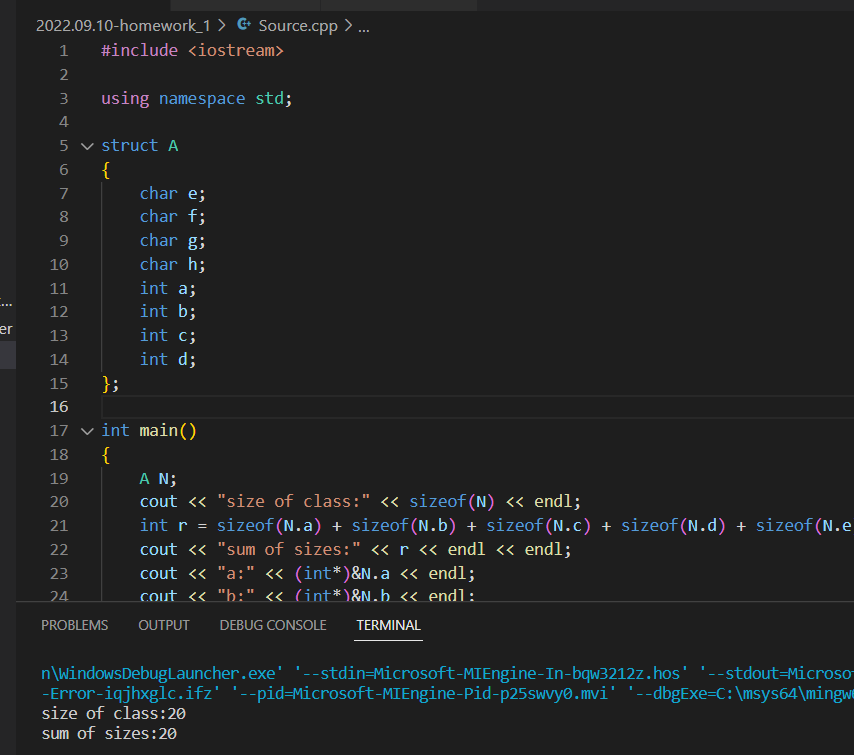
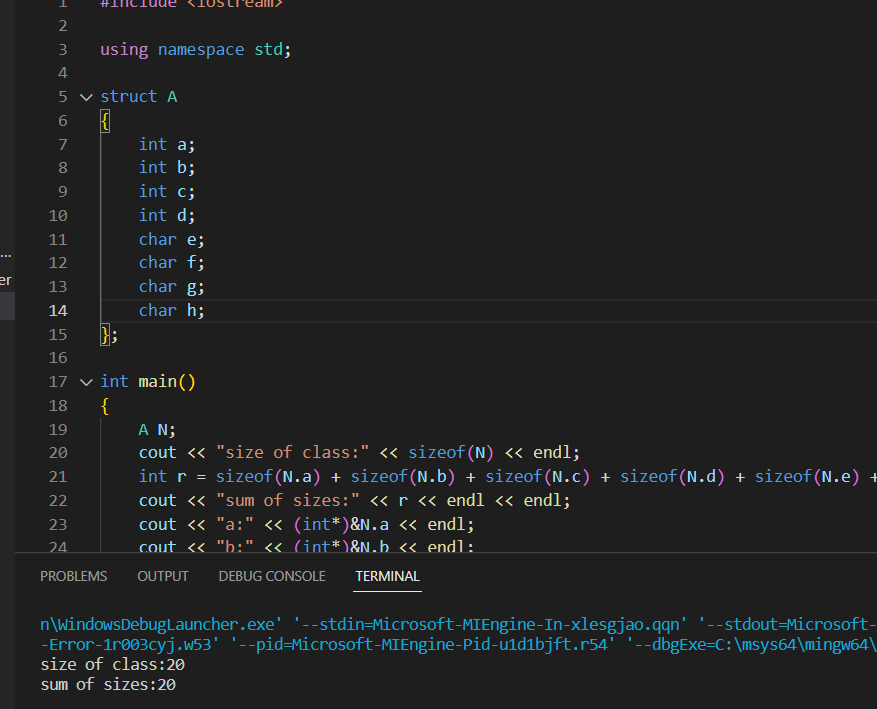
Эксперименты с памятью (задача №1)

Пусть выравнивание с структуре работает так:

1. Читаем первое поле, допустим это char, размер которого равен 1.
2. Далее читаем второе, допустим int (4).
3. Затем читаем снова char, но уже не «ячейкой памяти» равной 1, а 4 (один байт на char, а три дополнительных для выравнивания)

И поступаем аналогично с остальными полями, т.е. получается такая «сортировка». Если нам нужно добиться минимального размера класса, то логичнее всего отсортировать наши создаваемые переменные по количеству занимаемой памяти. Пусть наше предположение верно, проверим его.

