**Лабораторная работа №3**

**Библиотеки**

**Цель работы:** Получение практических навыков в работе со статическими и динамическими библиотеками.

**Функционал для библиотек:**

Любая библиотека из данной лабораторной работы должна содержать следующий функционал:

* Итеративную функцию бинарного поиска;
* Рекурсивную функцию бинарного поиска;
* Массив целых чисел размером 1024 элемента.

Реализацию функций можно взять [отсюда](https://rosettacode.org/wiki/Binary_search#C), а массив сгенерировать [тут](https://onlinetools.com/integer/create-integer-array).

**Постановка задачи для Windows:**

**Проект <Ваши Инициалы>Lib:**

Должен содержать код для создания статической библиотеки с заданным функционалом (см. выше). Код должен быть кроссплатформенным, так как будет применяться и для Linux/macOS (Для тех кто не берёт код из указанного выше источника).

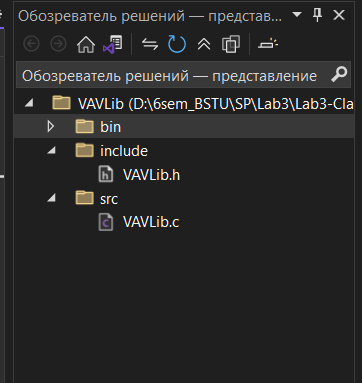
**VAVLib.c:**

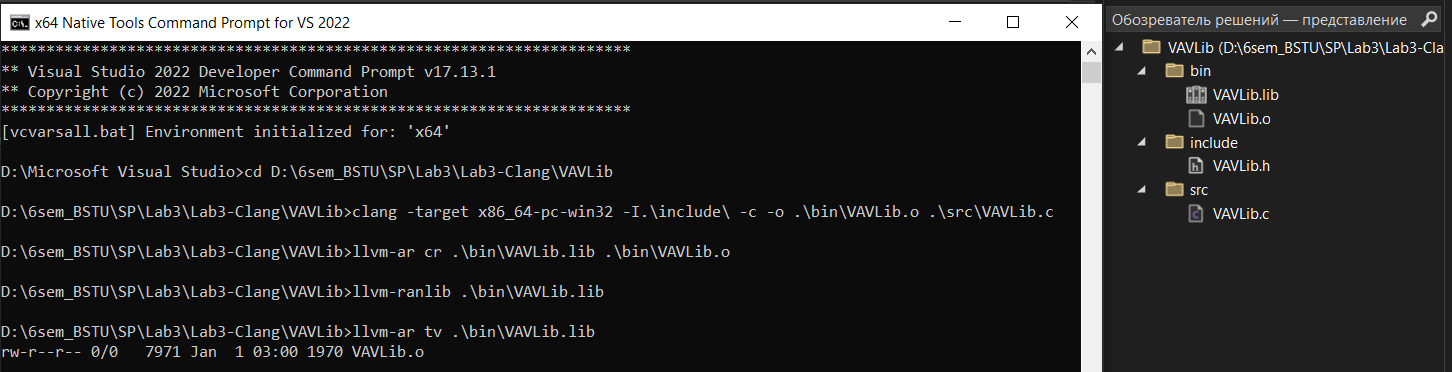
|  |
| --- |
| #include "VAVLib.h"  const int array[1024] = {  4, 13, 23, 32, 41, 51, 61, 70, 79, 89, 98, 108,  118, 127, 136, 146, 156, 165, 174, 184, 194, 203, 212, 222,  232, 241, 251, 260, 269, 279, 288, 298, 307, 317, 327, 336,  345, 355, 364, 374, 384, 393, 402, 412, 422, 431, 440, 450,  460, 469, 478, 488, 498, 507, 517, 526, 535, 545, 554, 564,  574, 583, 592, 602, 611, 621, 630, 640, 650, 659, 668, 678,  688, 697, 706, 716, 725, 735, 745, 754, 763, 773, 782, 792,  802, 811, 820, 830, 840, 849, 858, 868, 878, 887, 896, 906,  915, 925, 935, 944, 953, 963, 972, 982, 992, 1001, 1010, 1020,  1030, 1039, 1049, 1058, 1067, 1077, 1086, 1096, 1106, 1115, 1124, 1134,  1143, 1153, 1163, 1172, 1181, 1191, 1201, 1210, 1219, 1229, 1239, 1248,  1257, 1267, 1277, 1286, 1295, 1305, 1314, 1324, 1334, 1343, 1352, 1362,  1371, 1381, 1391, 1400, 1409, 1419, 1429, 1438, 1447, 1457, 1466, 1476,  1486, 1495, 1504, 1514, 1524, 1533, 1542, 1552, 1561, 1571, 1581, 1590,  1599, 1609, 1619, 1628, 1637, 1647, 1656, 1666, 1676, 1685, 1694, 1704,  1713, 1723, 1733, 1742, 1751, 1761, 1771, 1780, 1789, 1799, 1808, 1818,  1828, 1837, 1846, 1856, 1866, 1875, 1884, 1894, 1903, 1913, 1923, 1932,  1941, 1951, 1961, 1970, 1979, 1989, 1998, 2008, 2018, 2027, 2036, 2046,  2055, 2065, 2075, 2084, 2093, 2103, 2112, 2122, 2132, 2141, 2150, 2160,  2170, 2179, 2188, 2198, 2207, 2217, 2227, 2236, 2245, 2255, 2264, 2274,  2284, 2293, 2302, 2312, 2322, 2331, 2340, 2350, 2359, 2369, 2379, 2388,  2397, 2407, 2417, 2426, 2435, 2445, 2454, 2464, 2474, 2483, 2492, 2502,  2511, 2521, 2531, 2540, 2549, 2559, 2568, 2578, 2588, 2597, 2606, 2616,  2625, 2635, 2645, 2654, 2663, 2673, 2683, 2692, 2701, 2711, 2720, 2730,  2740, 2749, 2758, 2768, 2777, 2787, 2797, 2806, 2815, 2825, 2834, 2844,  2854, 2863, 2872, 2882, 2892, 2901, 2910, 2920, 2929, 2939, 2949, 2958,  2967, 2977, 2987, 2996, 3005, 3015, 3024, 3034, 3044, 3053, 3062, 3072,  3081, 3091, 3101, 3110, 3119, 3129, 3138, 3148, 3158, 3167, 3176, 3186,  3195, 3205, 3215, 3224, 3233, 3243, 3253, 3262, 3271, 3281, 3290, 3300,  3310, 3319, 3328, 3338, 3347, 3357, 3367, 3376, 3385, 3395, 3404, 3414,  3424, 3433, 3442, 3452, 3461, 3471, 3481, 3490, 3499, 3509, 3518, 3528,  3538, 3547, 3556, 3566, 3576, 3585, 3594, 3604, 3613, 3623, 3633, 3642,  3651, 3661, 3670, 3680, 3690, 3699, 3708, 3718, 3727, 3737, 3747, 3756,  3765, 3775, 3784, 3794, 3804, 3813, 3822, 3832, 3841, 3851, 3861, 3870,  3879, 3889, 3898, 3908, 3918, 3927, 3936, 3946, 3955, 3965, 3975, 3984,  3993, 4003, 4013, 4022, 4031, 4041, 4050, 4060, 4070, 4079, 4088, 4098,  4107, 4117, 4127, 4136, 4145, 4155, 4164, 4174, 4184, 4193, 4202, 4212,  4221, 4231, 4241, 4250, 4259, 4269, 4279, 4288, 4297, 4307, 4316, 4326,  4336, 4345, 4354, 4364, 4373, 4383, 4393, 4402, 4411, 4421, 4430, 4440,  4450, 4459, 4468, 4478, 4487, 4497, 4507, 4516, 4525, 4535, 4544, 4554,  4564, 4573, 4582, 4592, 4602, 4611, 4620, 4630, 4639, 4649, 4659, 4668,  4677, 4687, 4696, 4706, 4716, 4725, 4734, 4744, 4753, 4763, 4773, 4782,  4791, 4801, 4811, 4820, 4829, 4839, 4848, 4858, 4868, 4877, 4886, 4896,  4905, 4915, 4925, 4934, 4943, 4953, 4963, 4972, 4981, 4991, 5000, 5010,  5020, 5029, 5038, 5048, 5057, 5067, 5077, 5086, 5095, 5105, 5114, 5124,  5134, 5143, 5152, 5162, 5171, 5181, 5191, 5200, 5209, 5219, 5228, 5238,  5248, 5257, 5266, 5276, 5286, 5295, 5304, 5314, 5323, 5333, 5343, 5352,  5361, 5371, 5380, 5390, 5400, 5409, 5418, 5428, 5437, 5447, 5457, 5466,  5475, 5485, 5494, 5504, 5514, 5523, 5532, 5542, 5551, 5561, 5571, 5580,  5589, 5599, 5608, 5618, 5628, 5637, 5646, 5656, 5666, 5675, 5684, 5694,  5703, 5713, 5723, 5732, 5741, 5751, 5760, 5770, 5780, 5789, 5798, 5808,  5817, 5827, 5837, 5846, 5855, 5865, 5874, 5884, 5894, 5903, 5912, 5922,  5931, 5941, 5951, 5960, 5969, 5979, 5989, 5998, 6007, 6017, 6026, 6036,  6046, 6055, 6064, 6074, 6083, 6093, 6103, 6112, 6121, 6131, 6140, 6150,  6160, 6169, 6178, 6188, 6197, 6207, 6217, 6226, 6235, 6245, 6254, 6264,  6274, 6283, 6292, 6302, 6311, 6321, 6331, 6340, 6349, 6359, 6368, 6378,  6388, 6397, 6406, 6416, 6426, 6435, 6444, 6454, 6463, 6473, 6483, 6492,  6501, 6511, 6520, 6530, 6540, 6549, 6558, 6568, 6577, 6587, 6597, 6606,  6615, 6625, 6634, 6644, 6654, 6663, 6672, 6682, 6691, 6701, 6711, 6720,  6729, 6739, 6749, 6758, 6767, 6777, 6786, 6796, 6806, 6815, 6824, 6834,  6843, 6853, 6863, 6872, 6881, 6891, 6901, 6910, 6919, 6929, 6938, 6948,  6958, 6967, 6976, 6986, 6995, 7005, 7015, 7024, 7033, 7043, 7052, 7062,  7072, 7081, 7090, 7100, 7109, 7119, 7129, 7138, 7147, 7157, 7166, 7176,  7186, 7195, 7204, 7214, 7223, 7233, 7243, 7252, 7261, 7271, 7280, 7290,  7300, 7309, 7318, 7328, 7337, 7347, 7357, 7366, 7375, 7385, 7394, 7404,  7414, 7423, 7432, 7442, 7451, 7461, 7471, 7480, 7489, 7499, 7508, 7518,  7528, 7537, 7546, 7556, 7566, 7575, 7584, 7594, 7603, 7613, 7623, 7632,  7641, 7651, 7660, 7670, 7680, 7689, 7698, 7708, 7717, 7727, 7737, 7746,  7755, 7765, 7774, 7784, 7794, 7803, 7812, 7822, 7831, 7841, 7851, 7860,  7869, 7879, 7888, 7898, 7908, 7917, 7926, 7936, 7946, 7955, 7964, 7974,  7983, 7993, 8003, 8012, 8021, 8031, 8040, 8050, 8060, 8069, 8078, 8088,  8097, 8107, 8117, 8126, 8135, 8145, 8154, 8164, 8174, 8183, 8192, 8202,  8211, 8221, 8231, 8240, 8249, 8259, 8268, 8278, 8288, 8297, 8306, 8316,  8325, 8335, 8345, 8354, 8363, 8373, 8382, 8392, 8402, 8411, 8420, 8430,  8439, 8449, 8459, 8468, 8477, 8487, 8496, 8506, 8516, 8525, 8534, 8544,  8553, 8563, 8573, 8582, 8591, 8601, 8611, 8620, 8629, 8639, 8648, 8658,  8668, 8677, 8686, 8696, 8705, 8715, 8725, 8734, 8743, 8753, 8762, 8772,  8782, 8791, 8800, 8810, 8819, 8829, 8839, 8848, 8857, 8867, 8876, 8886,  8896, 8905, 8914, 8924, 8933, 8943, 8953, 8962, 8971, 8981, 8991, 9000,  9009, 9019, 9028, 9038, 9048, 9057, 9066, 9076, 9085, 9095, 9105, 9114,  9123, 9133, 9142, 9152, 9162, 9171, 9180, 9190, 9199, 9209, 9219, 9228,  9237, 9247, 9256, 9266, 9276, 9285, 9294, 9304, 9313, 9323, 9333, 9342,  9351, 9361, 9371, 9380, 9389, 9399, 9408, 9418, 9428, 9437, 9446, 9456,  9465, 9475, 9485, 9494, 9503, 9513, 9522, 9532, 9542, 9551, 9560, 9570,  9579, 9589, 9599, 9608, 9617, 9627, 9636, 9646, 9656, 9665, 9674, 9684,  9693, 9703, 9713, 9722  };  int bsearch\_vav(const int\* a, int n, int x) {  int i = 0, j = n - 1;  while (i <= j) {  int k = i + ((j - i) / 2);  if (a[k] == x) {  return k;  }  else if (a[k] < x) {  i = k + 1;  }  else {  j = k - 1;  }  }  return -1;  }  int bsearch\_r\_vav(const int\* a, int x, int i, int j) {  if (j < i) {  return -1;  }  int k = i + ((j - i) / 2);  if (a[k] == x) {  return k;  }  else if (a[k] < x) {  return bsearch\_r\_vav(a, x, k + 1, j);  }  else {  return bsearch\_r\_vav(a, x, i, k - 1);  }  } |

**VAVLib.h:**

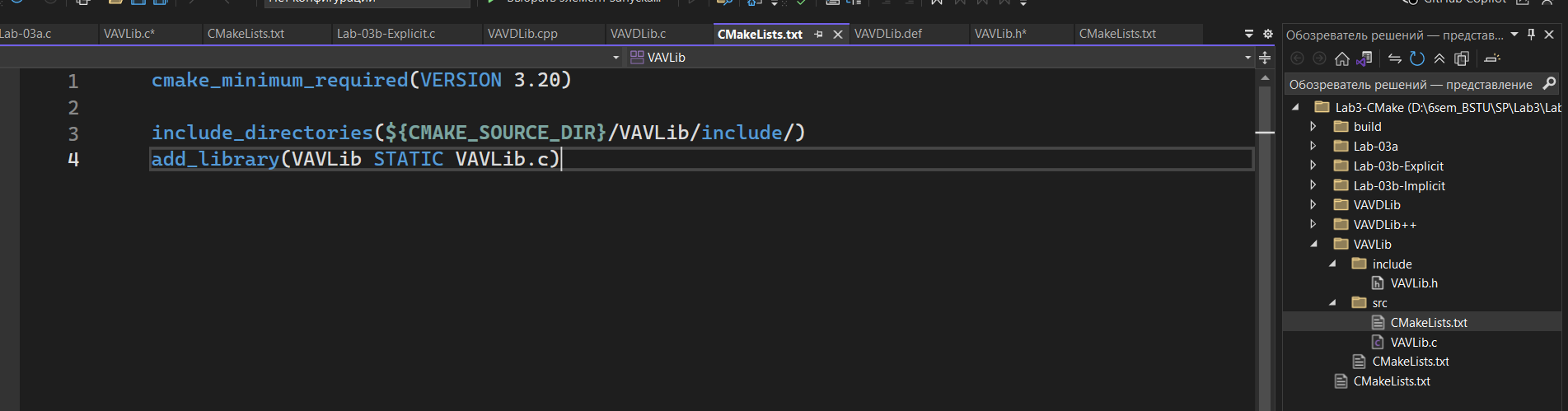
|  |
| --- |
| #include <stdio.h>  extern const int array[1024];  int bsearch\_vav(const int\* a, int n, int x);  int bsearch\_r\_vav(const int\* a, int x, int i, int j); |

