Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

по курсу «Объектно-ориентированное программирование»

на тему «Изучение механизм исключений в языке C++»

Выполнили студенты группы 19ВВ3:

Ланцов Андрей

Трубаненко Александр

Приняли:

Евсеева Ю.И.

**2021**

**Цель работы:** Изучить механизм исключений в языке C++.

**Задания:**

1. Доработать шаблонный класс из предыдущей лабораторной работы таким образом, чтобы в программе осуществлялась генерация и обработка исключительных ситуаций.

Исключительные ситуации генерируются:

1. В конструкторе с параметром при попытке создать контейнер больше максимального

размера;

1. В операции []

При попытке обратиться к элементу с номером меньше 0 или больше текущего размера контейнера ;

1. В операции +

При попытке добавить элемент с номером больше максимального размера;

1. В операции –

При попытке удалить элемент из пустого контейнера.

1. Создать программу, использующую механизм обработки исключительных ситуаций, позволяющую выполнять перевод вещественного числа из одной системы счисления в другую.

4) Перевод числа из восьмеричной системы счисления в шестеричную.

**Ход работы:**

Задание 1

Листинг:

template <class T>  
class Set{  
 std::vector<T> set;  
 size\_t level;  
public:  
 Set(){  
 level = 0;  
 }  
 Set(Set<T>& left){  
 \*this = left;  
 }  
  
 Set(size\_t arr\_size){  
 if (arr\_size == set.max\_size()){  
 throw std::bad\_alloc();  
 }  
 else{  
 set.resize(arr\_size);  
 level = arr\_size;  
 }  
 }  
  
 Set(T\* \_set,size\_t arr\_size){  
 this->level = 0;  
 for (int i = 0; i < arr\_size; ++i) {  
 if (!has(\_set[i])){  
 \*this += \_set[i];  
 }  
 }  
 }  
 ~Set(){  
 set.clear();  
 }  
  
public:  
 bool has(T& item){  
 int current\_level = set.size();  
 for (int j = 0; j < current\_level; ++j) {*//check value is already in set* if(set[j]==item){  
 return 1;  
 }  
 }  
 return 0;  
 }  
  
 void resize(size\_t new\_size){  
 set.resize(new\_size);  
 }  
  
 T& operator[](size\_t index){  
 if (index > this->level){  
 throw std::out\_of\_range("Access violation");  
 }  
 else{  
 return set[index];  
 }  
 }  
  
 void operator+=(T& item){  
 if (this->level+1 > set.max\_size()){  
 throw std::bad\_alloc();  
 }  
 else{  
 if (!has(item)){  
 set.push\_back(item);  
 level++;  
 }  
 }  
 }  
  
  
 Set<T>\* operator+(Set<T>& left){  
 Set<T>\* new\_set = new Set<T>;  
 \*new\_set = Set(\*this);  
  
 if ((this->level+left.level) > new\_set->set.max\_size()){  
 throw std::bad\_alloc();  
 }  
 else{  
 new\_set->resize(this->level+left.level);  
 new\_set->level = this->level;*//copy right one into new set* int left\_level = left.level;  
  
 for (int i = 0; i < left\_level; ++i) {*//unite with left one* if (!this->has(left.set[i])){  
 \*new\_set += left.set[i];  
 }  
 }  
 return new\_set;  
 }  
 }  
 Set<T>\* operator\*(Set<T>& left){  
 Set<T>\* new\_set = new Set<T>;  
 int left\_level = left.level;  
  
 for (int i = 0; i < left\_level; ++i) {  
 if (this->has(left.set[i])){  
 \*new\_set += left.set[i];  
 }  
 }  
 return new\_set;  
 }  
  
  
};