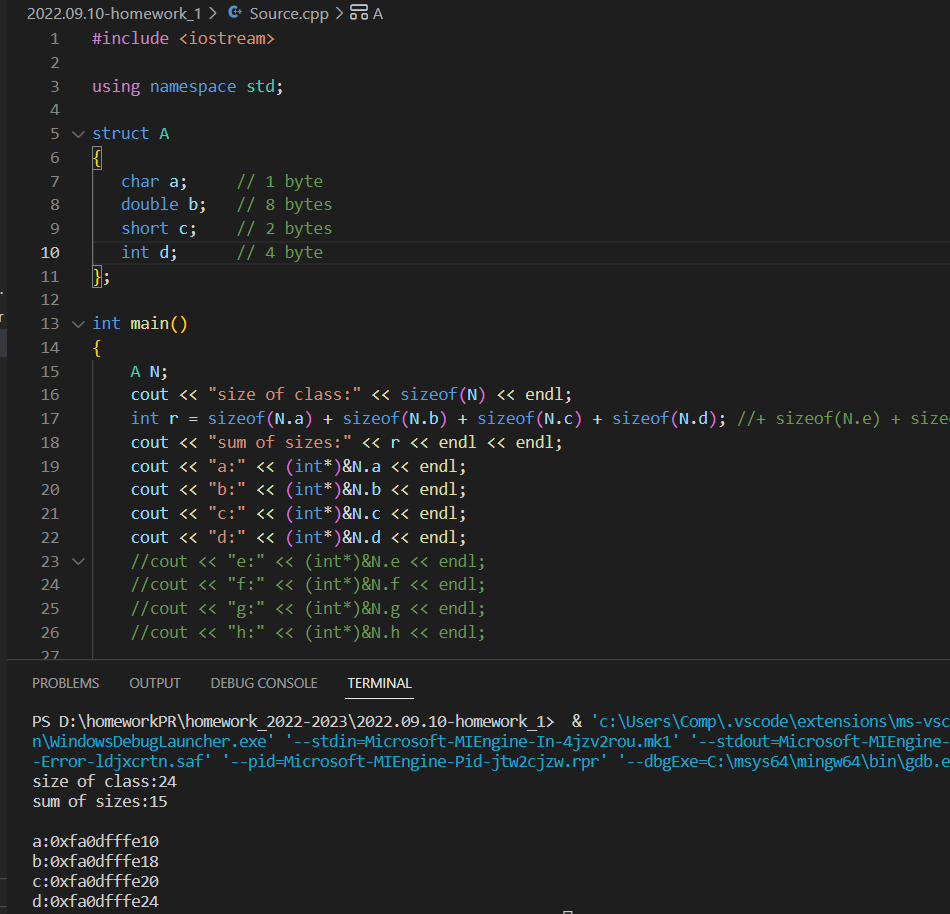


Наше предположение оказалось неверно (если бы оно было верно, то в одном случае мы бы получили размер класса равный 20, а в другом – 32).

Значит всё устроено хитрее.

Эксперименты – это здорово, но поступим рациональнее и залезем в гугл. Перейдя по первой ссылке (<https://docs.microsoft.com/ru-ru/cpp/cpp/alignment-cpp-declarations?view=msvc-170>) наткнемся на статью, в которой сказано: «Для простых типов данных компилятор назначает адреса, которые кратны размеру в байтах для типа данных. Например, компилятор назначает адреса переменным типа long , кратным 4, задав нижний 2 бита адреса равным нулю». Также в этой статье говориться, что компилятор делает размер структуры кратным 4.

Начнем проверку этой гипотезы с новым набором разных типов данных. Будем следить за размером и ячейками памяти.