Clang





CMake



**Проект Lab-03a:**

Разработать клиентское приложение которое будет использовать статическую библиотеку. Код также должен быть кроссплатформенным, так как и в этом случае, код будет применяться для Linux/macOS. Приложение должно быть консольным и может быть вызывано в виде:

***Lab-03a <Имя вызываемой функции> [Искомое число]***

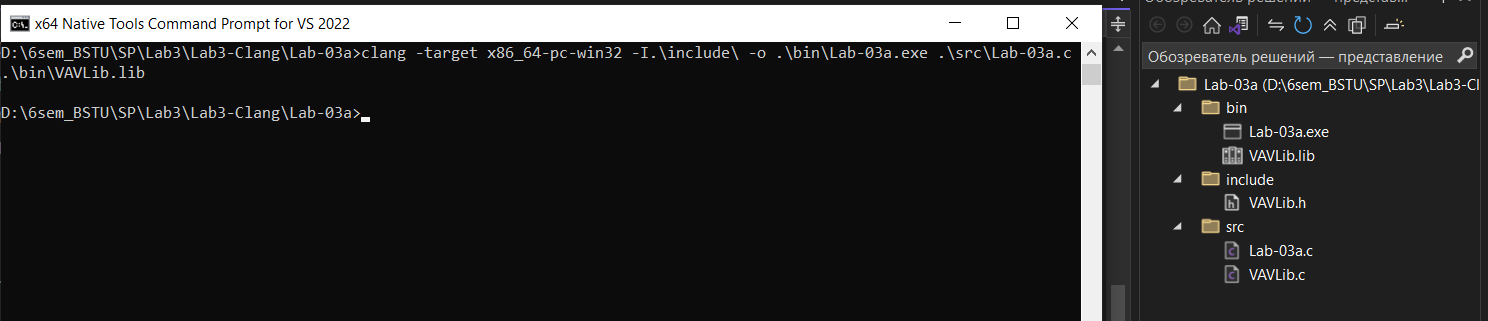
Если искомое число не задано, то оно запрашивается для ввода с консоли. В результате выполнения этого приложения должен быть выведен один из следующих результатов:

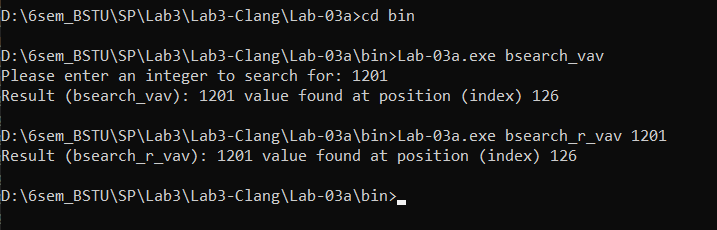
* Не указана вызываемая функция!
* <Имя вызванной функции>: Заданное число не найдено!
* <Имя вызванной функции>: Число <Х> найдено на позиции <Y>!

**Lab-03a.c:**

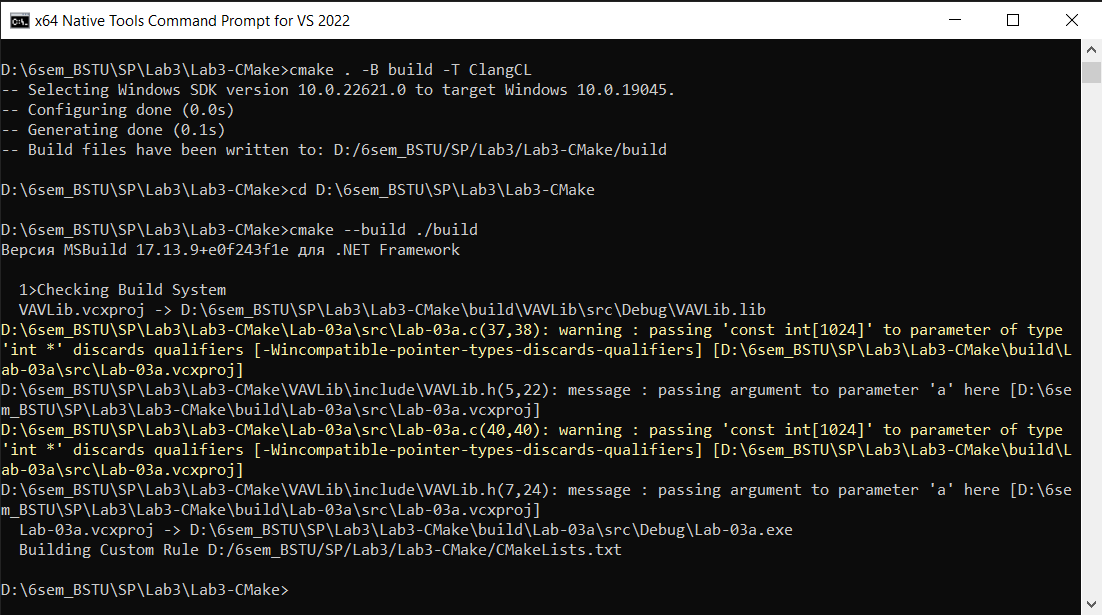
|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include "VAVLib.h"  int main(int arg\_count, char\* arg\_vector[]) {  int value\_to\_find = 0;  if (arg\_count < 2) {  fprintf(stderr, "Error (you didn't write the method)\n");  fprintf(stderr, "Available methods: 'bsearch\_vav', 'bsearch\_r\_vav'\n");  return 1;  }  char\* search\_method\_name = arg\_vector[1];  if (arg\_count > 2) {  value\_to\_find = atoi(arg\_vector[2]);  if (value\_to\_find == 0 && strcmp(arg\_vector[2], "0") != 0) {  fprintf(stderr, "Error (The second argument '%s' is not a valid number)\n", arg\_vector[2]);  return 1;  }  }  else {  printf("Please enter an integer to search for: ");  if (scanf\_s("%d", &value\_to\_find) != 1) {  fprintf(stderr, "Input error (An integer was expected)\n");  return 1;  }  }  int found\_at\_index = -1;  const int last\_element\_index = 1024 - 1;  if (strcmp(search\_method\_name, "bsearch\_vav") == 0) {  found\_at\_index = bsearch\_vav(array, last\_element\_index, value\_to\_find);  }  else if (strcmp(search\_method\_name, "bsearch\_r\_vav") == 0) {  found\_at\_index = bsearch\_r\_vav(array, value\_to\_find, 0, last\_element\_index);  }  else {  fprintf(stderr, "Error: Unknown search method for '%s'\n", search\_method\_name);  fprintf(stderr, "Use 'bsearch\_vav' or 'bsearch\_r\_vav'.\n");  return 1;  }  if (found\_at\_index == -1) {  printf("Result (%s): %d value is not found in the dataset\n",  search\_method\_name, value\_to\_find);  }  else {  printf("Result (%s): %d value found at position (index) %d\n",  search\_method\_name, value\_to\_find, found\_at\_index);  }  return 0;  } |

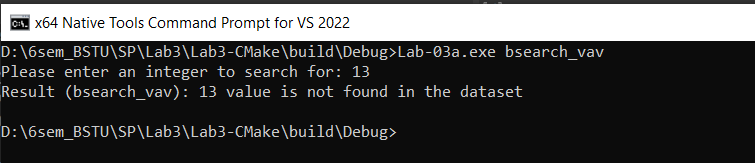
**Clang**





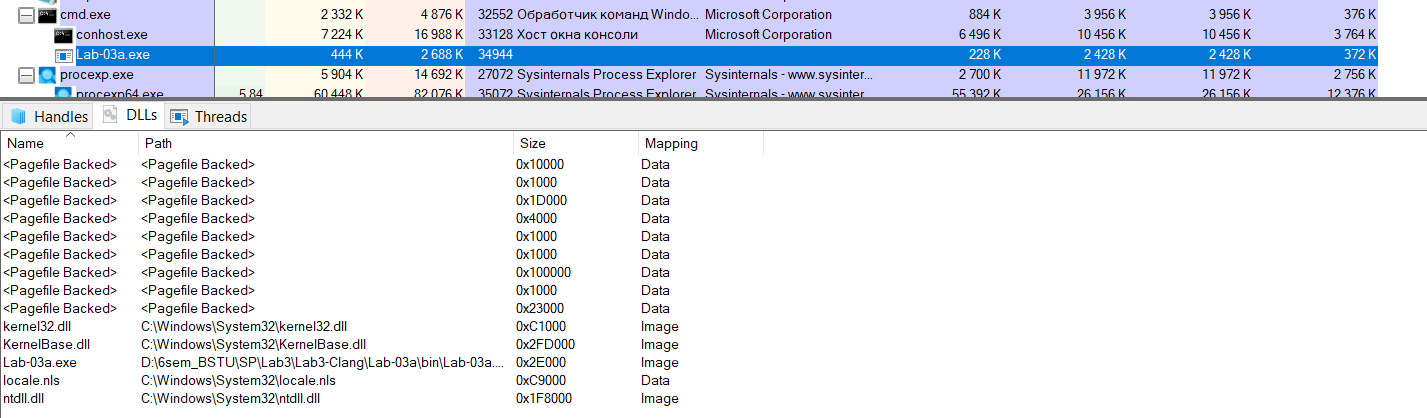
**CMake**





Проверить факт подключения библиотеки через Process Explorer.

***Вопрос:*** *Является ли это возможным? Почему?*

**

Когда библиотека линкуется статически, ее код копируется непосредственно в конечный исполняемый файл. Библиотека становится неотъемлемой частью самого исполняемого файла. Process Explorer показывает динамически загруженные модули.

**Проект <Ваши Инициалы>DLib:**

Разработать динамическую библиотеку на языке **C** с заданным функционалом (см. выше). Также должна быть реализована функция **DllMain** c логированием событий в некоторый файл (путь к файлу должен быть абсолютным).

Экспорт функций и переменных организовать через **.def** файл так, чтобы:

* Одна из функций экспортировалась без имени;
* Одна из функций экспортировалась с именем отличающимся от оригинального.

