GCC Code Coverage Report

				Exec	Total	Coverage
Directory:	./					
File:	gtests/gtests.cpp		Lines:	100	100	100.0 %
Date:	2021-03-15 14:28:04		Branches:	73	300	24.3 %

```
Line Branch Exec Source
   1               #include <gtest/gtest.h>
   2               #include <cstring>
   3
   4               #include "utils.h"
   5
   6               #define ERR -1
   7
   8             8 TEST(data_compatator, data_comparator0) {
   9             4     size_t *a = nullptr;
  10             4     size_t *b = nullptr;
  11
      ✓✗✓✗
  12  ✗✓✗✗       4     EXPECT_EQ(ERR, date_comparator(a, b));
      ✗✗✗✗
  13             4 }
  14
  15             8 TEST(data_compatator, data_comparator1) {
  16             4     int equal = EQUAL;
  17
  18             4     size_t a[3] = {10, 10, 2010};
  19             4     size_t b[3] = {10, 10, 2010};
  20
      ✓✗✓✗
  21  ✗✓✗✗       4     EXPECT_EQ(equal, date_comparator(a, b));
      ✗✗✗✗
  22             4 }
  23
  24             8 TEST(data_compatator, data_comparator2) {
  25             4     size_t a0[3] = {10, 10, 2010};
  26             4     size_t b0[3] = {10, 10, 2011};
  27             4     size_t a1[3] = {10, 9, 2010};
  28             4     size_t b1[3] = {10, 10, 2010};
  29             4     size_t a2[3] = {9, 10, 2010};
  30             4     size_t b2[3] = {10, 10, 2010};
  31
      ✓✗✓✗
  32  ✗✓✗✗       4     EXPECT_EQ(RHS_IS_LARGER, date_comparator(a0, b0));
      ✗✗✗✗
      ✓✗✓✗
  33  ✗✓✗✗       4     EXPECT_EQ(RHS_IS_LARGER, date_comparator(a1, b1));
      ✗✗✗✗
      ✓✗✓✗
  34  ✗✓✗✗       4     EXPECT_EQ(RHS_IS_LARGER, date_comparator(a2, b2));
      ✗✗✗✗
  35             4 }
  36
  37             8 TEST(data_compatator, data_comparator3) {
  38             4     int lhs_is_larger = LHS_IS_LARGER;
  39
  40             4     size_t a3[3] = {10, 10, 2011};
  41             4     size_t b3[3] = {10, 10, 2010};
```

```
           4    size_t a4[3] = {10, 10, 2010};
           4    size_t b4[3] = {10, 9, 2010};
           4    size_t a5[3] = {10, 10, 2010};
           4    size_t b5[3] = {9, 10, 2010};

   ✓✗✓✗
   ✗✓✗✗    4    EXPECT_EQ(lhs_is_larger, date_comparator(a3, b3));
   ✗✗✗✗
   ✓✗✓✗
   ✗✓✗✗    4    EXPECT_EQ(lhs_is_larger, date_comparator(a4, b4));
   ✗✗✗✗
   ✓✗✓✗
   ✗✓✗✗    4    EXPECT_EQ(lhs_is_larger, date_comparator(a5, b5));
   ✗✗✗✗
           4}

           8TEST(date_parse, date_parse0) {
           4    char *date0 = nullptr;
     ✓✗    8    std::string date1 = "10:10:2022";
     ✓✗    8    std::string date2 = "10:13:2020";
     ✓✗    8    std::string date3 = "33:09:2010";

           4    size_t *a0 = nullptr;
           4    size_t a1[3] = {0, 0, 0};

   ✓✗✓✗
   ✗✓✗✗    4    EXPECT_EQ(ERR, parse_date(date0, a0));
   ✗✗✗✗
   ✓✗✓✗
   ✗✓✗✗    4    EXPECT_EQ(ERR, parse_date(date1.c_str(), a1));
   ✗✗✗✗
   ✓✗✓✗
   ✗✓✗✗    4    EXPECT_EQ(ERR, parse_date(date1.c_str(), a1));
   ✗✗✗✗
   ✓✗✓✗
   ✗✓✗✗    4    EXPECT_EQ(ERR, parse_date(date2.c_str(), a1));
   ✗✗✗✗
   ✓✗✓✗
   ✗✓✗✗    4    EXPECT_EQ(ERR, parse_date(date3.c_str(), a1));
   ✗✗✗✗
           4}