Адреса:

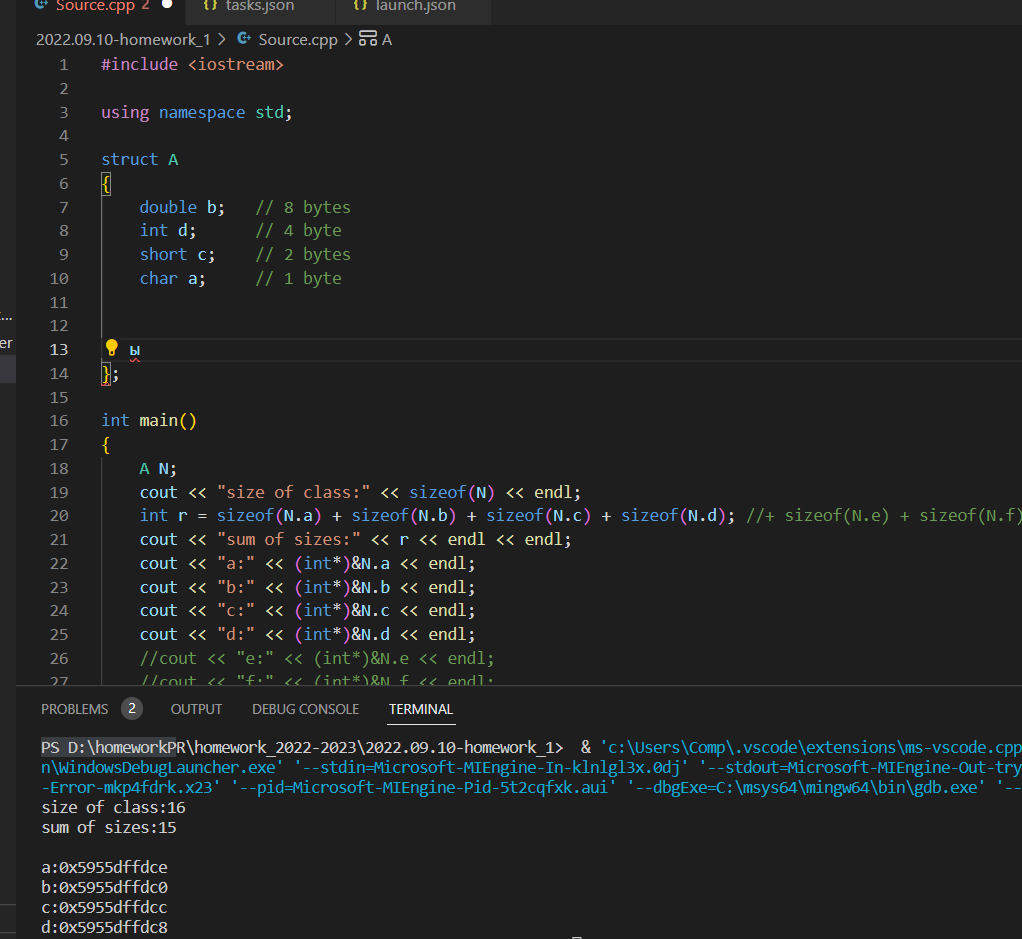
a:0xfa0dfffe10

b:0xfa0dfffe18

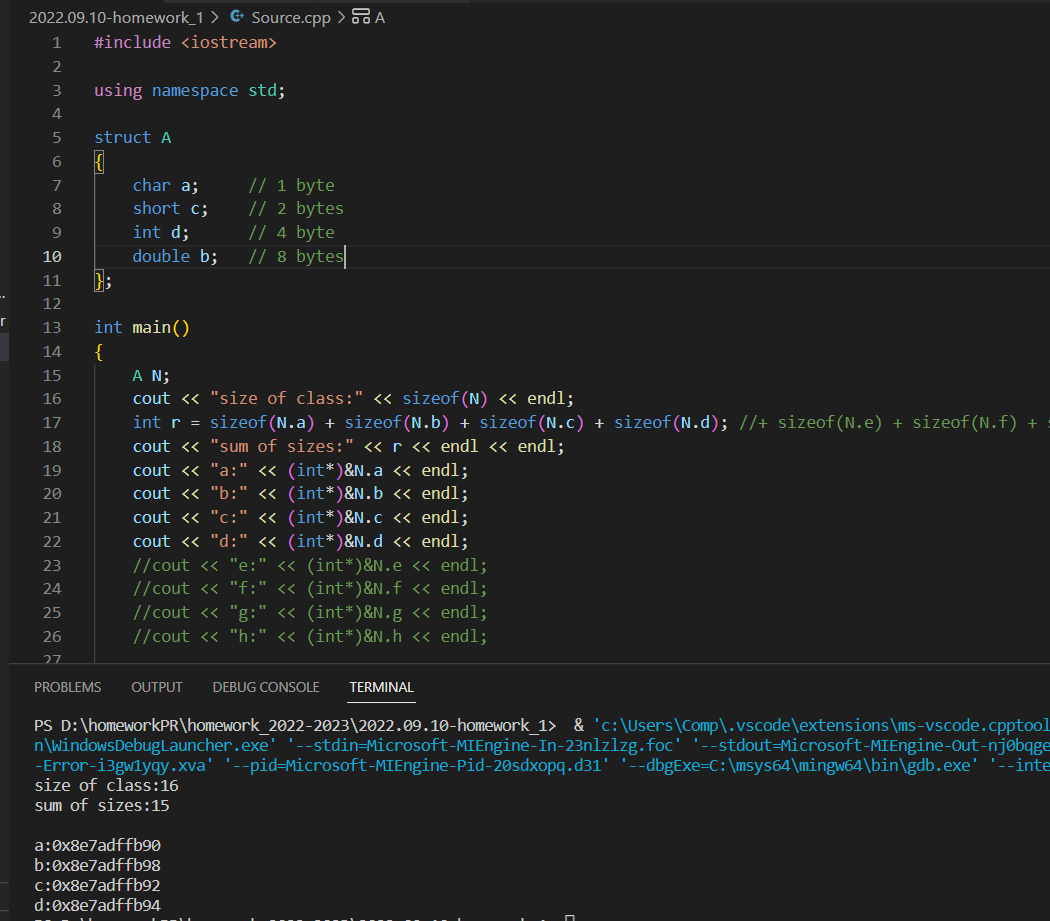
c:0xfa0dfffe20

d:0xfa0dfffe24

a записано в ячейку …10, ближайшая ячейка, чей адрес делится на 8 это …18. На этом примере гипотеза работает, переставим поля для минимального размера.