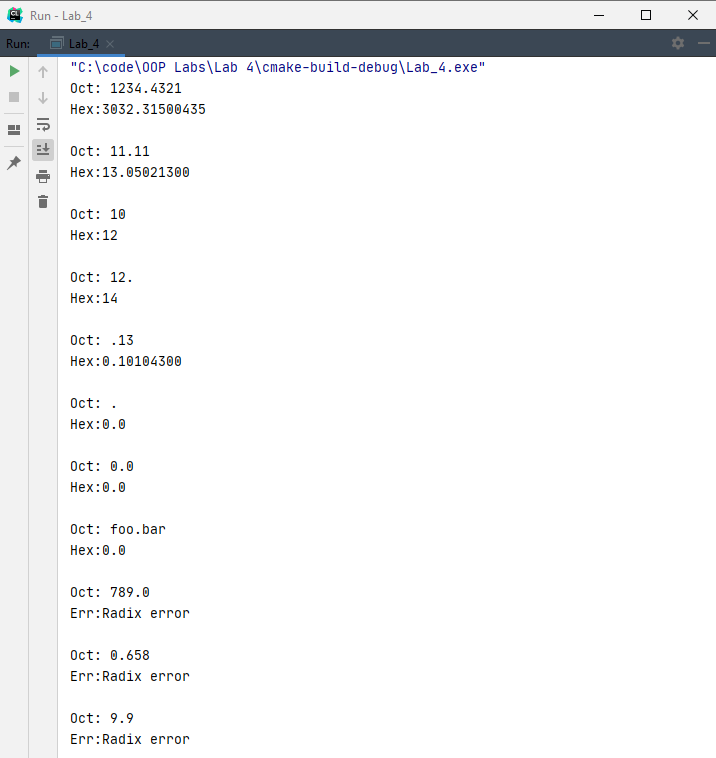
void test\_set(){  
 int arr1[] = {1,2,3};  
 int arr2[] = {1,3,3};  
 int arr3[] = {2,2,4,5};  
  
 std::string strs1[] = {"foo","bar","baz"};  
 std::string strs2[] = {"foo","foo","baz"};  
 std::string strs3[] = {"foobar","bar","baz","foo","foo"};  
  
 Set<int> set\_int1(arr1,3);  
 Set<int> set\_int2(arr2,3);  
 Set<int> set\_int3(arr3,4);  
  
 try{  
 set\_int3[5];  
 } catch (std::out\_of\_range& err) {  
 std::cout << err.what();  
 }  
  
 Set<int> big\_set(185185183);  
 try{  
 int i = 0;  
 while (true){  
 big\_set += i;  
 i++;  
 }  
 } catch (std::bad\_alloc& err) {  
 std::cout << err.what();  
 }  
  
 Set<std::string> set\_str1(strs1,3);  
 Set<std::string> set\_str2(strs2,3);  
 Set<std::string> set\_str3(strs3,5);  
  
 Set<int>\* new\_int\_set1 = set\_int2+set\_int3;  
 Set<int>\* new\_int\_set2 = set\_int1\*set\_int3;  
 Set<int>\* new\_int\_set3 = set\_int2\*set\_int3;  
  
 try{  
 Set<std::string>\* new\_str\_set1 = set\_str1+set\_str2;  
 Set<std::string>\* new\_str\_set2 = set\_str2\*set\_str3;  
 } catch (std::bad\_alloc& err) {  
 std::cout << err.what();  
 }  
}

Задание 2

Листинг:  
*//constants*#define **SIZE** (sizeof(int)\*2)  
  
static char\* integral\_conversion(int valint, int integral\_precision,int src\_radix,int dst\_radix){  
 int dec = 0;  
 int src\_integer = valint;  
 int mod;  
 int i1 = 0;*//degree* int dst\_integral = dec;  
 char digits[17] = "0123456789ABCDEF";  
  
 *//convert src\_integer part from src\_radix into 10* while (src\_integer != 0) {  
 mod = src\_integer % 10;  
 src\_integer /= 10;  
  
 if (mod >= src\_radix){  
 throw std::exception("Radix error");  
 }  
 dec = dec + mod \* pow(src\_radix, i1++);  
 }  
  
 *//convert integral part from 10 into dst\_radix* char\* integral\_part = new char[integral\_precision+1];  
 mod = 0;  
 for (int i = 0; i < integral\_precision; i++) {  
 mod = dec % dst\_radix;  
 dec /= dst\_radix;  
 integral\_part[integral\_precision - i - 1] = digits[mod];  
 }  
 integral\_part[integral\_precision] = '\0';  
 return integral\_part;  
}  
  
static char\* float\_conversion(int valfloat, int src\_radix,int dst\_radix){  
 char digits[17] = "0123456789ABCDEF";  
  