**VAVDLib.c**

|  |
| --- |
| #include <windows.h>  #include <stdio.h>  #include <errno.h>  #include <time.h>  #include "VAVDLib.h"  const int array[1024] = {  4, 13, 23, 32, 41, 51, 61, 70, 79, 89, 98, 108,  118, 127, 136, 146, 156, 165, 174, 184, 194, 203, 212, 222,  232, 241, 251, 260, 269, 279, 288, 298, 307, 317, 327, 336,  345, 355, 364, 374, 384, 393, 402, 412, 422, 431, 440, 450,  460, 469, 478, 488, 498, 507, 517, 526, 535, 545, 554, 564,  574, 583, 592, 602, 611, 621, 630, 640, 650, 659, 668, 678,  688, 697, 706, 716, 725, 735, 745, 754, 763, 773, 782, 792,  802, 811, 820, 830, 840, 849, 858, 868, 878, 887, 896, 906,  915, 925, 935, 944, 953, 963, 972, 982, 992, 1001, 1010, 1020,  1030, 1039, 1049, 1058, 1067, 1077, 1086, 1096, 1106, 1115, 1124, 1134,  1143, 1153, 1163, 1172, 1181, 1191, 1201, 1210, 1219, 1229, 1239, 1248,  1257, 1267, 1277, 1286, 1295, 1305, 1314, 1324, 1334, 1343, 1352, 1362,  1371, 1381, 1391, 1400, 1409, 1419, 1429, 1438, 1447, 1457, 1466, 1476,  1486, 1495, 1504, 1514, 1524, 1533, 1542, 1552, 1561, 1571, 1581, 1590,  1599, 1609, 1619, 1628, 1637, 1647, 1656, 1666, 1676, 1685, 1694, 1704,  1713, 1723, 1733, 1742, 1751, 1761, 1771, 1780, 1789, 1799, 1808, 1818,  1828, 1837, 1846, 1856, 1866, 1875, 1884, 1894, 1903, 1913, 1923, 1932,  1941, 1951, 1961, 1970, 1979, 1989, 1998, 2008, 2018, 2027, 2036, 2046,  2055, 2065, 2075, 2084, 2093, 2103, 2112, 2122, 2132, 2141, 2150, 2160,  2170, 2179, 2188, 2198, 2207, 2217, 2227, 2236, 2245, 2255, 2264, 2274,  2284, 2293, 2302, 2312, 2322, 2331, 2340, 2350, 2359, 2369, 2379, 2388,  2397, 2407, 2417, 2426, 2435, 2445, 2454, 2464, 2474, 2483, 2492, 2502,  2511, 2521, 2531, 2540, 2549, 2559, 2568, 2578, 2588, 2597, 2606, 2616,  2625, 2635, 2645, 2654, 2663, 2673, 2683, 2692, 2701, 2711, 2720, 2730,  2740, 2749, 2758, 2768, 2777, 2787, 2797, 2806, 2815, 2825, 2834, 2844,  2854, 2863, 2872, 2882, 2892, 2901, 2910, 2920, 2929, 2939, 2949, 2958,  2967, 2977, 2987, 2996, 3005, 3015, 3024, 3034, 3044, 3053, 3062, 3072,  3081, 3091, 3101, 3110, 3119, 3129, 3138, 3148, 3158, 3167, 3176, 3186,  3195, 3205, 3215, 3224, 3233, 3243, 3253, 3262, 3271, 3281, 3290, 3300,  3310, 3319, 3328, 3338, 3347, 3357, 3367, 3376, 3385, 3395, 3404, 3414,  3424, 3433, 3442, 3452, 3461, 3471, 3481, 3490, 3499, 3509, 3518, 3528,  3538, 3547, 3556, 3566, 3576, 3585, 3594, 3604, 3613, 3623, 3633, 3642,  3651, 3661, 3670, 3680, 3690, 3699, 3708, 3718, 3727, 3737, 3747, 3756,  3765, 3775, 3784, 3794, 3804, 3813, 3822, 3832, 3841, 3851, 3861, 3870,  3879, 3889, 3898, 3908, 3918, 3927, 3936, 3946, 3955, 3965, 3975, 3984,  3993, 4003, 4013, 4022, 4031, 4041, 4050, 4060, 4070, 4079, 4088, 4098,  4107, 4117, 4127, 4136, 4145, 4155, 4164, 4174, 4184, 4193, 4202, 4212,  4221, 4231, 4241, 4250, 4259, 4269, 4279, 4288, 4297, 4307, 4316, 4326,  4336, 4345, 4354, 4364, 4373, 4383, 4393, 4402, 4411, 4421, 4430, 4440,  4450, 4459, 4468, 4478, 4487, 4497, 4507, 4516, 4525, 4535, 4544, 4554,  4564, 4573, 4582, 4592, 4602, 4611, 4620, 4630, 4639, 4649, 4659, 4668,  4677, 4687, 4696, 4706, 4716, 4725, 4734, 4744, 4753, 4763, 4773, 4782,  4791, 4801, 4811, 4820, 4829, 4839, 4848, 4858, 4868, 4877, 4886, 4896,  4905, 4915, 4925, 4934, 4943, 4953, 4963, 4972, 4981, 4991, 5000, 5010,  5020, 5029, 5038, 5048, 5057, 5067, 5077, 5086, 5095, 5105, 5114, 5124,  5134, 5143, 5152, 5162, 5171, 5181, 5191, 5200, 5209, 5219, 5228, 5238,  5248, 5257, 5266, 5276, 5286, 5295, 5304, 5314, 5323, 5333, 5343, 5352,  5361, 5371, 5380, 5390, 5400, 5409, 5418, 5428, 5437, 5447, 5457, 5466,  5475, 5485, 5494, 5504, 5514, 5523, 5532, 5542, 5551, 5561, 5571, 5580,  5589, 5599, 5608, 5618, 5628, 5637, 5646, 5656, 5666, 5675, 5684, 5694,  5703, 5713, 5723, 5732, 5741, 5751, 5760, 5770, 5780, 5789, 5798, 5808,  5817, 5827, 5837, 5846, 5855, 5865, 5874, 5884, 5894, 5903, 5912, 5922,  5931, 5941, 5951, 5960, 5969, 5979, 5989, 5998, 6007, 6017, 6026, 6036,  6046, 6055, 6064, 6074, 6083, 6093, 6103, 6112, 6121, 6131, 6140, 6150,  6160, 6169, 6178, 6188, 6197, 6207, 6217, 6226, 6235, 6245, 6254, 6264,  6274, 6283, 6292, 6302, 6311, 6321, 6331, 6340, 6349, 6359, 6368, 6378,  6388, 6397, 6406, 6416, 6426, 6435, 6444, 6454, 6463, 6473, 6483, 6492,  6501, 6511, 6520, 6530, 6540, 6549, 6558, 6568, 6577, 6587, 6597, 6606,  6615, 6625, 6634, 6644, 6654, 6663, 6672, 6682, 6691, 6701, 6711, 6720,  6729, 6739, 6749, 6758, 6767, 6777, 6786, 6796, 6806, 6815, 6824, 6834,  6843, 6853, 6863, 6872, 6881, 6891, 6901, 6910, 6919, 6929, 6938, 6948,  6958, 6967, 6976, 6986, 6995, 7005, 7015, 7024, 7033, 7043, 7052, 7062,  7072, 7081, 7090, 7100, 7109, 7119, 7129, 7138, 7147, 7157, 7166, 7176,  7186, 7195, 7204, 7214, 7223, 7233, 7243, 7252, 7261, 7271, 7280, 7290,  7300, 7309, 7318, 7328, 7337, 7347, 7357, 7366, 7375, 7385, 7394, 7404,  7414, 7423, 7432, 7442, 7451, 7461, 7471, 7480, 7489, 7499, 7508, 7518,  7528, 7537, 7546, 7556, 7566, 7575, 7584, 7594, 7603, 7613, 7623, 7632,  7641, 7651, 7660, 7670, 7680, 7689, 7698, 7708, 7717, 7727, 7737, 7746,  7755, 7765, 7774, 7784, 7794, 7803, 7812, 7822, 7831, 7841, 7851, 7860,  7869, 7879, 7888, 7898, 7908, 7917, 7926, 7936, 7946, 7955, 7964, 7974,  7983, 7993, 8003, 8012, 8021, 8031, 8040, 8050, 8060, 8069, 8078, 8088,  8097, 8107, 8117, 8126, 8135, 8145, 8154, 8164, 8174, 8183, 8192, 8202,  8211, 8221, 8231, 8240, 8249, 8259, 8268, 8278, 8288, 8297, 8306, 8316,  8325, 8335, 8345, 8354, 8363, 8373, 8382, 8392, 8402, 8411, 8420, 8430,  8439, 8449, 8459, 8468, 8477, 8487, 8496, 8506, 8516, 8525, 8534, 8544,  8553, 8563, 8573, 8582, 8591, 8601, 8611, 8620, 8629, 8639, 8648, 8658,  8668, 8677, 8686, 8696, 8705, 8715, 8725, 8734, 8743, 8753, 8762, 8772,  8782, 8791, 8800, 8810, 8819, 8829, 8839, 8848, 8857, 8867, 8876, 8886,  8896, 8905, 8914, 8924, 8933, 8943, 8953, 8962, 8971, 8981, 8991, 9000,  9009, 9019, 9028, 9038, 9048, 9057, 9066, 9076, 9085, 9095, 9105, 9114,  9123, 9133, 9142, 9152, 9162, 9171, 9180, 9190, 9199, 9209, 9219, 9228,  9237, 9247, 9256, 9266, 9276, 9285, 9294, 9304, 9313, 9323, 9333, 9342,  9351, 9361, 9371, 9380, 9389, 9399, 9408, 9418, 9428, 9437, 9446, 9456,  9465, 9475, 9485, 9494, 9503, 9513, 9522, 9532, 9542, 9551, 9560, 9570,  9579, 9589, 9599, 9608, 9617, 9627, 9636, 9646, 9656, 9665, 9674, 9684,  9693, 9703, 9713, 9722  };  int bsearch\_vav(const int\* a, int n, int x) {  int i = 0, j = n - 1;  while (i <= j) {  int k = i + ((j - i) / 2);  if (a[k] == x) {  return k;  }  else if (a[k] < x) {  i = k + 1;  }  else {  j = k - 1;  }  }  return -1;  }  int bsearch\_r\_vav(const int\* a, int x, int i, int j) {  if (j < i) {  return -1;  }  int k = i + ((j - i) / 2);  if (a[k] == x) {  return k;  }  else if (a[k] < x) {  return bsearch\_r\_vav(a, x, k + 1, j);  }  else {  return bsearch\_r\_vav(a, x, i, k - 1);  }  }  void logWithTime(FILE\* logFile, const char\* message) {  time\_t now = time(NULL);  struct tm\* tm\_info = localtime(&now);  char timeStr[20];  strftime(timeStr, sizeof(timeStr), "%Y-%m-%d %H:%M:%S", tm\_info);  fprintf(logFile, "[%s] %s\n", timeStr, message);  }  BOOL APIENTRY DllMain(HMODULE hModule, DWORD ul\_reason\_for\_call, LPVOID lpReserved) {  FILE\* logFile;  logFile = fopen("D:\\log\_3lab.txt", "a");  if (!logFile) {  printf("Error opening log file: %d\n", errno);  return FALSE;  }  switch (ul\_reason\_for\_call) {  case DLL\_PROCESS\_ATTACH:  logWithTime(logFile, "The DLL is projected onto the address space\n");  break;  case DLL\_THREAD\_ATTACH:  logWithTime(logFile, "A stream is being created\n");  break;  case DLL\_THREAD\_DETACH:  logWithTime(logFile, "The stream is ending correctly\n");  break;  case DLL\_PROCESS\_DETACH:  logWithTime(logFile, "The DLL is disconnected from the address space of the process\n");  break;  }  fclose(logFile);  return TRUE;  } |

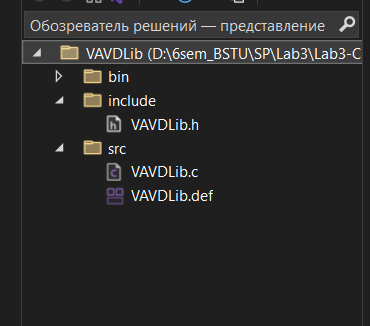
**VAVDLib.h**

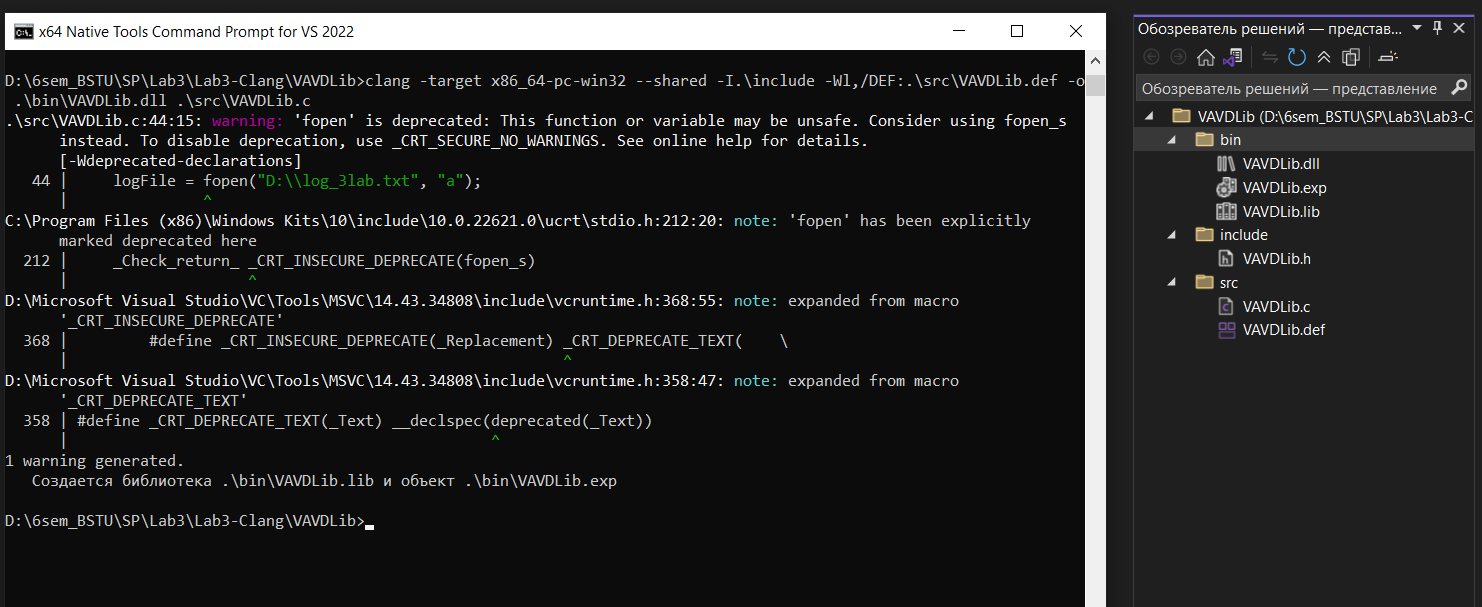
|  |
| --- |
| #ifndef MYLIBAPI  #define MYLIBAPI  extern const int array[1024];  int bsearch\_vav(const int\* a, int n, int x);  int bsearch\_r\_vav(const int\* a, int x, int i, int j);  #endif |

**VAVDLib.def**

|  |
| --- |
| LIBRARY VAVDLib  EXPORTS  iterative\_binary\_search = bsearch\_vav @1  bsearch\_r\_vav @2  array @3 |

**Clang**

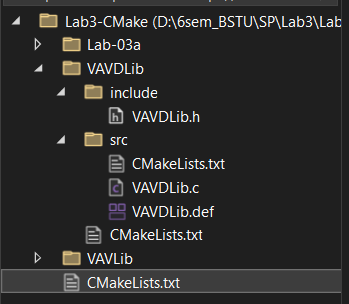


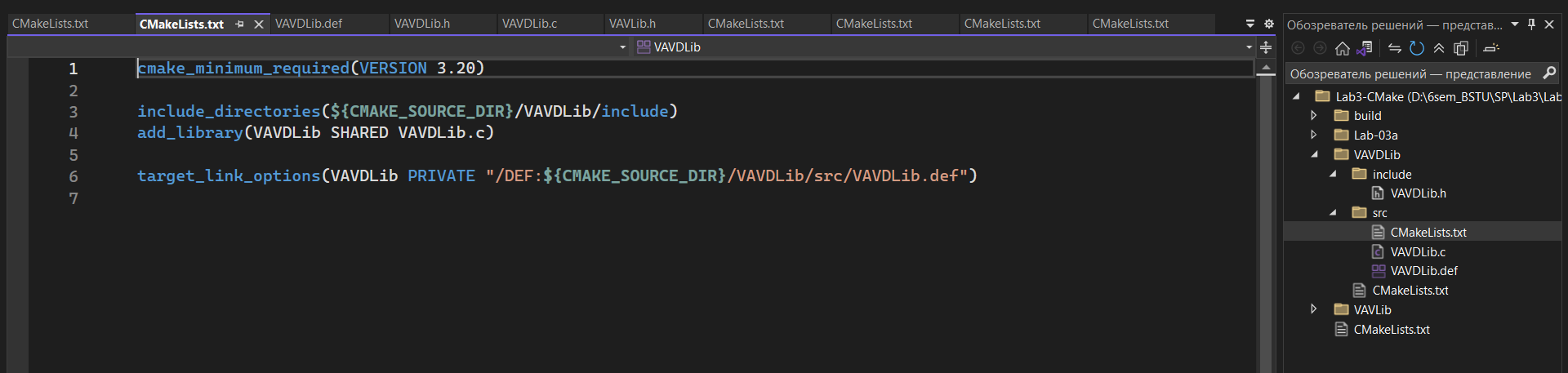


Флаг --shared указывает компоновщику, что нужно создать разделяемую библиотеку, а не исполняемый файл (executable).

