В соответствии с заданием нужно было написать функцию, которая обрезает пробелы в конце переданной ей строки. Функция должна быть написана в расчёте на работу с очень длинными строками с очень большим количеством пробелов, оптимизирована по количеству обращений к памяти.

Самое простое решение «в лоб» было бы просто, начиная с начала массива символов строки C-style посчитать общее число символов перебирая элементы до символа конца строки «\0». А затем вернуться назад, к последнему символу, отличному от пробела, установив символ конца строки сразу, после него. Но при таком решении мы сначала обращаемся к памяти N число раз, где N – это число символов в строке, а затем обращаемся к памяти еще M раз, где M – число пробелов в конце строки.

Альтернативным вариантом решения задачи было бы при первом прохождении строки с начала в конец запоминать положение первого найденного пробела и, если пройдя до конца строки, не обнаружили символов отличных от пробела – обрезать строку сразу за этим индексом. Если же мы встретим далее отличные от пробела символы, то продолжить поиск в массиве следующего начала интервала пробелов и т.д. В этом случае после прохождения строки от начала в конец и и найдя ее длину не нужно возвращаться по массиву для поиска последнего не пробельного символа и число обращений к памяти сокращается до N (числа символов в строке).

При реализации использовался компилятор GCC версии 12.2.0 и стандарт языка С++ 17.

void

TrimRight( char \*s )

{

//Определяем длину строки без пробелов

int length = 0;

int index = 0;

bool isSpaceLine = false;

**while** (s[length] != '\0')

{

while (s[length] != '\0' && s[length] != ' ')

++length;

if (s[length] == '\0')

break;

if (s[length] == ' ')

{

index = length;

isSpaceLine = true;

++length;

while (s[length] != '\0' && s[length] == ' ')

++length;

if (s[length] == '\0')

break;

if (s[length] != ' ')

isSpaceLine = false;

}

}

if (isSpaceLine == true)

s[index] = '\0';

};

void Trim\_Naive(char \* p\_char)

{

auto p\_begin\_or\_last\_non\_ws{p\_char};

for(;;)

{

switch(\*p\_char)

{

case '\0':

{

break;

}

case ' ':

{

++p\_char;

continue;

}

default:

{

p\_begin\_or\_last\_non\_ws = p\_char;

++p\_char;

continue;

}

}

break;

}

if(' ' == \*p\_begin\_or\_last\_non\_ws)

{

p\_begin\_or\_last\_non\_ws[0] = '\0';

}

else

{

p\_begin\_or\_last\_non\_ws[1] = '\0';

}

return;