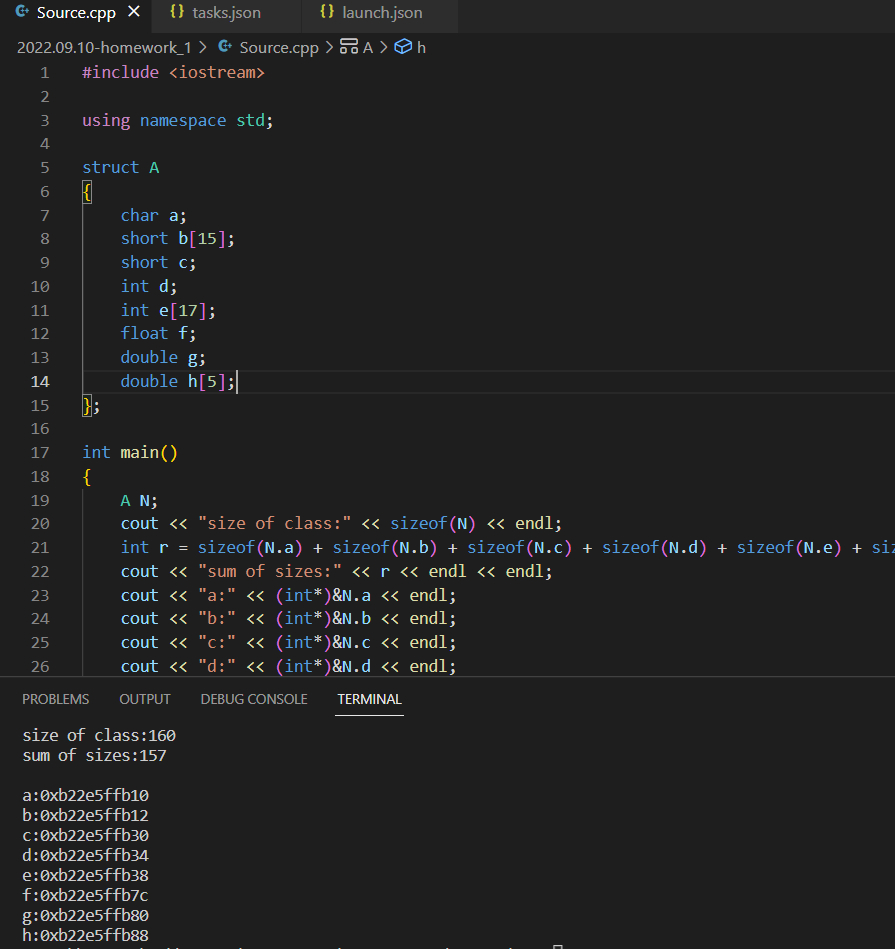


Получили минимальный размер, с учётом того, что компилятор выровнял размер до делящегося на 4. Если бы предыдущая гипотеза была верна, то «перевернув» запись мы бы получили максимальный размер, проверим.