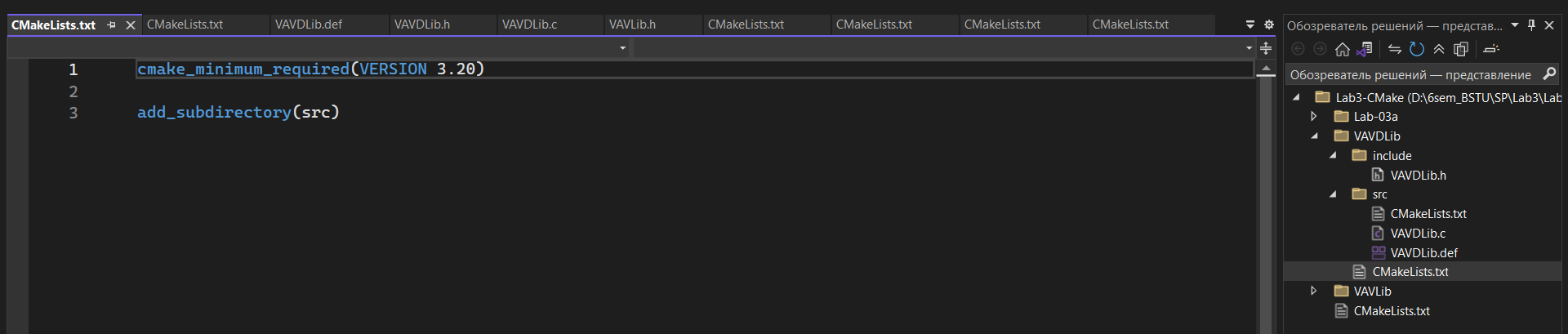
Опция -Wl,/DEF является способом передать специфичную для компоновщика опцию, которая указывает ему использовать файл определения модуля (.def) при создании DLL.

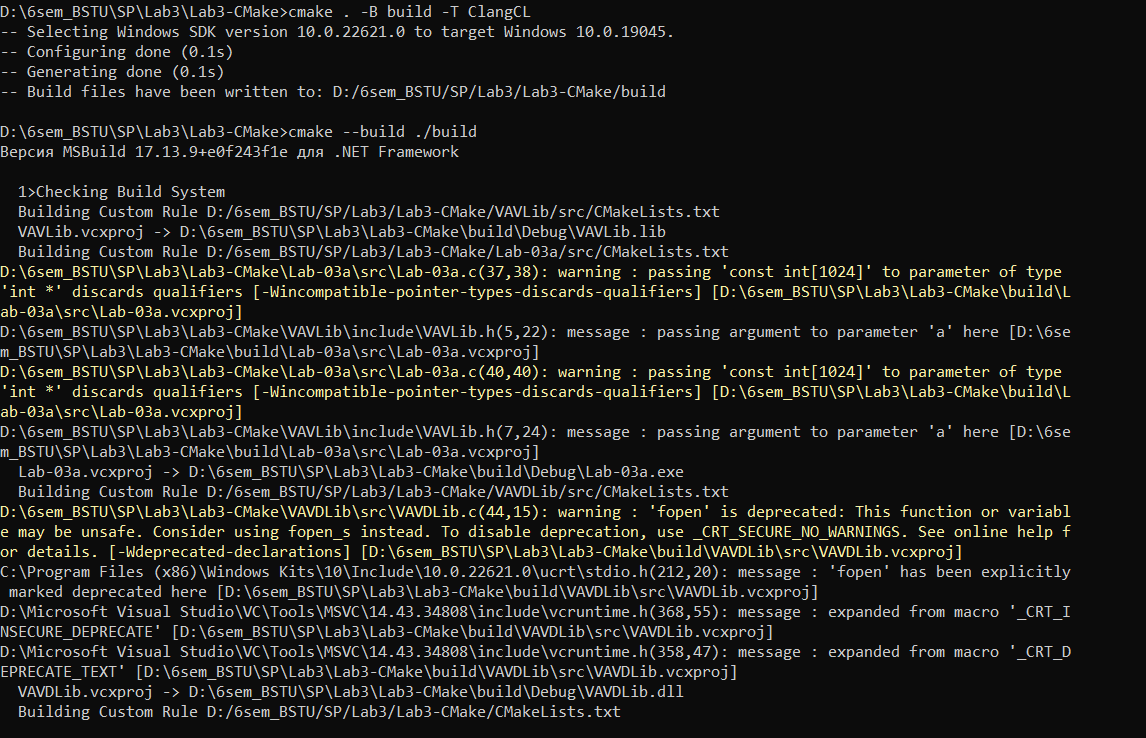
**CMake**

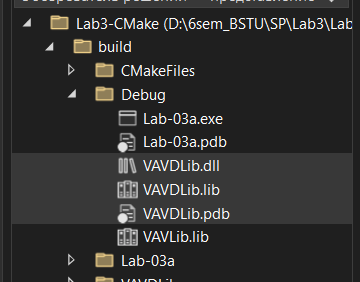




Команда target\_link\_options в файле CMakeLists.txt используется для добавления опций компоновщика (linker options) к конкретной цели (target), такой как исполняемый файл или библиотека.







**Проект <Ваши Инициалы>DLib++:**

Разработать динамическую библиотеку на языке **C++** с заданным функционалом (см. выше). Также должна быть реализована функция **DllMain** c логированием событий в некоторый файл (путь к файлу должен быть абсолютным).

Экспорт функций и переменных организовать через **\_\_declspec(dllexport)**. Одной из функций указать модификатор **extern “C”**.

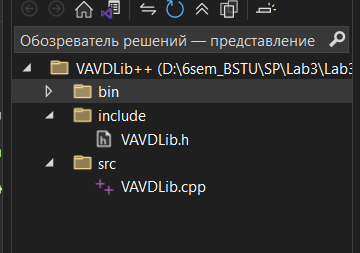
**VAVDLib++.h**

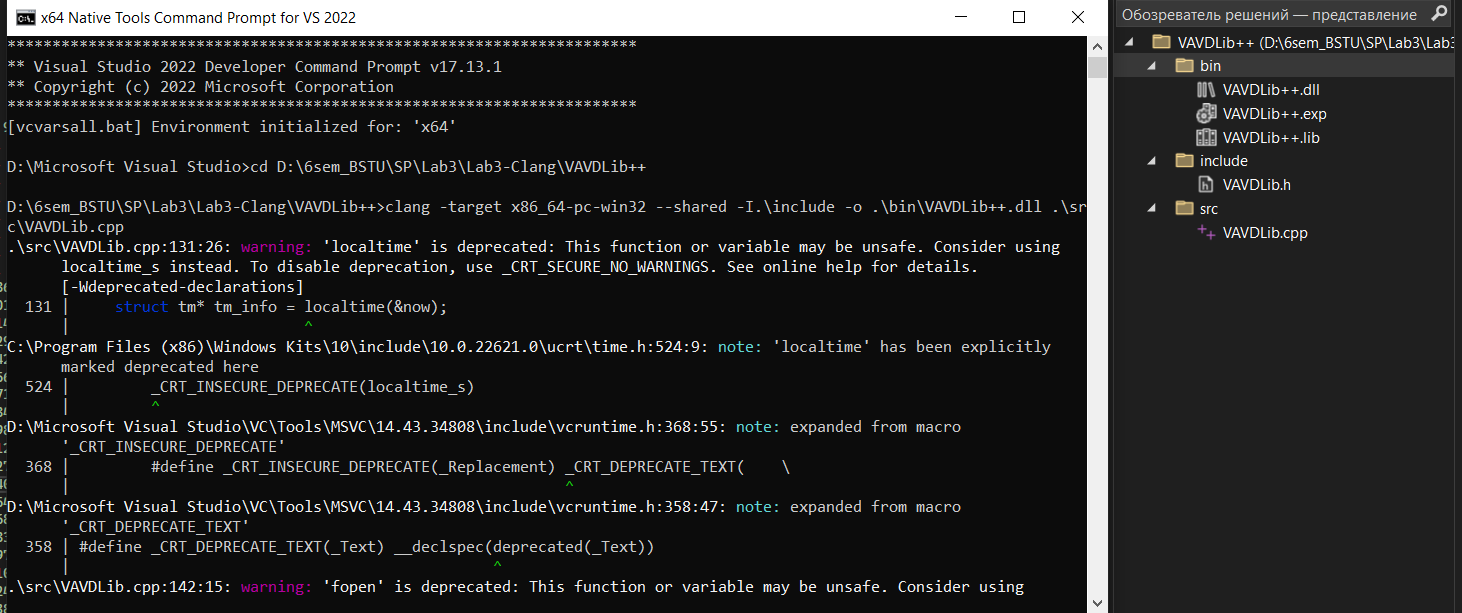
|  |
| --- |
| #ifndef MYLIBAPI  #define MYLIBAPI \_\_declspec(dllexport)  #endif  extern MYLIBAPI const int array[1024];  extern "C" MYLIBAPI int bsearch\_vav(const int\* a, int n, int x);  extern MYLIBAPI int bsearch\_r\_vav(const int\* a, int x, int i, int j); |