 *//convert float part from src\_radix into 10* double frc = 0, fldigit;  
 int digit = 0;  
  
 int i=1;  
 while(valfloat > 0){  
 digit = valfloat % 10;  
 valfloat /=10;*//get integral part* if (digit >= src\_radix){  
 throw std::exception("Radix error");  
 }  
  
 frc += (digit / (pow(src\_radix, i)));  
 i++;  
 }  
  
 *//convert float part from 10 int dst\_radix* char\* float\_part = new char[**SIZE** + 1];  
 int digit2;  
  
 for (int i = 0; i < **SIZE**; i++) {  
 frc \*= dst\_radix;  
 digit2 = frc;  
 frc -= digit2;  
 float\_part[i] = digits[digit2];  
 }  
 float\_part[**SIZE**] = '\0';  
 return float\_part;  
}  
  
static void reverse\_str(std::string& str){  
 int n = str.length();  
 for (int i = 0; i < n / 2; i++)  
 std::swap(str[i], str[n - i - 1]);  
}  
  
void radix\_convert(std::string& src\_str,std::string& buffer,int src\_radix, int dst\_radix) {  
 if (!((src\_radix > 0 && src\_radix < 16) || (dst\_radix > 0 && dst\_radix < 16))){  
 throw std::exception("Radix must be equal to 0...16");  
 }  
  
 double val = strtod(src\_str.c\_str(),**NULL**);  
 if (val == 0){  
 buffer = "0.0";  
 return;  
 }  
 else if(val == **HUGE\_VAL** || val==-**HUGE\_VAL**){  
 throw std::exception("HUGE VALUE");  
 }  
  
 int dot\_position = src\_str.find('.');  
 int valint = val;  
 std::string float\_part\_reversed = src\_str.substr(dot\_position+1,src\_str.length());  
 reverse\_str(float\_part\_reversed);  
  
 int valfloat = atoi(float\_part\_reversed.c\_str());  
  
 if (dot\_position >= -1){*//integer value* int integral\_precision = src\_str.length()+1;  
 char\* integral\_part;  
 char\* float\_part;  
  
 if(dot\_position == -1 || dot\_position == src\_str.length() - 1){*//xx format == xx.0 == X.* try{  
 integral\_part = integral\_conversion(valint,integral\_precision,src\_radix,dst\_radix);  
 } catch (std::exception& err) {  
 throw err;  
 }  
 *//to clear nills at the top* int int\_result\_start;  
 for (int i = 0; i < integral\_precision; i++) {  
 if (integral\_part[i] != '0') {  
 int\_result\_start = i;  
 break;  
 }  
 }  
 buffer = integral\_part+int\_result\_start;  
 delete[] integral\_part;  
 return;  
 }  
 else if(dot\_position == 0){*//.0 format == 0.xxx* try{  
 float\_part = float\_conversion(valfloat,src\_radix,dst\_radix);  
 } catch (std::exception& err) {  
 throw err;  
 }  
  
 buffer = "0.";  
 buffer.append(float\_part);  
 delete[] float\_part;  
 return;  
 }  
 else{*//x.x format* try{  
 integral\_part = integral\_conversion(valint,integral\_precision,src\_radix,dst\_radix);  
 } catch (std::exception& err) {  
 throw err;  
 }  
 int int\_result\_start=0;  
 for (int i = 0; i < integral\_precision; i++) {  
 if (integral\_part[i] != '0') {  
 int\_result\_start = i;  
 break;  
 }  
 }  
  
 try{  
 float\_part = float\_conversion(valfloat,src\_radix,dst\_radix);  
 } catch (std::exception& err) {  
 throw err;  
 }  
  
 buffer = integral\_part+int\_result\_start;  
 buffer.append(".");  
 buffer.append( float\_part);  
  
 delete[] integral\_part;  
 delete[] float\_part;  
 }  
 }  
}

void test\_radix(){  
 std::string test\_values[] = {"1234.4321","11.11","10","12.",".13",".","0.0","foo.bar","789.0","0.658","9.9"};std::string result;  
  
 int testes\_count = sizeof(test\_values)/sizeof(std::string);  
  
 for (int i = 0; i < testes\_count; ++i) {  
 std::cout << "Oct: "<< test\_values[i] << '\n';  
 result.clear();  
 try {  
 radix\_convert(test\_values[i],result, 8, 6);  
 std::cout << "Hex:" << result << "\n\n";  
 }catch (std::exception& err) {  
 std::cout << "Err:" << err.what() << "\n\n";  
 }  
 }  
}

Результат работы программы:



**Вывод:** В ходе выполнения лабораторной работы были изучены Изучить механизмы исключений в языке C++.