Объектно-ориентированное программирование

Отчёт

к экзаменационному проекту № 4

Мальцевой Софьи Алексеевны,
студентки 3 курса группы АБ335
направления подготовки
09.03.02 «Информационные системы и технологии»

Сыктывкар 2023

Класс Permutation

Заголовочный файл класса Permutation.h

В классе есть два поля: целочисленная переменная N и указатель на первый элемент массива целых чисел * p. Оба поля приватные и доступные только внутри класса. Прописываем их в заголовке класса.

В классе есть публичные конструктор, деструктор и метод Step(), который позволяет пользователю запустить перестановку.

В конструктор передаем значение N и создаем динамический массив p размера N, с помощью метода SetValues() помещаем значения от 1 до N в порядке возрастания, а после передаем в поток исходный массив с помощью метода Print().

В деструкторе освобождаем занятую память с помощью delete [] p.

Метод Step() в заголовке класса подробно описан не будет.

Остальные методы описаны под ключевым словом «private» (слово опущено, в классах по умолчанию «private»). К ним относятся методы SetValues(), SetNullValues(), Create() и Print(). Все методы не возвращают значений, поэтому тип методов — void.

Методы SetValues() и SetNullValues() похожи, только первым мы пользуемся при создании объекта, чтобы заполнить массив p начальными значениями, а второй используется тогда, когда все перестановки были совершены, и новых без повторения старых уже быть не может, поэтому массив p внутри цикла for заполняется нулями. Кроме того, метод SetNullValues() совершает вызов метода Print(), чтобы показать пользователю новую перестановку.

Методы Create() — по сути, главный в генерации перестановки — и Print() просто проинициализированы в заголовочном файле и будут расписаны в файле Permutation.cpp.

Файл Permutation.cpp

Подключаем директивой #include заголовочный файл класса Permutation.h и стандартную библиотеку <iostream> для использования стандартного потока вывода cout в методе Print().

В файле описываем методы Create(), Print()и Step(). Перед каждым методом раскрываем область видимости Permutation::.

Метод Create() совершает перестановку элементов в массиве. Его принцип работы заключается в следующем: проходимся в цикле for по массиву с его конца до первого элемента, находим индекс элемента arrj, который меньше элемента с индексом arrj + 1. Если arrj == -1, то все элементы расположены в порядке убывания, и новых перестановок больше нет, поэтому вызывается функция SetNullValues(), которая заполняет массив нулями и печатает его.

Дальше идём в цикле `while` по массиву с его последнего элемента до первого, чтобы найти наименьший индекс элемента, который больше элемента с индексом `arrj`. После этого меняем элементы местами.

Оставшиеся элементы (после того индекса `arrj`, который был найден сначала) выстраиваем в порядке возрастания. Для этого берем элементы на границе (`arrj + 1` и `N - 1`) и меняем их местами, сужаем границы на единицу с обеих сторон и меняем элементы местами до тех пор, пока левая граница меньше правой. В конце вызываем метод `Print()`.

Для наглядности операций этого метода приведем пример: если пользователь ввёл `N`, равное `7`, и в результате нескольких перестановок получил такую последовательность: `{1,2,7,6,5,4,3}`. И для новой перестановки `arrj = 1`(`p[arrj] = 2`). Тогда `arrh = 6` (`p[arrh] = 3`). Меняем эти два элемента и получаем временный результат `{1,3,7,6,5,4,2}`. Затем меняем местами элементы с индексом `2` и `6` — получаем `{1,3,2,6,5,4,7}`. Сужаем границы до индексов `3` и `5` — `{1,3,2,4,5,6,7}`. Если сузить границы ещё на единицу, то получим индексы `4` и `4`, что не соотвествует условию цикла `while` (левая граница меньше правой). В результате все элементы после элемента с индексом `1` выстроились по возрастанию.

Метод `Print()` использует поток вывода cout, перед которым раскрываем область видимости `std::`, так как в начале программы не использовалось выражение `using namespace std`. Выводим с помощью цикла `for` текущее расположение элементов в массиве `p` в поток. Оформляем в виде кортежей (`{}`).

Последний метод — `Step()` — доступен пользователю. Внутри метода есть условие: если первый элемент массива равен нулю (как и все остальные, исходя из условия, что минимальное значение в массиве может быть `1`), то все перестановки уже были выполнены, и массив был заполнен нулями, поэтому нет смысла заново вызывать метод `Create()`, чтобы при проверке условия вызвалась функция `SetNullValues()`, которая бы совершила проход по циклу и в итоге вызвала метод `Print()`, поэтому сразу внутри `Step()` при нулевом первом элементе происходит вызов функции `Print()`. Если размер массива равен `1`, то новых перестановок из единицы быть не может, и сразу идёт вызов метода `SetNullValues()`. И если не выполнилось ни одно из предыдущих условий, вызывается метод `Create()`.

Из всех методов, которые есть в классе, константный только один — `Print()`. Сделать метод `Step()` константным нельзя, так как в нём происходит вызов неконстантных методов `SetNullValues()` и `Create()`. Все остальные методы неконстантные, потому что меняют поля объекта класса.

Файл `main.cpp`

В начале файла с помощью директивы `#include` подключен заголовок `Permutation.h` и стандартная библиотека `<iostream>` для потокового ввода `cin` и вывода `cout`.

Содержал основную функцию `int main()`. Были сомнения в том, как конкретно организовать работу пользователя с программой: если пользователь хочет делать сколько ему угодно перестановок, то после каждой из них стоит спрашивать, продолжать или нет эти перестановки (даже если значения массива давно равны нулям). Однако это хорошо работает на массиве размером не больше 4, когда количество перестановок равно `4! = 24`. Отвечать на вопросы командной строки более чем сто раз подряд может быть затруднительно. Поэтому этот вариант был предусмотрен и спрятан в комментарии как вариант, который имеет право на существование на маленьких массивах.

В итоге в файле есть две функции — `int factorial()` и `int main()`.

Функция `factorial()` — рекурсивная функция, принимает на вход целое число и возвращает его, умноженный на факториал этого же числа, уменьшенного на единицу. Вызов продолжается до тех пор, пока на вход не поступит единица.

В главном методе `main()` происходит ввод числа `N`, создание объекта класса `Permutation`, вычисление факториала от `N`, вызов метода `Step()` внутри цикла `for` от `1` до `N!`. При этом на последней итерации цикла будет выведен массив нулей, потому что количество перестановок всего `N! - 1` без самого первого распределения элементов в массиве по возрастанию.

Кроме того, предусмотрена обработка исключения внутри функции `main()`. Если пользователь введёт `N`, равное `0` (то есть массив нулевого размера), будет выброшено исключение (`throw 0`), потому что содержимое `main()` находится в блоке `try`, сразу за которым идёт блок `catch`, который вылавливает исключение типа `int`.