**VAVDLib++.cpp:**

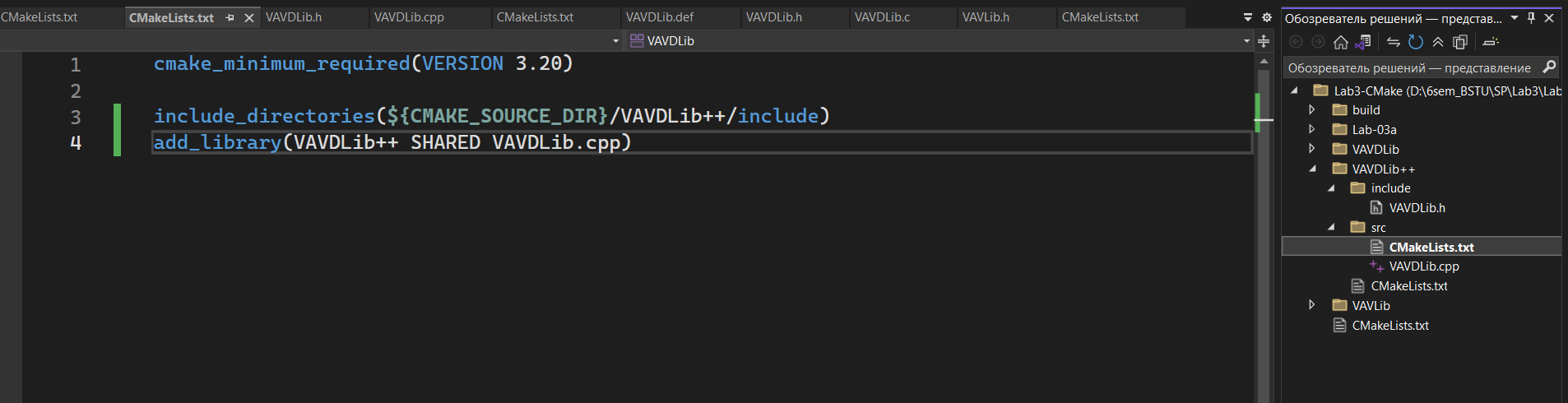
|  |
| --- |
| #include <windows.h>  #include <stdio.h>  #include <time.h>  #include <errno.h>  #include "VAVDLib.h"  const int array[1024] = {  4, 13, 23, 32, 41, 51, 61, 70, 79, 89, 98, 108,  118, 127, 136, 146, 156, 165, 174, 184, 194, 203, 212, 222,  232, 241, 251, 260, 269, 279, 288, 298, 307, 317, 327, 336,  345, 355, 364, 374, 384, 393, 402, 412, 422, 431, 440, 450,  460, 469, 478, 488, 498, 507, 517, 526, 535, 545, 554, 564,  574, 583, 592, 602, 611, 621, 630, 640, 650, 659, 668, 678,  688, 697, 706, 716, 725, 735, 745, 754, 763, 773, 782, 792,  802, 811, 820, 830, 840, 849, 858, 868, 878, 887, 896, 906,  915, 925, 935, 944, 953, 963, 972, 982, 992, 1001, 1010, 1020,  1030, 1039, 1049, 1058, 1067, 1077, 1086, 1096, 1106, 1115, 1124, 1134,  1143, 1153, 1163, 1172, 1181, 1191, 1201, 1210, 1219, 1229, 1239, 1248,  1257, 1267, 1277, 1286, 1295, 1305, 1314, 1324, 1334, 1343, 1352, 1362,  1371, 1381, 1391, 1400, 1409, 1419, 1429, 1438, 1447, 1457, 1466, 1476,  1486, 1495, 1504, 1514, 1524, 1533, 1542, 1552, 1561, 1571, 1581, 1590,  1599, 1609, 1619, 1628, 1637, 1647, 1656, 1666, 1676, 1685, 1694, 1704,  1713, 1723, 1733, 1742, 1751, 1761, 1771, 1780, 1789, 1799, 1808, 1818,  1828, 1837, 1846, 1856, 1866, 1875, 1884, 1894, 1903, 1913, 1923, 1932,  1941, 1951, 1961, 1970, 1979, 1989, 1998, 2008, 2018, 2027, 2036, 2046,  2055, 2065, 2075, 2084, 2093, 2103, 2112, 2122, 2132, 2141, 2150, 2160,  2170, 2179, 2188, 2198, 2207, 2217, 2227, 2236, 2245, 2255, 2264, 2274,  2284, 2293, 2302, 2312, 2322, 2331, 2340, 2350, 2359, 2369, 2379, 2388,  2397, 2407, 2417, 2426, 2435, 2445, 2454, 2464, 2474, 2483, 2492, 2502,  2511, 2521, 2531, 2540, 2549, 2559, 2568, 2578, 2588, 2597, 2606, 2616,  2625, 2635, 2645, 2654, 2663, 2673, 2683, 2692, 2701, 2711, 2720, 2730,  2740, 2749, 2758, 2768, 2777, 2787, 2797, 2806, 2815, 2825, 2834, 2844,  2854, 2863, 2872, 2882, 2892, 2901, 2910, 2920, 2929, 2939, 2949, 2958,  2967, 2977, 2987, 2996, 3005, 3015, 3024, 3034, 3044, 3053, 3062, 3072,  3081, 3091, 3101, 3110, 3119, 3129, 3138, 3148, 3158, 3167, 3176, 3186,  3195, 3205, 3215, 3224, 3233, 3243, 3253, 3262, 3271, 3281, 3290, 3300,  3310, 3319, 3328, 3338, 3347, 3357, 3367, 3376, 3385, 3395, 3404, 3414,  3424, 3433, 3442, 3452, 3461, 3471, 3481, 3490, 3499, 3509, 3518, 3528,  3538, 3547, 3556, 3566, 3576, 3585, 3594, 3604, 3613, 3623, 3633, 3642,  3651, 3661, 3670, 3680, 3690, 3699, 3708, 3718, 3727, 3737, 3747, 3756,  3765, 3775, 3784, 3794, 3804, 3813, 3822, 3832, 3841, 3851, 3861, 3870,  3879, 3889, 3898, 3908, 3918, 3927, 3936, 3946, 3955, 3965, 3975, 3984,  3993, 4003, 4013, 4022, 4031, 4041, 4050, 4060, 4070, 4079, 4088, 4098,  4107, 4117, 4127, 4136, 4145, 4155, 4164, 4174, 4184, 4193, 4202, 4212,  4221, 4231, 4241, 4250, 4259, 4269, 4279, 4288, 4297, 4307, 4316, 4326,  4336, 4345, 4354, 4364, 4373, 4383, 4393, 4402, 4411, 4421, 4430, 4440,  4450, 4459, 4468, 4478, 4487, 4497, 4507, 4516, 4525, 4535, 4544, 4554,  4564, 4573, 4582, 4592, 4602, 4611, 4620, 4630, 4639, 4649, 4659, 4668,  4677, 4687, 4696, 4706, 4716, 4725, 4734, 4744, 4753, 4763, 4773, 4782,  4791, 4801, 4811, 4820, 4829, 4839, 4848, 4858, 4868, 4877, 4886, 4896,  4905, 4915, 4925, 4934, 4943, 4953, 4963, 4972, 4981, 4991, 5000, 5010,  5020, 5029, 5038, 5048, 5057, 5067, 5077, 5086, 5095, 5105, 5114, 5124,  5134, 5143, 5152, 5162, 5171, 5181, 5191, 5200, 5209, 5219, 5228, 5238,  5248, 5257, 5266, 5276, 5286, 5295, 5304, 5314, 5323, 5333, 5343, 5352,  5361, 5371, 5380, 5390, 5400, 5409, 5418, 5428, 5437, 5447, 5457, 5466,  5475, 5485, 5494, 5504, 5514, 5523, 5532, 5542, 5551, 5561, 5571, 5580,  5589, 5599, 5608, 5618, 5628, 5637, 5646, 5656, 5666, 5675, 5684, 5694,  5703, 5713, 5723, 5732, 5741, 5751, 5760, 5770, 5780, 5789, 5798, 5808,  5817, 5827, 5837, 5846, 5855, 5865, 5874, 5884, 5894, 5903, 5912, 5922,  5931, 5941, 5951, 5960, 5969, 5979, 5989, 5998, 6007, 6017, 6026, 6036,  6046, 6055, 6064, 6074, 6083, 6093, 6103, 6112, 6121, 6131, 6140, 6150,  6160, 6169, 6178, 6188, 6197, 6207, 6217, 6226, 6235, 6245, 6254, 6264,  6274, 6283, 6292, 6302, 6311, 6321, 6331, 6340, 6349, 6359, 6368, 6378,  6388, 6397, 6406, 6416, 6426, 6435, 6444, 6454, 6463, 6473, 6483, 6492,  6501, 6511, 6520, 6530, 6540, 6549, 6558, 6568, 6577, 6587, 6597, 6606,  6615, 6625, 6634, 6644, 6654, 6663, 6672, 6682, 6691, 6701, 6711, 6720,  6729, 6739, 6749, 6758, 6767, 6777, 6786, 6796, 6806, 6815, 6824, 6834,  6843, 6853, 6863, 6872, 6881, 6891, 6901, 6910, 6919, 6929, 6938, 6948,  6958, 6967, 6976, 6986, 6995, 7005, 7015, 7024, 7033, 7043, 7052, 7062,  7072, 7081, 7090, 7100, 7109, 7119, 7129, 7138, 7147, 7157, 7166, 7176,  7186, 7195, 7204, 7214, 7223, 7233, 7243, 7252, 7261, 7271, 7280, 7290,  7300, 7309, 7318, 7328, 7337, 7347, 7357, 7366, 7375, 7385, 7394, 7404,  7414, 7423, 7432, 7442, 7451, 7461, 7471, 7480, 7489, 7499, 7508, 7518,  7528, 7537, 7546, 7556, 7566, 7575, 7584, 7594, 7603, 7613, 7623, 7632,  7641, 7651, 7660, 7670, 7680, 7689, 7698, 7708, 7717, 7727, 7737, 7746,  7755, 7765, 7774, 7784, 7794, 7803, 7812, 7822, 7831, 7841, 7851, 7860,  7869, 7879, 7888, 7898, 7908, 7917, 7926, 7936, 7946, 7955, 7964, 7974,  7983, 7993, 8003, 8012, 8021, 8031, 8040, 8050, 8060, 8069, 8078, 8088,  8097, 8107, 8117, 8126, 8135, 8145, 8154, 8164, 8174, 8183, 8192, 8202,  8211, 8221, 8231, 8240, 8249, 8259, 8268, 8278, 8288, 8297, 8306, 8316,  8325, 8335, 8345, 8354, 8363, 8373, 8382, 8392, 8402, 8411, 8420, 8430,  8439, 8449, 8459, 8468, 8477, 8487, 8496, 8506, 8516, 8525, 8534, 8544,  8553, 8563, 8573, 8582, 8591, 8601, 8611, 8620, 8629, 8639, 8648, 8658,  8668, 8677, 8686, 8696, 8705, 8715, 8725, 8734, 8743, 8753, 8762, 8772,  8782, 8791, 8800, 8810, 8819, 8829, 8839, 8848, 8857, 8867, 8876, 8886,  8896, 8905, 8914, 8924, 8933, 8943, 8953, 8962, 8971, 8981, 8991, 9000,  9009, 9019, 9028, 9038, 9048, 9057, 9066, 9076, 9085, 9095, 9105, 9114,  9123, 9133, 9142, 9152, 9162, 9171, 9180, 9190, 9199, 9209, 9219, 9228,  9237, 9247, 9256, 9266, 9276, 9285, 9294, 9304, 9313, 9323, 9333, 9342,  9351, 9361, 9371, 9380, 9389, 9399, 9408, 9418, 9428, 9437, 9446, 9456,  9465, 9475, 9485, 9494, 9503, 9513, 9522, 9532, 9542, 9551, 9560, 9570,  9579, 9589, 9599, 9608, 9617, 9627, 9636, 9646, 9656, 9665, 9674, 9684,  9693, 9703, 9713, 9722  };  int bsearch\_vav(const int\* a, int n, int x) {  int i = 0, j = n - 1;  while (i <= j) {  int k = i + ((j - i) / 2);  if (a[k] == x) {  return k;  }  else if (a[k] < x) {  i = k + 1;  }  else {  j = k - 1;  }  }  return -1;  }  int bsearch\_r\_vav(const int\* a, int x, int i, int j) {  if (j < i) {  return -1;  }  int k = i + ((j - i) / 2);  if (a[k] == x) {  return k;  }  else if (a[k] < x) {  return bsearch\_r\_vav(a, x, k + 1, j);  }  else {  return bsearch\_r\_vav(a, x, i, k - 1);  }  }  void logWithTime(FILE\* logFile, const char\* message) {  time\_t now = time(NULL);  struct tm\* tm\_info = localtime(&now);  char timeStr[20];  strftime(timeStr, sizeof(timeStr), "%Y-%m-%d %H:%M:%S", tm\_info);  fprintf(logFile, "[%s] %s\n", timeStr, message);  }  BOOL APIENTRY DllMain(HMODULE hModule, DWORD ul\_reason\_for\_call, LPVOID lpReserved) {  FILE\* logFile;  logFile = fopen("D:\\log\_3lab.txt", "a");  if (!logFile) {  printf("Error opening log file: %d\n", errno);  return FALSE;  }  switch (ul\_reason\_for\_call) {  case DLL\_PROCESS\_ATTACH:  logWithTime(logFile, "The DLL is projected onto the address space\n");  break;  case DLL\_THREAD\_ATTACH:  logWithTime(logFile, "A stream is being created\n");  break;  case DLL\_THREAD\_DETACH:  logWithTime(logFile, "The stream is ending correctly\n");  break;  case DLL\_PROCESS\_DETACH:  logWithTime(logFile, "The DLL is disconnected from the address space of the process\n");  break;  }  fclose(logFile);  return TRUE;  } |

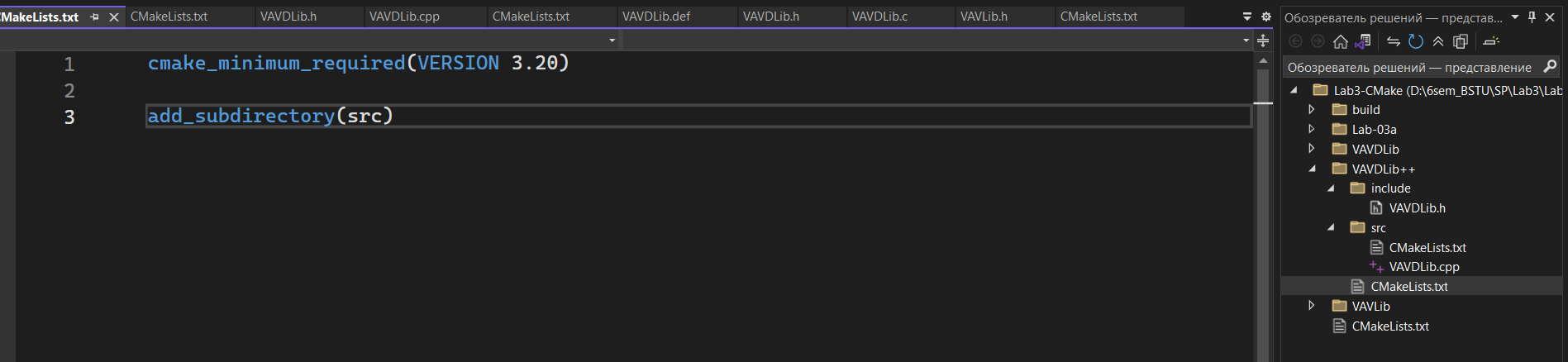
**Clang**

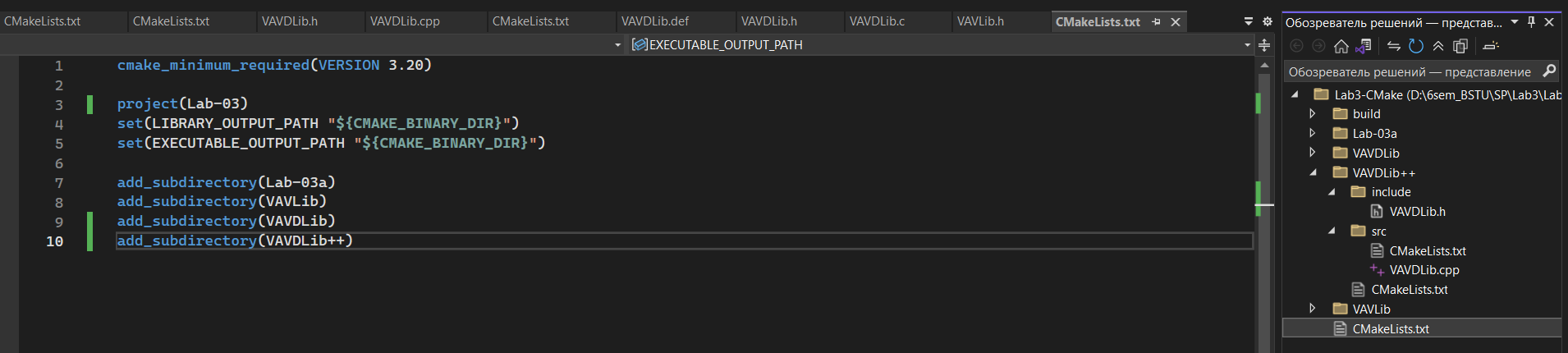
****

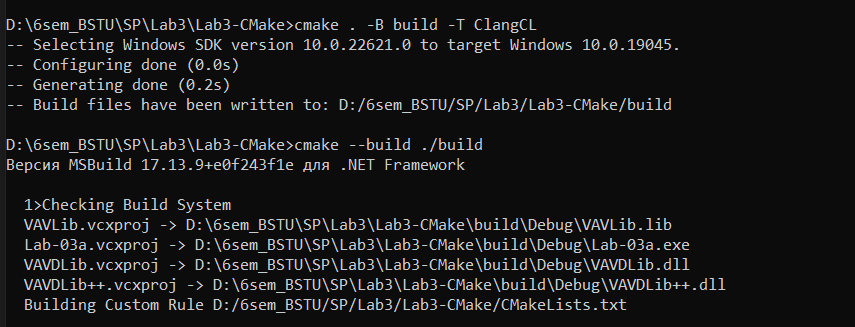
****

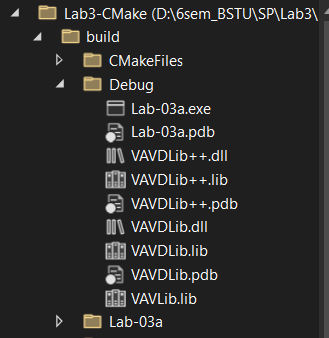
**CMake**

****

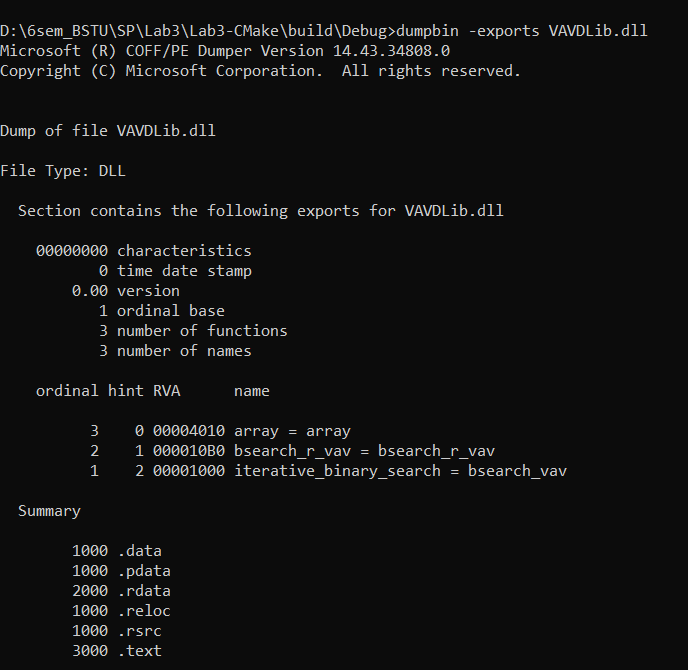
****

****

****

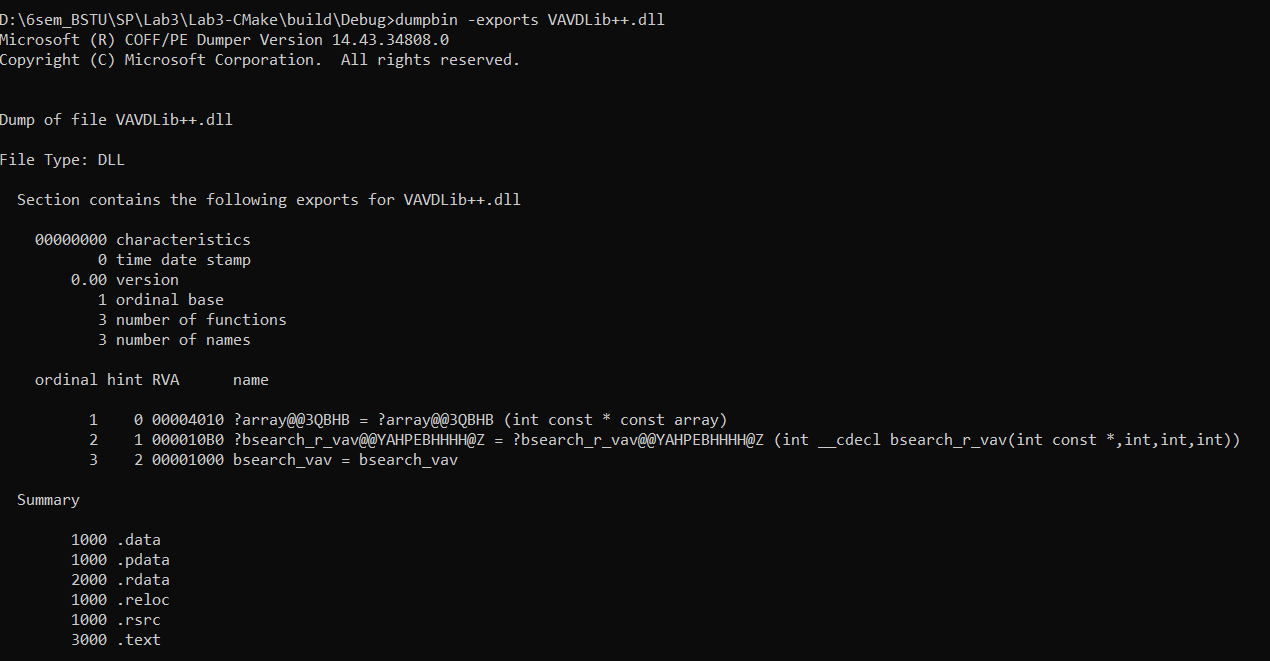
****

Изучить таблицы экспорта обеих библиотек с помощью утилиты **dumpbin** (данная утилита доступна из Developer версии терминала Visual Studio)и предоставить скриншоты с их содержимым! Имя функции преобразованное компилятором C++ сохраните для дальнейших вызовов этой функции!



