Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Реферат

По дисциплине «Основы алгоритмизации и программирования»

На тему «Отличия использования управляющего символа ‘\n’ и манипулятора endl. Связанность потоков»

Выполнила:

Студент(ка) 1 курса 7 группы

Подшиваленко Диана Игоревна

Проверил:

Белодед Николай Иванович

2023, Минск

**Содержание**

1. Предисловие … 3
2. Использование управляющего символа ‘\n’ и манипулятора endl … 4
3. Первое отличие … 5
4. Второе отличие … 7
5. Связанность потоков cin и cout … 8
6. Выводы … 10

**Предисловие**

В данном реферате будут рассмотрены основные отличия использования управляющего символа ‘\n’ и манипулятора endl. Также будет затронута тема связанности потоков cin и cout.

**1. Использование управляющего символа ‘\n’ и манипулятора endl**

Управляющие символы (или как их ещё называют — escape-последовательность) — символы, которые выталкиваются в поток вывода, с целью форматирования вывода или печати некоторых управляющих знаков С++. Все управляющие символы, при использовании, обрамляются двойными кавычками, если необходимо вывести какое-то сообщение, то управляющие символы можно записывать сразу в сообщении, в любом его месте. Одним из наиболее известных управляющих символов является ‘\n’, который осуществляет переход на новую строку. Рассмотрим пример с его использованием:

#include <iostream>

#include <Windows.h>

#include <iomanip>

#include <ctime>

using namespace std;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "Hello!\n";

cout << "I am a computer programmer";

}

Результат выполнения:

Как видим, данный управляющий символ прекрасно выполняет своё предназначение. После печати строки “Hello!” был осуществлён переход на новую строку.

Система ввода/вывода С++ включает второй способ изменения параметров форматирования потока. Для этого используются специальные функции, называемые манипуляторами (manipulators), которые могут включаться в выражения ввода/вывода. Одним из наиболее часто используемых манипуляторов является манипулятор endl, который, как и управляющий символ ‘\n’, предназначен для перехода на новую строку. Рассмотрим пример с его использованием:

#include <iostream>

#include <Windows.h>

#include <iomanip>

#include <ctime>

using namespace std;

int main() {

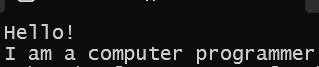
SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "Hello!" << endl;

cout << "I am a computer programmer";

}

Результат выполнения:

Здесь также видим, что манипулятор endl успешно выполняет переход на новую строку.

Так зачем же в языке С++ существует и ‘\n’, и endl, если они оба выполняют одну и ту же функцию? Давайте разбираться.

**2. Первое отличие**

Первое отличие заключается в скорости выполнения программы. Давайте рассмотрим это на простеньком примере, в котором будем в цикле выводить много раз одну и ту же строку, каждый раз делая переход на новую. При этом замерим время выполнения программы.

Управляющий символ ‘\n’:

#include <iostream>

#include <Windows.h>

#include <iomanip>

#include <ctime>

using namespace std;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n;

cin >> n;

clock\_t start = clock();

for (int i = 0; i < n; i++) {

cout << "I am a computer programmer" << '\n';

}

clock\_t end = clock();

double duration = (float)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.5f", duration);

}

Результаты работы:

для 1000 раз:

для 10000 раз:

для 100000 раз:

для 500000 раз:

для 1000000 раз:

Манипулятор endl:

#include <iostream>

#include <Windows.h>

#include <iomanip>

#include <ctime>

using namespace std;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n;

cin >> n;

clock\_t start = clock();

for (int i = 0; i < n; i++) {

cout << "I am a computer programmer" << endl;

}

clock\_t end = clock();

double duration = (float)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.5f", duration);

}

Результаты работы:

для 1000 раз:

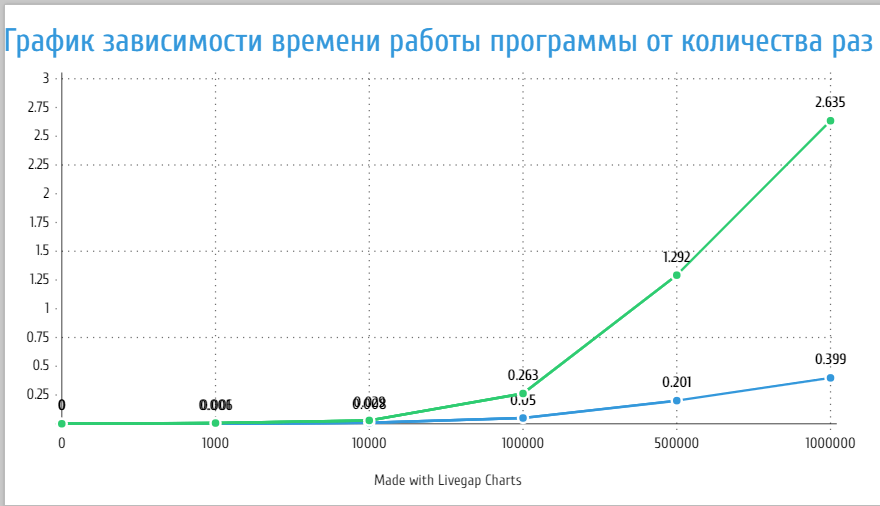
для 10000 раз:

для 100000 раз:

для 500000 раз:

для 1000000 раз:

Использую результаты наших замеров, построим график зависимости времени работы от количества раз вывода строки.



Как видим, манипулятор endl работает намного медленнее, чем управляющий символ ‘\n’.

Почему же так происходит?

**3. Второе отличие**

Давайте запустим нашу программу в режиме отладки и посмотрим за происходящим. Для большей наглядности изменим поток cout на ofstream (так как будем работать с файлом) и будем наблюдать за изменением в файле.

#include <iostream>

#include <Windows.h>

#include <iomanip>

#include <ctime>

#include <fstream>

using namespace std;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

ofstream out("t.txt", 'w');

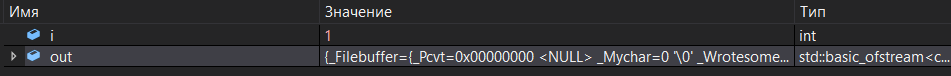
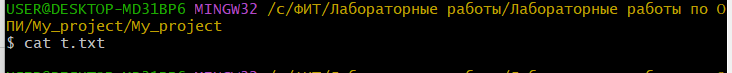
for (int i = 0; i < 100; i++) {

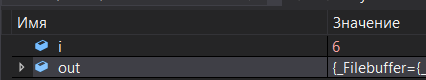
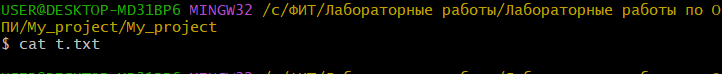
out << "I am a computer programmer. " << "I am learn C++"

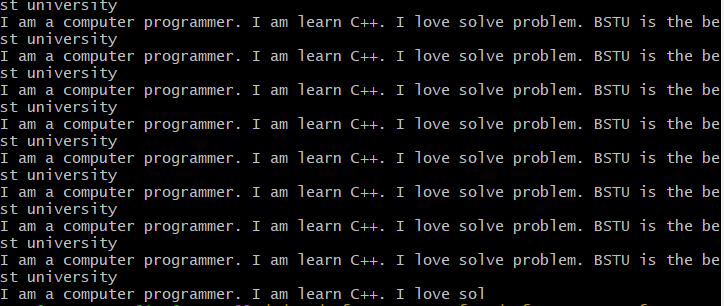
<<'\n';

}

}

Выполним первую итерацию и откроем файл для проверки содержимого.

Файл пуст. Странно да? Давайте выполним ещё 5 итераций.

И снова пусто. Давайте выполним 50 итераций.