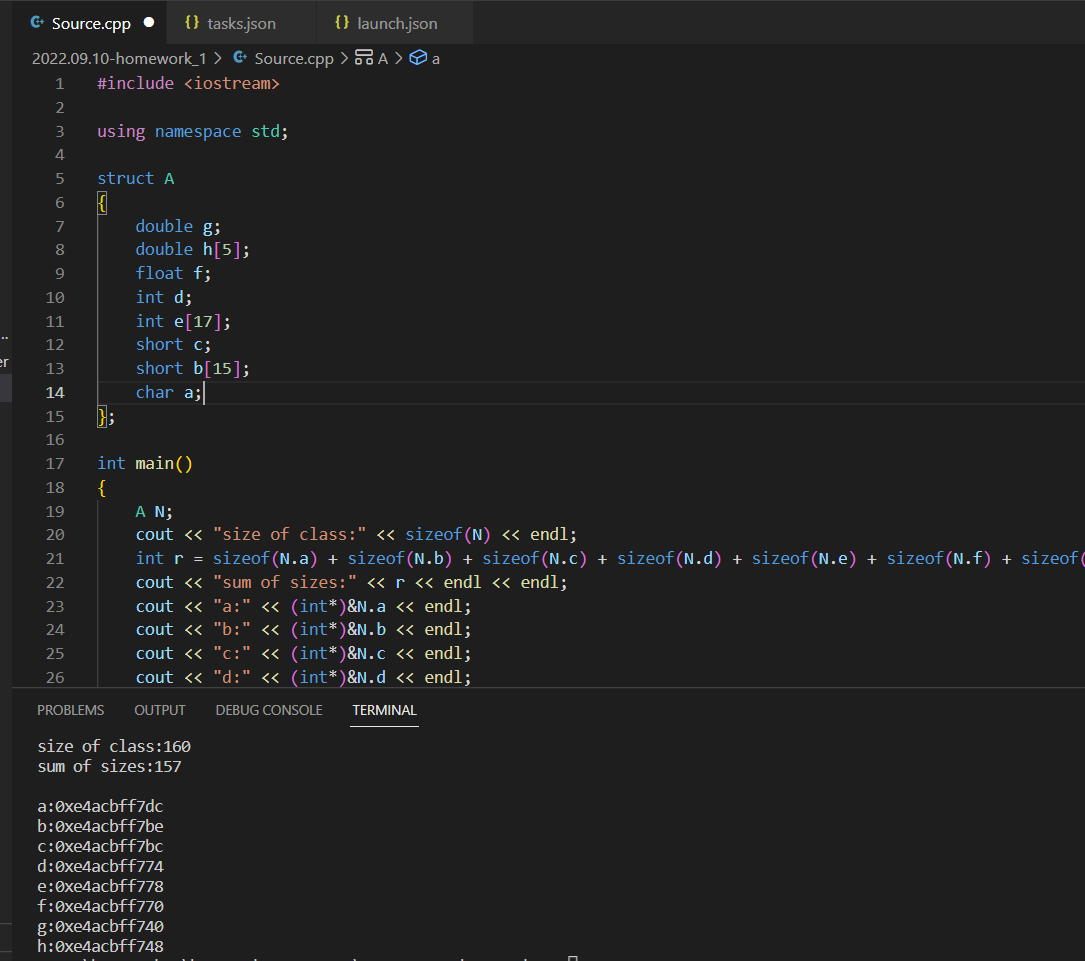


Но этого не происходит. Изначально компилятор взял ячейку с адресом …90, заполнил её, ячейка с адресом …91 не кратна 2, значит в нее ничего не записалось, а вот следующая (…92) кратна двум – значит в нее и в …93 записан short. Переведем из 16-ной с.с. адрес d (...b94) = …2964. Это число делится на 4, следовательно запишем в эту ячейку. И т.д.