Простые имена. Это **C-связывание (C linkage)**. Имена не искажены.

bsearch\_vav экспортируется под именем iterative\_binary\_search (ч рез .def файл).



?array@@3QBHB **искаженное (mangled) имя C++**. Компилятор C++ закодировал тип переменной (const int[]) и другую информацию в имя символа. Это C++ связывание.?bsearch\_r\_vav@@YAHPEBHHHH@Z аналогично.

bsearch\_vav – простое имя. Это **C-связывание**, достигнутое благодаря использованию extern "C" .

**Проект Lab-03b-Explicit:**

Разработать клиентское приложение которое сможет **явно** подключить любую из двух динамических библиотек. Приложение должно быть консольным и может быть вызывано в виде:

***Lab-03b-ex <Подключаемая библиотека> <Имя или номер вызываемой функции> [Искомое число]***

**Информация!** Параметры «Подключаемая библиотека» и «Имя вызываемой функции» должны быть непосредственно значениями передаваемыми в функции **LoadLibrary** и **GetProcAddress**.

Если искомое число не задано, то оно запрашивается для ввода с консоли (для наглядности работы динамической библиотеки поместите этот код непосредственно перед вызовом функции поиска). В результате выполнения этого приложения должен быть выведен один из следующих результатов:

* Не указана загружаемая библиотека!
* Не указана вызываемая функция!
* Загружаемая библиотека не найдена!
* Искомая функция <Имя вызываемой функции> не найдена в библиотеке <Имя загруженной библиотеки>!
* <Имя вызванной функции>: Заданное число не найдено!
* <Имя вызванной функции>: Число <Х> найдено на позиции <Y>!

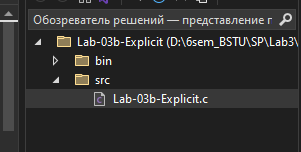
Изучить таблицу импорта приложения с помощью утилиты **dumpbin** и предоставить скриншот с её содержимым!

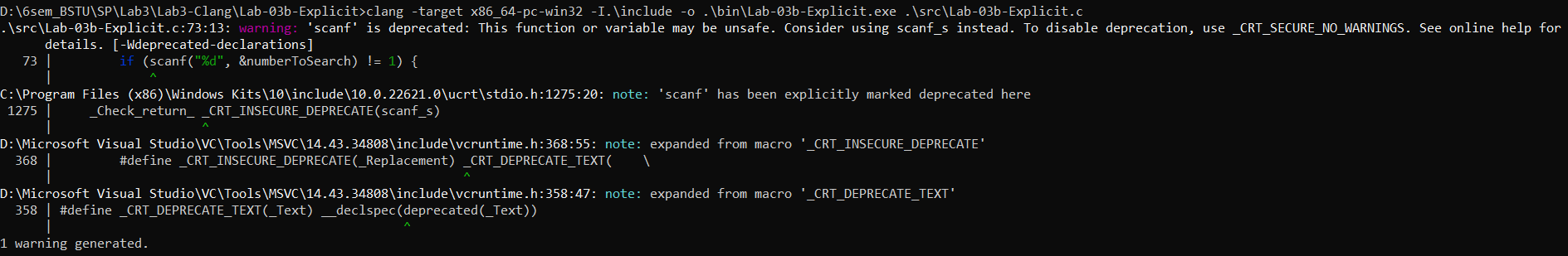
***Вопрос:*** *Что оказалось в таблице импорта? Почему?*

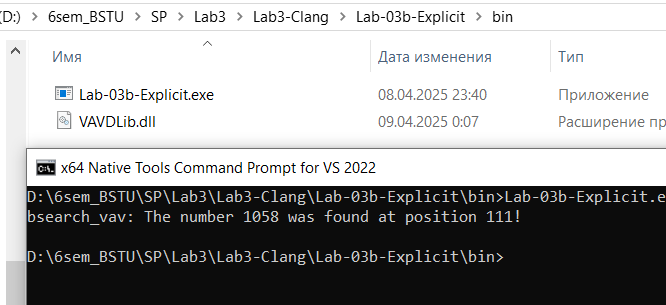
Проверить факт подключения библиотеки через Process Explorer. (Для этого не указывайте число в аргументах при запуске приложения и дайте приложению войти в режим ожидания ввода с консоли).

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <windows.h>  typedef int (\*IterativeBSearchFunc)(const int\*, int, int);  typedef int (\*RecursiveBSearchFunc)(const int\*, int, int, int);  typedef const int\* LibArrayPtr;  int main(int argc, char\* argv[])  {  if (argc < 3)  {  if (argc < 2) {  printf("The library to download is not specified\n");  }  else {  printf("The function being called is not specified\n");  }  printf("Usage: %s <LibraryName.dll> <FunctionName|Number> [NumberToSearch]\n", argv[0]);  printf("Example Labs: VAVDLib.dll, VAVDLib++.dll\n");  printf("Example Funcs: 1. bsearch\_vav, 2. bsearch\_r\_vav\n");  return 1;  }  char\* libFileName = argv[1];  char\* funcIdentifier = argv[2];  HMODULE hLib = LoadLibrary(libFileName);  if (NULL == hLib)  {  printf("The downloaded library was not found (%s, Error: %lu)\n", libFileName, GetLastError());  return 1;  }  IterativeBSearchFunc funcIterative = NULL;  RecursiveBSearchFunc funcRecursive = NULL;  LibArrayPtr loadedArray = NULL;  if (0 == strcmp(libFileName, "VAVDLib.dll"))  {  funcIterative = (IterativeBSearchFunc)GetProcAddress(hLib, "iterative\_binary\_search");  funcRecursive = (RecursiveBSearchFunc)GetProcAddress(hLib, "bsearch\_r\_vav");  loadedArray = (LibArrayPtr)GetProcAddress(hLib, "array");  }  else if (0 == strcmp(libFileName, "VAVDLib++.dll"))  {  funcIterative = (IterativeBSearchFunc)GetProcAddress(hLib, "bsearch\_vav");  funcRecursive = (RecursiveBSearchFunc)GetProcAddress(hLib, "?bsearch\_r\_vav@@YAHPEBHHHH@Z");  loadedArray = (LibArrayPtr)GetProcAddress(hLib, "?array@@3QBHB");  }  if (funcIterative == NULL && funcRecursive == NULL)  {  printf("The required functions were not found in the library %s\n", libFileName);  FreeLibrary(hLib);  return 1;  }  if (loadedArray == NULL) {  printf("Could not load array from library %s\n", libFileName);  FreeLibrary(hLib);  return 1;  }  int numberToSearch;  if (argc >= 4) {  numberToSearch = atoi(argv[3]);  }  else {  printf("Enter the number: ");  if (scanf("%d", &numberToSearch) != 1) {  printf("Error\n");  FreeLibrary(hLib);  return 1;  }  }  int foundPosition = -1;  int isIterativeCall = (strcmp(funcIdentifier, "bsearch\_vav") == 0 || strcmp(funcIdentifier, "1") == 0);  int isRecursiveCall = (strcmp(funcIdentifier, "bsearch\_r\_vav") == 0 || strcmp(funcIdentifier, "2") == 0);  if (isIterativeCall)  {  if (funcIterative != NULL) {  foundPosition = funcIterative((const int\*)loadedArray, 1024, numberToSearch);  }  else {  printf("Iterative function pointer is invalid for %s!\n", libFileName);  FreeLibrary(hLib);  return 1;  }  }  else if (isRecursiveCall)  {  if (funcRecursive != NULL) {  foundPosition = funcRecursive((const int\*)loadedArray, numberToSearch, 0, 1023);  }  else {  printf("Recursive function pointer is invalid for %s!\n", libFileName);  FreeLibrary(hLib);  return 1;  }  }  else  {  printf("The required %s function is not supported!\n", funcIdentifier);  FreeLibrary(hLib);  return 1;  }  if (foundPosition != -1) {  printf("%s: The number %d was found at position %d!\n", funcIdentifier, numberToSearch, foundPosition);  }  else {  printf("%s: The specified number was not found!\n", funcIdentifier);  }  FreeLibrary(hLib);  return 0;  } |

