int as argument.*Numbers::Numbers(**int** val) {  
 **if** (!Numbers::isInRange(val)) {  
 std::cout << **"Number not in range. Exiting program."**;  
 exit(-1);  
 }  
 number = val;  
 text = stringify(val);  
}  
  
*// Converts an int value as argument.  
// Private function. Assumes value is in range.*std::string Numbers::stringify(**int** val) {  
 **if** (val == 0) {  
 **return** lessThan20[0];  
 }  
  
 std::string temp\_str;  
 **int** temp\_int;  
  
 *// Tens* temp\_int = val % 100;  
 **if** (temp\_int < 20 && temp\_int != 0) {  
 temp\_str = lessThan20[temp\_int];  
 } **else** {  
 temp\_str = tens[ temp\_int / 10 ];  
 temp\_int = temp\_int % 10;  
 **if** (temp\_int != 0) {  
 temp\_str += **"-"** + lessThan20[ temp\_int % 10 ];  
 }  
 }  
  
 *// Hundred* temp\_int = val % 1000;  
 temp\_int = temp\_int / 100; *// Get the hundreds digit.* **if** ( temp\_int != 0 ) {  
 temp\_str = lessThan20[ temp\_int ]  
 + **" "** + hundred + **" "** + temp\_str;  
 }  
  
 *// Thousand* temp\_int = val / 1000;  
 **if** (temp\_int != 0) {  
 temp\_str = lessThan20[temp\_int] + **" "** + thousand + **" "** + temp\_str;  
 }  
  
 **return** temp\_str;  
}  
  
*// Tests if value is in accepted range.***bool** Numbers::isInRange(**const int**& val) {  
 **if** (val < min) { **return false**; }  
 **else if** (val > max ) { **return false**; }  
 **else return true**;  
}  
  
*// UX Functions***void** Numbers::print() {  
 std::cout << text;  
}  
  
**void** Numbers::setNumber(**const int** val) {  
 **if** (!Numbers::isInRange(val)) {  
 std::cout << **"Number not in range. Exiting program."**;  
 exit(-1);  
 }  
 number = val;  
 text = stringify(val);  
}