           8TEST(date_parse, date_parse1) {
     ✓✗    8    std::string date1 = "10:10:2021";

                size_t a0[3];
           4    size_t a1[3] = {10, 10, 2021};

     ✓✗    4    parse_date(date1.c_str(), &a0[0]);

   ✗✓✗✗
   ✗✗✗✗    4    EXPECT_TRUE(0 == std::memcmp(a0, a1, sizeof(a0)));
     ✗✗
           4}

           8TEST(tasks_comparator, tasks_comparator0) {
```

```
           4    Task task1 = {1, 1, {10, 10, 21}, nullptr};
           4    Task task2 = {1, 1, {10, 10, 20}, nullptr};

           4    Task task3 = {1, 1, {10, 10, 20}, nullptr};
           4    Task task4 = {1, 2, {10, 10, 20}, nullptr};


     ✓✗    4    int res1 = tasks_comparator(&task1, &task2);
     ✓✗    4    int res2 = tasks_comparator(&task3, &task4);

   ✓✗✗✓
   ✗✗✗✗    4    EXPECT_EQ(LHS_IS_LARGER, res1);
     ✗✗
   ✓✗✗✓
   ✗✗✗✗    4    EXPECT_EQ(RHS_IS_LARGER, res2);
     ✗✗
           4}

           8TEST(sort, sort) {
           4    Task task1 = {1, 1, {10, 10, 2020}, nullptr};
           4    Task task2 = {2, 1, {1, 10, 2020}, nullptr};
           4    Task task3 = {3, 3, {10, 9, 2020}, nullptr};
           4    Task task4 = {4, 2, {1, 10, 1817}, nullptr};
           4    Task task5 = {5, 10, {10, 10, 2001}, nullptr};

           4    Task tasks0[5] = {task1, task2, task3, task4, task5};
           4    Task tasks1[5] = {task2, task1, task4, task3, task5};

     ✓✗    4    sort(tasks0, 5);

   ✗✓✗✗
   ✗✗✗✗    4    EXPECT_TRUE(0 == std::memcmp(tasks0, tasks1, sizeof(tasks0)));
     ✗✗
           4}

           8TEST(grow_buffer, grow_buffer) {
           4    Tasks *tasks = create_array_of_tasks();

   ✓✗✓✗
   ✗✓✗✗    4    EXPECT_EQ(0, grow_tasks(tasks));
   ✗✗✗✗

   ✓✗✗✓
   ✗✗✗✗    4    EXPECT_EQ(START_SIZE_OF_TASKS_BUFFER * 2, tasks->cells_amount);
     ✗✗
           4    free(tasks->buffer);
           4    free(tasks);
           4}

           8TEST(push_back_task, push_back_task) {
     ✓✗    4    Tasks *tasks = create_array_of_tasks();
           4    size_t a[3] = {1, 2, 3};

           4    Task *task = (Task *) calloc(1, sizeof(Task));
           4    task->number = 1;
           4    task->priority = 1;
     ✓✓   16    for (size_t i = 0; i < 3; ++i) {
          12        task->date[i] = i + 1;
```

```
128                  }
129
     ✓✗✓✗
130 ✗✓✗✗    4        EXPECT_EQ(0, push_back_task(tasks, task));
     ✗✗✗✗
131
     ✓✗✗✓
132 ✗✗✗✗    4        EXPECT_EQ(1, tasks->tasks_amount);
       ✗✗
     ✓✗✗✓
133 ✗✗✗✗    4        EXPECT_EQ(1, tasks->buffer->number);
       ✗✗
     ✓✗✗✓
134 ✗✗✗✗    4        EXPECT_EQ(1, tasks->buffer->priority);
       ✗✗
     ✗✓✗✗
135 ✗✗✗✗    4        EXPECT_TRUE(0 == std::memcmp(a, tasks->buffer->date, sizeof(a)));
       ✗✗
     ✓✗✗✓
136 ✗✗✗✗    4        EXPECT_EQ(nullptr, tasks->buffer->description);
       ✗✗
137
138             4        free(tasks);
139             4 }
140
141             2 int main(int argc, char **argv) {
142             2        testing::InitGoogleTest(&argc, argv);
143             2        return RUN_ALL_TESTS();
144               }
```

Generated by: GCOVR (Version 4.2)