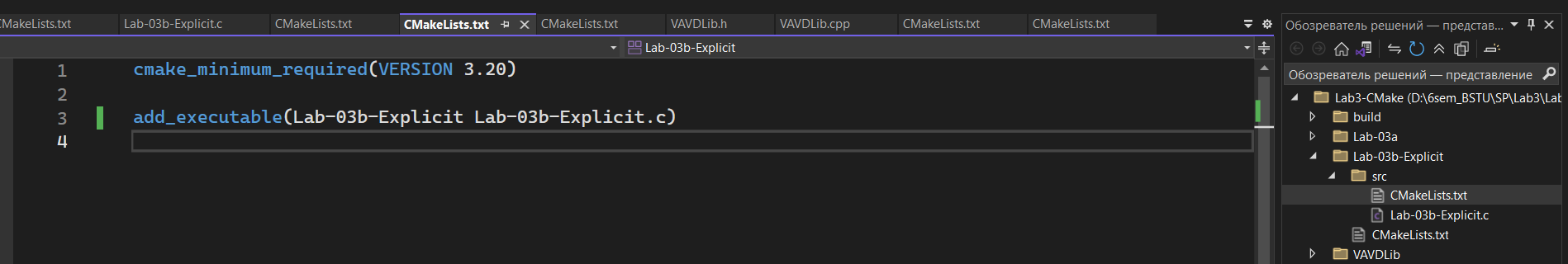
**Clang**

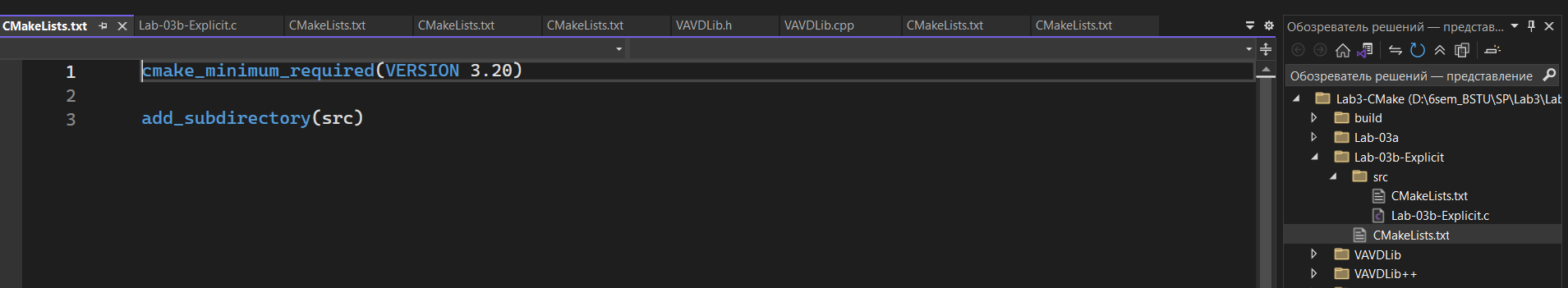
****

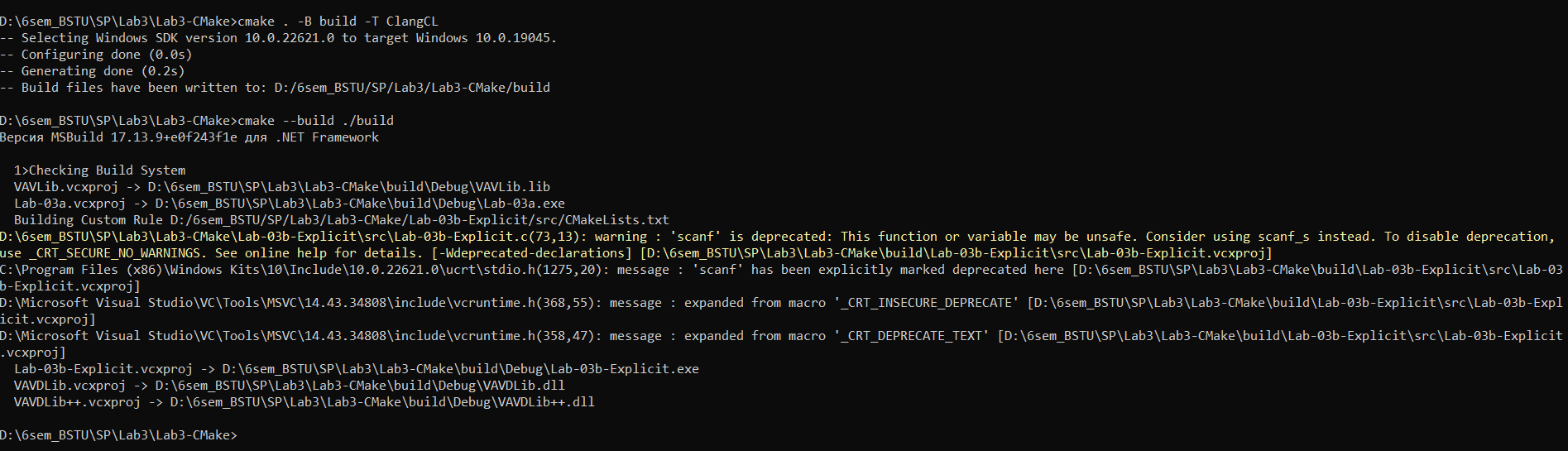
****

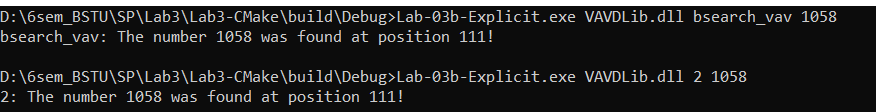
****

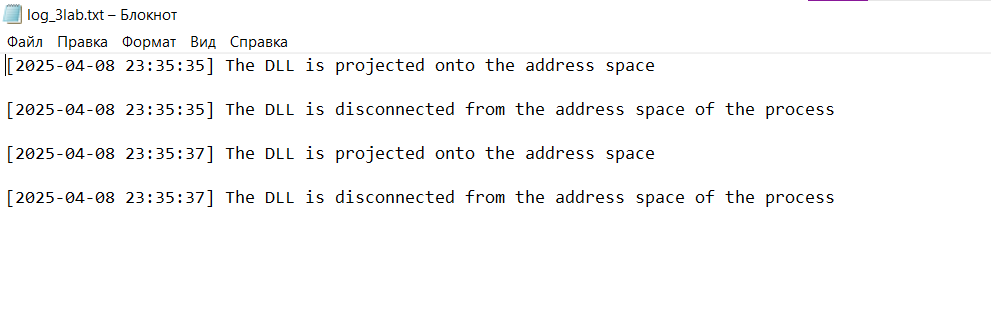
**CMake**

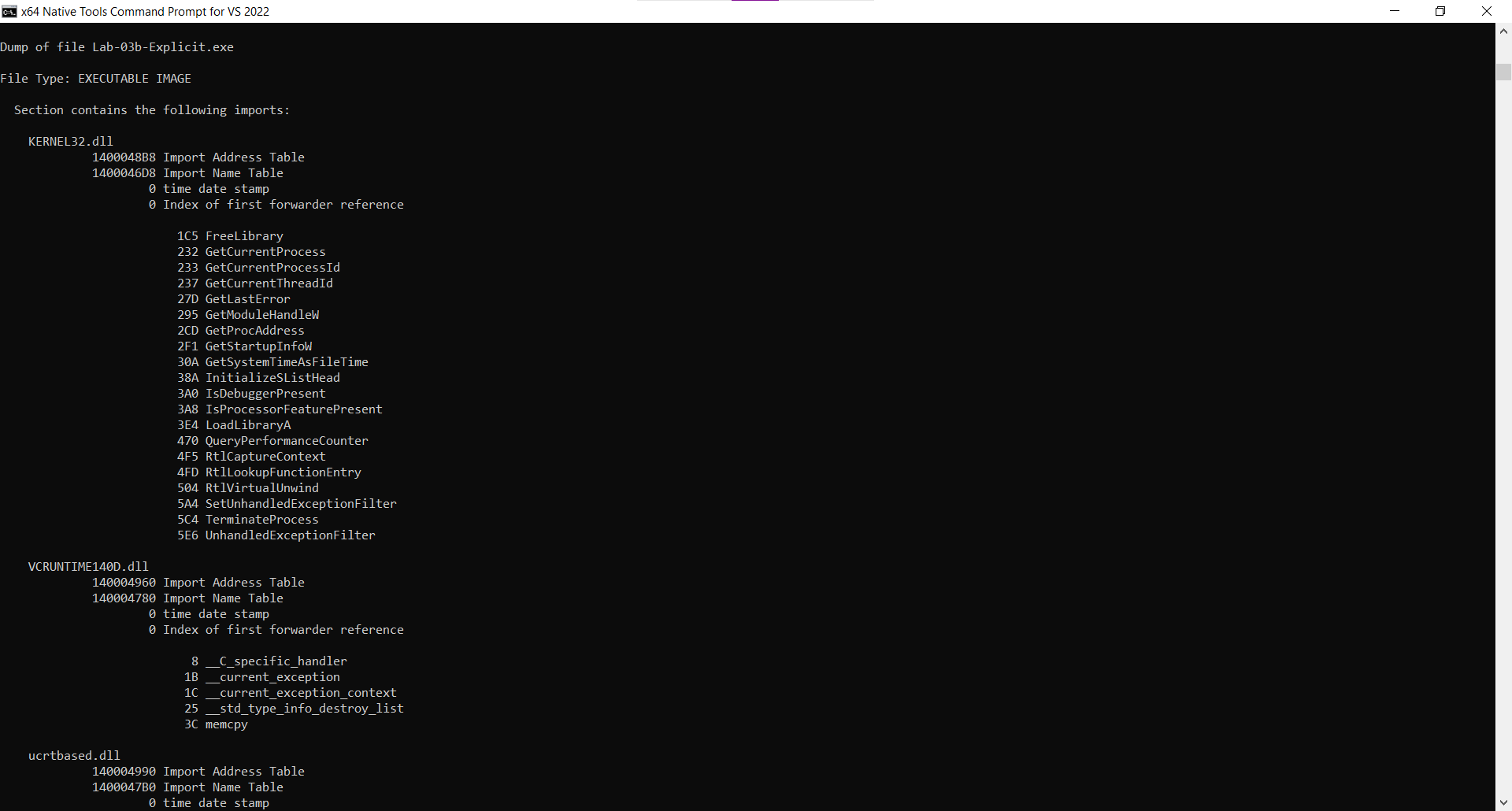
****

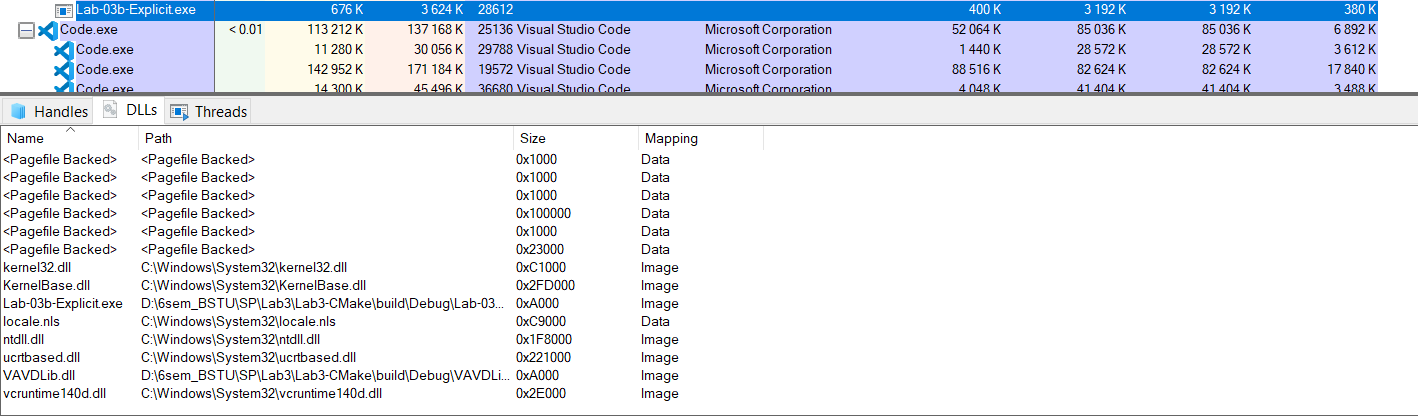
****

****

****

****

****

****

**Проект Lab-03b-Implicit:**

Разработать клиентское приложение которое неявно подключает одну любую из двух динамических библиотек. Приложение должно быть консольным и может быть вызывано в виде:

***Lab-03b-im <Имя вызываемой функции> [Искомое число]***

Если искомое число не задано, то оно запрашивается для ввода с консоли. В результате выполнения этого приложения должен быть выведен один из следующих результатов:

* Не указана вызываемая функция!
* <Имя вызванной функции>: Заданное число не найдено!
* <Имя вызванной функции>: Число <Х> найдено на позиции <Y>!

Изучить таблицу импорта приложения с помощью утилиты **dumpbin** и предоставить скриншот с её содержимым!

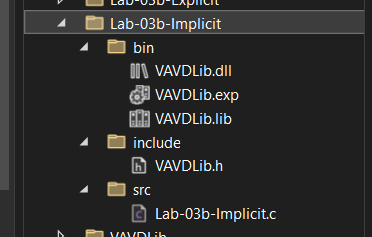
***Вопрос:*** *Что оказалось в таблице импорта? Почему?*

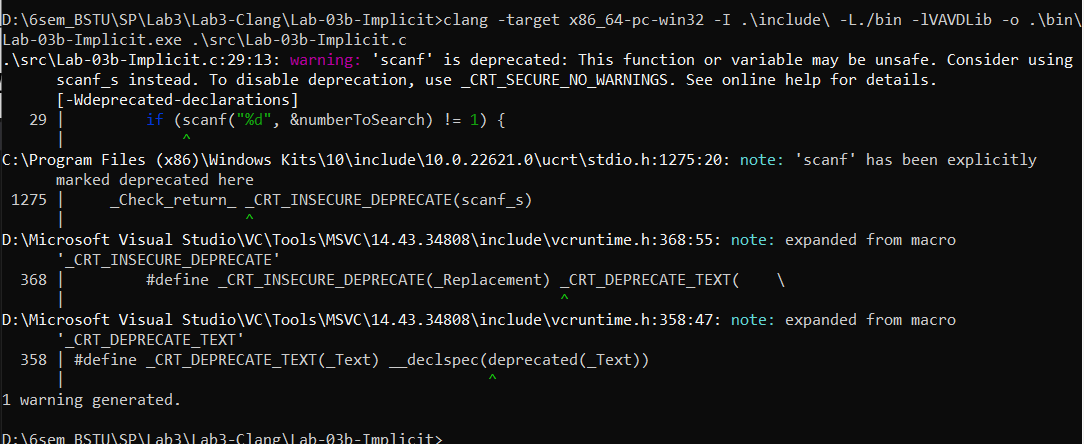
Проверить факт подключения библиотеки через Process Explorer. (Для этого не указывайте число в аргументах при запуске приложения и дайте приложению войти в режим ожидания ввода с консоли).

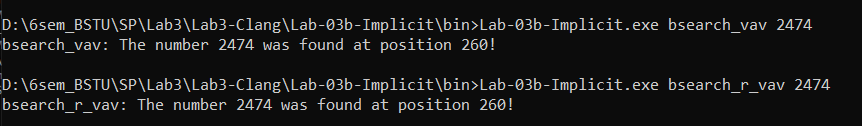
Lab-03b-Implicit.c:

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <windows.h>  \_\_declspec(dllimport) int iterative\_binary\_search(const int\*, int, int);  \_\_declspec(dllimport) int bsearch\_r\_vav(const int\*, int, int, int);  \_\_declspec(dllimport) const int array[1024];  int main(int argc, char\* argv[])  {  if (argc < 2)  {  printf("The function being called is not specified\n");  printf("Usage: %s <FunctionName|Number> [NumberToSearch]\n", argv[0]);  printf("Example Funcs: 1. bsearch\_vav, 2. bsearch\_r\_vav\n");  return 1;  }  char\* funcIdentifier = argv[1];  int numberToSearch;  if (argc >= 3) {  numberToSearch = atoi(argv[3]);  }  else {  printf("Enter the number: ");  if (scanf("%d", &numberToSearch) != 1) {  printf("Error\n");  return 1;  }  }  int foundPosition = -1;  int isIterativeCall = (strcmp(funcIdentifier, "bsearch\_vav") == 0 || strcmp(funcIdentifier, "1") == 0);  int isRecursiveCall = (strcmp(funcIdentifier, "bsearch\_r\_vav") == 0 || strcmp(funcIdentifier, "2") == 0);  if (isIterativeCall) foundPosition = iterative\_binary\_search(array, 1024, numberToSearch);  else if (isRecursiveCall) foundPosition = bsearch\_r\_vav(array, numberToSearch, 0, 1023);  else  {  printf("The required %s function is not supported!\n", funcIdentifier);  return 1;  }  if (foundPosition != -1) {  printf("%s: The number %d was found at position %d!\n", funcIdentifier, numberToSearch, foundPosition);  }  else {  printf("%s: The specified number was not found!\n", funcIdentifier);  }  return 0;  } |