Ура! Как видим,в файле что-то появилось. Давайте проверим сколько там вывелось строк.

Хотя было сделано 50 итераций, в файл вывелось только 43 строки. Давайте выполним нашу программу до конца и посмотрим на результат.

Как видим, после завершения программы все наши строки были выведены в файл.

Давайте теперь попробуем сделать только 1 изменение. Заменим ‘\n’ на endl.

#include <iostream>

#include <Windows.h>

#include <iomanip>

#include <ctime>

#include <fstream>

using namespace std;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

ofstream out("t.txt", 'w');

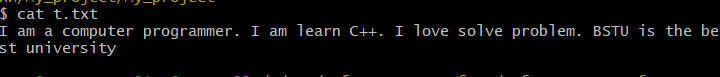
for (int i = 0; i < 100; i++) {

out << "I am a computer programmer. " << "I am learn C++. " << "I love solve problem. " << "BSTU is the best university"

<< endl;

}

}

Выполним первую итерацию и проверим содержимое файла.

В данном случае вывелось сразу. Давайте сделаем 5 итераций и посмотрим на результат количества строк.

Наши строки сразу выводятся в файл.

Вот и второе отличие ‘\n’ от endl. Потоки (cout, cerr, ofstream) имеют свой буфер. Когда мы используем ‘\n’ для перевода строки, то этот символ записывается в буфер. И потом, когда место в буфере закончится и поток сбросит всё в файл (или консоль), то ‘\n’ укажет на перевод строки. А вот манипулятор endl выполняет сразу сброс накопившейся в буфере информации в файл (или консоль). Он использует метод flush(), который принудительно это делает.

Это является причиной более медленной работы программы, когда мы используем endl.

**4. Связанность потоков cin и cout**

Давайте теперь будем считывать числа из файла, перенаправляя cin и cout, и выводить в другой файл каждое число с новой строки.

#include <iostream>

#include <Windows.h>

#include <iomanip>

#include <ctime>

#include <fstream>

using namespace std;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int a;

clock\_t start = clock();

for (int i = 0; i < 100000; i++) {

cin >> a;

cout << a << endl;

}

clock\_t end = clock();

double duration = (float)(end - start) / CLOCKS\_PER\_SEC;

clock\_t start1 = clock();

for (int i = 0; i < 100000; i++) {

cin >> a;

cout << a << '\n';

}

clock\_t end1 = clock();

double duration1 = (float)(end1 - start1) / CLOCKS\_PER\_SEC;

printf("time = %.5f\n", duration);

printf("time = %.5f", duration1);

}

Посмотрим результаты замеров времени (первый результат endl, второй ‘\n’)

На первый взгляд данные результаты могут показаться странными. Может возникнуть вопрос: почему ‘\n’ и endl работают почти за одно и тоже время. По результатам исследования выше было выявлено, что endl работает медленней. Так почему же в данном случае разница во времени стала настолько мала?

Дело в том, что в данном примере мы также используем поток для ввода cin. А по умолчанию потоки cout и cin связаны между собой и перед каждым cin буфер потока cout сбрасывается. А как было выявлено, это работает намного медленнее. Так что же делать?

Есть решение. Для разрыва связи между cin и cout используем команду cin.tie(nullptr);

Результат выполнения:

Как видим, разница снова стала существенной.

**Выводы**

На основании результатов исследований можно сделать следующие выводы:

1) Потоки вывода (cout, cerr, ofstream) не сразу выводят содержимое, а буферизируют его.

2) Управляющий символ ‘\n’ указывает на перевод строки и это происходит только после сброса буфера, а манипулятор endl вызывает переход на новую строку и принудительно заставляет поток сбросить буфер.

3) Использование endl может приводить к снижению скорости вывода, ‘\n’ работает быстрее.

4) Используйте ‘\n’, когда важен весь вывод целиком и нужно экономить время.

5) Используйте endl, когда скорость не так важна, но важно сразу видеть результат (например, в отладке).