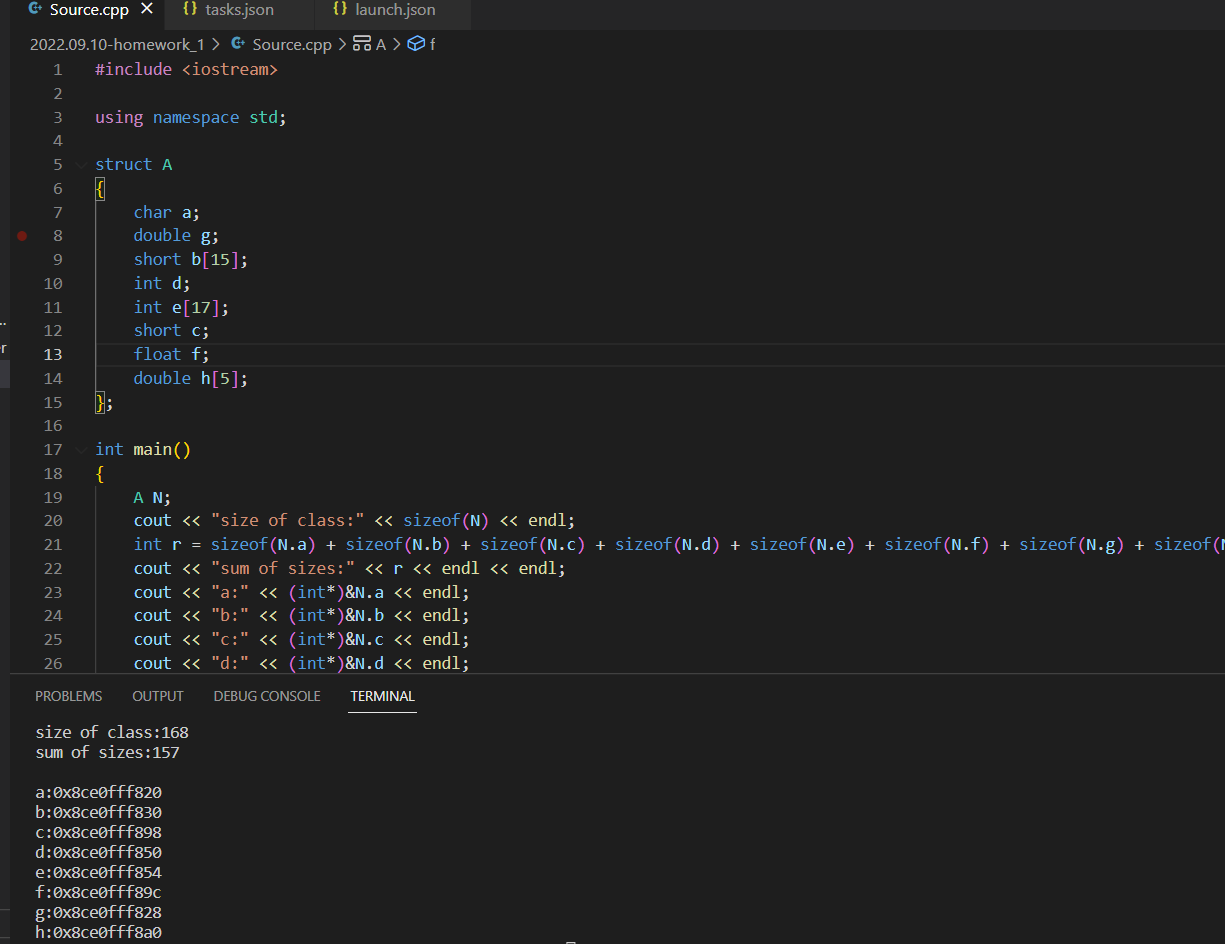
Теперь создадим 8 переменных и добьемся минимального и максимального размера класса.



Создали вот такую структуру, в ней уже получили минимальный размер, т.к ближайшее число, которое делится на 4 – это 160.



Теперь, зная принцип, найдем максимальный размер.



Максимальный размер = 168 байт. После записи char компилятор оставит 7 байт свободными и перейдет к ячейки кратной 8. После записи массива short оставит 2 байта. После записи short c оставит 2 байта. Итого 11 пустых байт.

Минимальный размер 160. Максимальный 168.

P.S. Компилятор g++