|  |
| --- |
| 23631-2 |
| Точные вычисления комбинаторных чисел |
| Отчет по лабораторной работе |

|  |
| --- |
| Решетняк Иван Витальевич  2019 |

# Постановка задачи:

Требуется реализовать программный пакет для точного подсчета комбинаторных чисел. Необходимые числа: число размещений, число размещений с повторениями, число перестановок, число сочетаний, число Стирлинга второго рода, число Белла.

Также для расчетов нужно создать класс беззнакового 32-битного целого числа, контролирующего переполнения.

Необходимо реализовать простой пользовательский интерфейс в виде командной строки.

# Решение задачи:

## number

*number –* вспомогательный класс, представляющий собой беззнаковое целое число с перегруженными арифметическими операторами +-\*. и методом возведения в степень. Внутри этих методов контролируется переполнение. При обнаружении переполнения генерируется исключение.

class number {

private:

using uint = unsigned int;

uint Value;

friend std::istream & operator>>( std::istream &, number & );

friend std::ostream & operator<<( std::ostream &, const number & );

public:

number() = default;

number( uint Val ) : Value( Val ) {}

explicit operator uint( void ) const;

number & operator=( const number &RV );

number operator+( const number &RV ) const;

number & operator+=( const number &RV );

number operator-( const number &RV ) const;

number & operator-=( const number &RV );

number operator\*( const number &RV ) const;

number & operator\*=( const number &RV );

number operator/( const number &RV ) const;

number & operator/=( const number &RV );

number operator++( int );

number & operator++( void );

number operator--( int );

number & operator--( void );

bool operator==( const number &RV ) const;

bool operator>( const number &RV ) const;

bool operator>=( const number &RV ) const;

bool operator<( const number &RV ) const;

bool operator<=( const number &RV ) const;

number pow( number Pow ) const;

};

Далее приводятся описания контроля переполнения в методах класса.

### Сложение A + B

Вычисляется C = A + B. Если C < A или C < B генерируется исключение.

### Вычитание A - B

Вычисляется C = A - B. Если C > A генерируется исключение.

### Умножение A \* B

Вычисляется C = A \* B. Если A != 0 и C / A != B (не проходится проверка делением) генерируется исключение.

### Деление A / B

Если B == 0 генерируется исключение.

### Возведение в степень A ^ P

Возведение в степень выполняется последовательным умножением числа A самого на себя P раз. Если при этих умножениях генерируется исключение, оно пробрасывается дальше.

## Комбинаторные числа

### Число размещений (без повторений)

Если k > n, возвращается 0, иначе число размещений находится перемножением чисел от (n – k + 1) до n включительно:

number Res = 1;

for (number i = m - n + 1; i <= m; ++i

Res \*= i;

return Res;

### Число размещений (с повторениями)

Число размещений находится возведением числа n в k-ю степень:

### return n.pow(k);

### Число перестановок

Число перестановок находится последовательным перемножением первых n натуральных чисел:

number Res = 1;

for (number i = 1; i <= n; i++)

Res \*= i;

return Res;

### Число сочетаний

Если n < k или n == 0 возвращается 0, иначе число сочетаний находится построением треугольника Паскаля (в любой момент хранятся 2 его строки):

std::vector<number> Current((unsigned)(n + 1), 1), Prev((unsigned)(n + 1), 1);

for (number y = 2; y <= n; y++) {

for (number x = 1; x <= y - 1; x++)

Current[(unsigned)x] = Prev[(unsigned)(x - 1)] + Prev[(unsigned)(x)];

Current.swap(Prev);

}

return Prev[(unsigned)k];

### Число Стирлинга второго рода

Если k == n возвращается 1, если k ==0 или n == 0 или k > n возвращается 0, иначе число Стирлинга второго рода находится по рекуррентной формуле построением пирамиды чисел Стирлинга (в любой момент хранятся 2 ее строки):

std::vector<number> Current((unsigned)(n + 1), 0), Prev((unsigned)(n + 1), 0);

Prev[1] = 1;

for (number y = 2; y <= n; y++) {

for (number x = 1; x <= y; x++)

Current[(unsigned)x] = x \* Prev[(unsigned)(x)] + Prev[(unsigned)(x - 1)];

Current.swap(Prev);

}

return Prev[(unsigned)k];

### Число Белла

Если n <= 1 возвращается 1, иначе число Белла вычисляется как сумма чисел Стирлинга в n-ой строке пирамиды – строится пирамида Стирлинга как описано выше и суммируются числа в последней полученной строке:

std::vector<number> Current((unsigned)(n + 1), 0), Prev((unsigned)(n + 1), 0);

Prev[1] = 1;

for (number y = 2; y <= n; y++) {

for (number x = 1; x <= y; x++)

Current[(unsigned)x] = x \* Prev[(unsigned)(x)] + Prev[(unsigned)(x - 1)];

Current.swap(Prev);

}

number Sum = 0;

for (const auto &it : Prev)

Sum += it;

return Sum;

# Работа с пакетом

* Максимальное значение числа размещений без повторений, которое удалось получить – A(13, 11) = 3113510400
* Максимальное значение числа размещений с повторениями, которое удалось получить – U(10, 9) = 1000000000
* Максимальное значение числа перестановок, которое удалось получить –   
  P(12) = 479001600
* Максимальное значение числа сочетаний, которое удалось получить –  
  C(34, 17) = 233606220
* Максимальное значение числа Стирлинга второго рода, которое удалось получить –  
  S(16, 8) = 2141764053
* Максимальное значение числа Белла, которое удалось получить –   
  B(15) = 1382958545

Все опробованные вычисления происходили за очень короткое время (< 1 секунды).