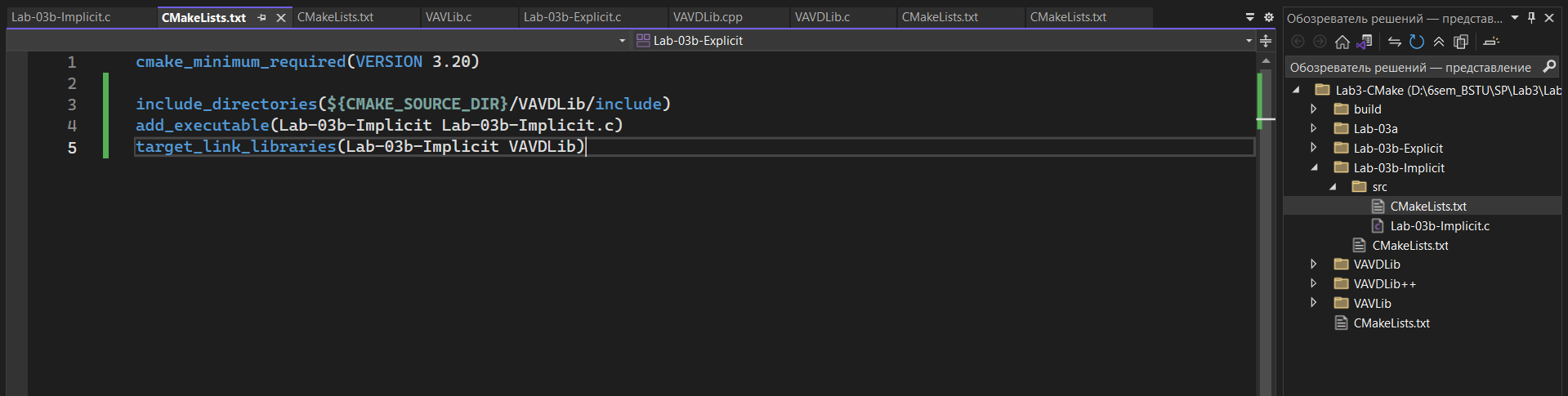
**Clang**

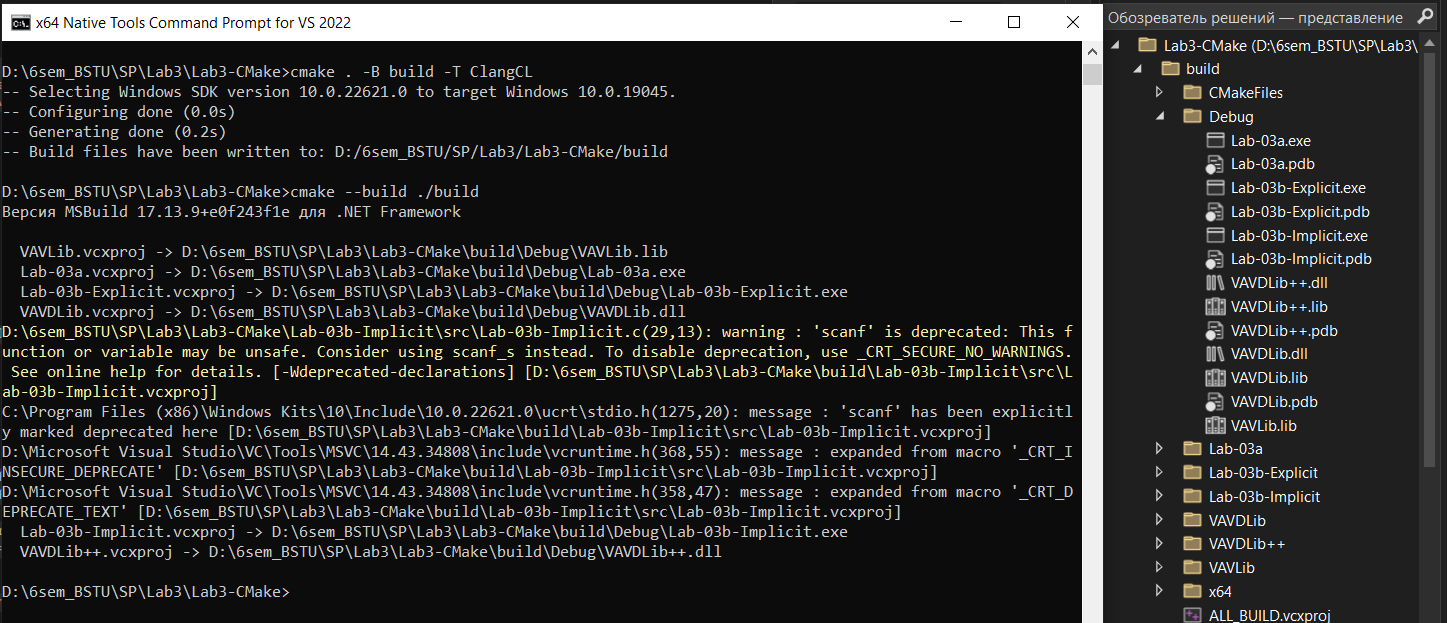
****

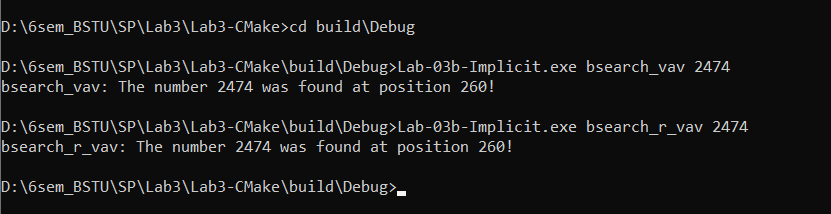
****

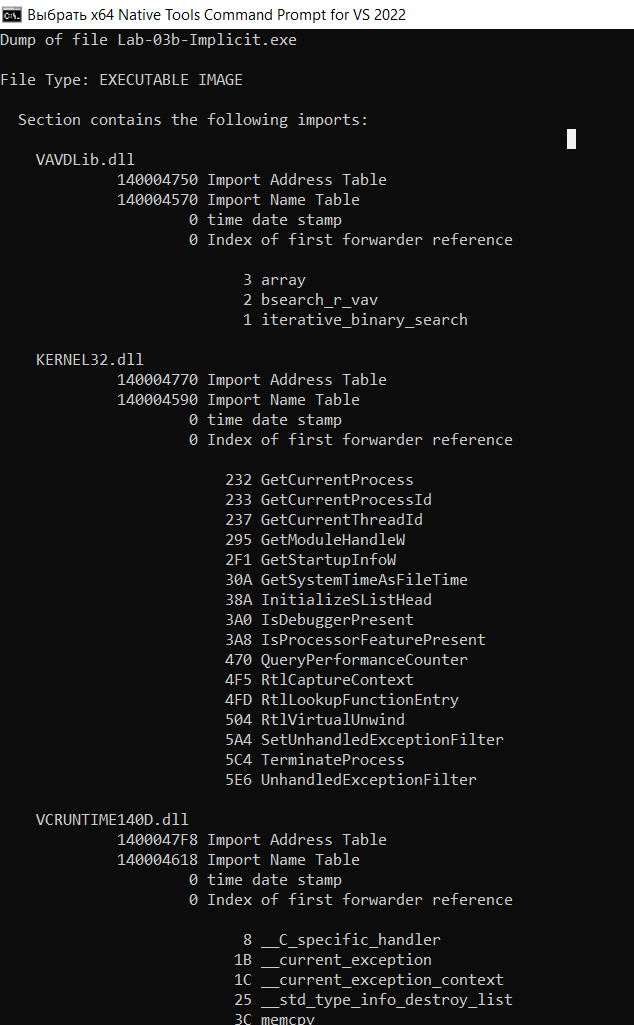
****

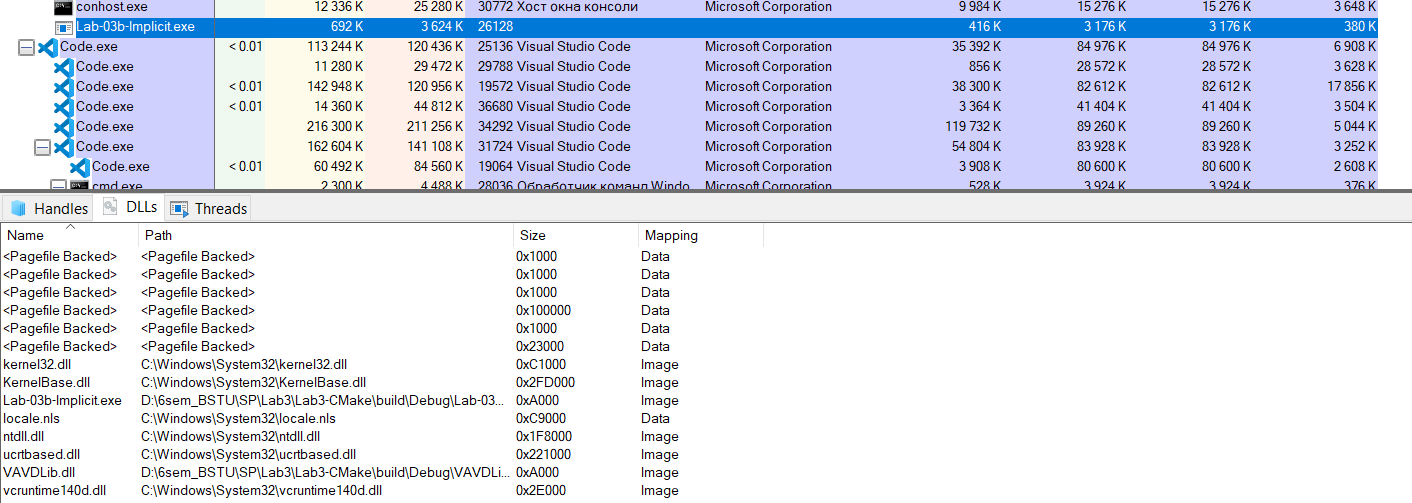
**CMake**

****

****

****

****

****

**Главный CMakeLists.txt:**

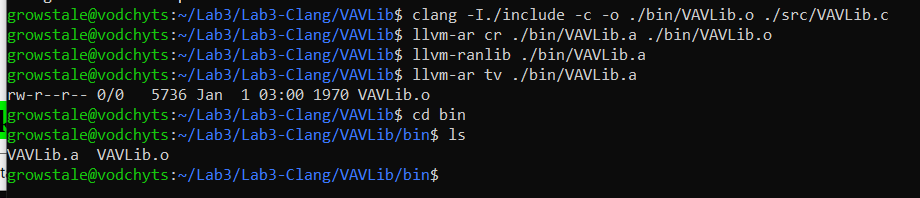
|  |
| --- |
| cmake\_minimum\_required(VERSION 3.20)  project(Lab-03)  set(LIBRARY\_OUTPUT\_PATH "${CMAKE\_BINARY\_DIR}")  set(EXECUTABLE\_OUTPUT\_PATH "${CMAKE\_BINARY\_DIR}")  add\_subdirectory(Lab-03a)  add\_subdirectory(VAVLib)  add\_subdirectory(VAVDLib)  add\_subdirectory(VAVDLib++)  add\_subdirectory(Lab-03b-Explicit)  add\_subdirectory(Lab-03b-Implicit) |

**Постановка задачи для Linux:**

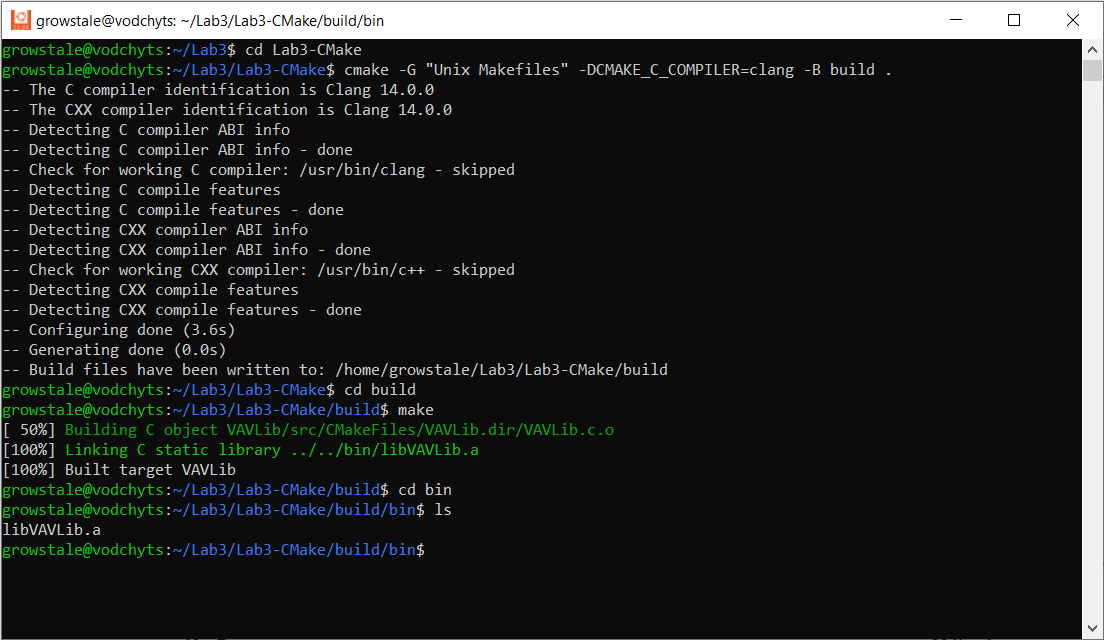
**Проект <Ваши Инициалы>Lib:**

Собрать статическую библиотеку из проекта с аналогичным названием на Windows.

**Clang**

****

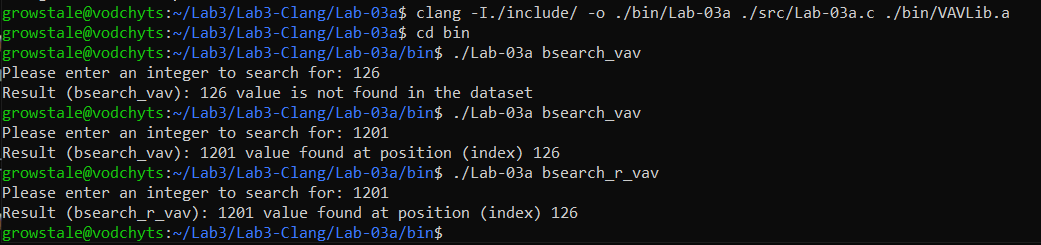
**CMake**



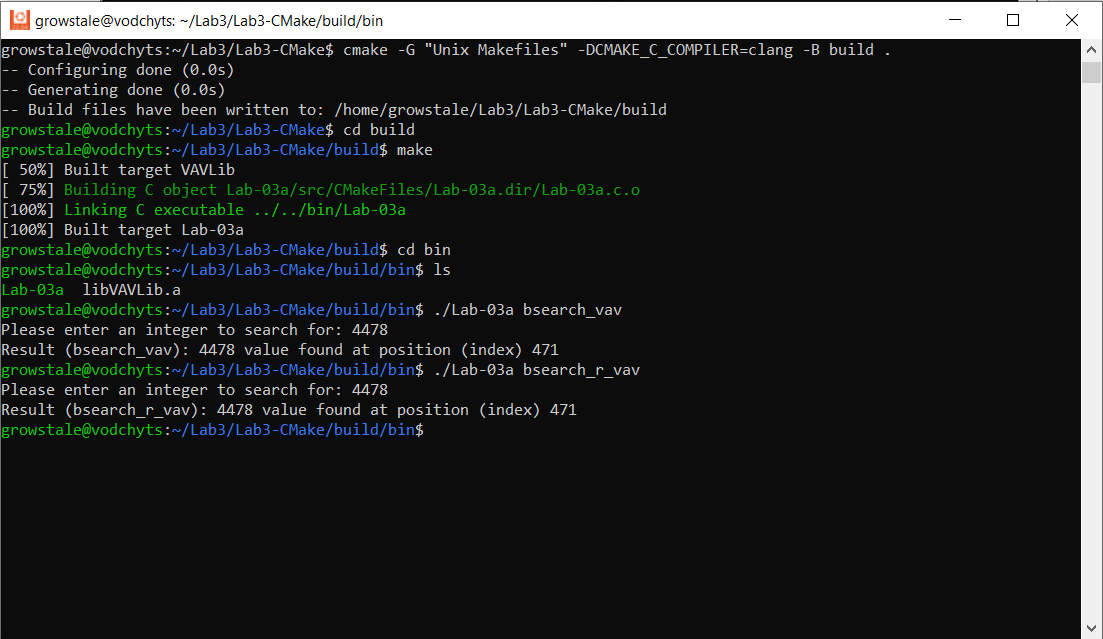
**Проект Lab-03a:**

Собрать приложение из проекта с аналогичным названием на Windows. Проверить работоспособность приложения.

**Clang**

****

**Cmake**

****

**Проект Lib<Ваши Инициалы>d:**

Разработать разделяемую библиотеку на языке **C** с заданным функционалом (см. выше). Также должна быть реализованы функции **constructor** и **destructor** с логированием событий в некоторый файл (путь к файлу должен быть абсолютным).

|  |
| --- |
| специальные атрибуты компилятора Clang, которые позволяют выполнять функции автоматически при загрузке и выгрузке библиотеки |

Просмотреть список экспортируемых символов через утилиту **objdump**/**readelf** и предоставить скриншот с его содержимым!

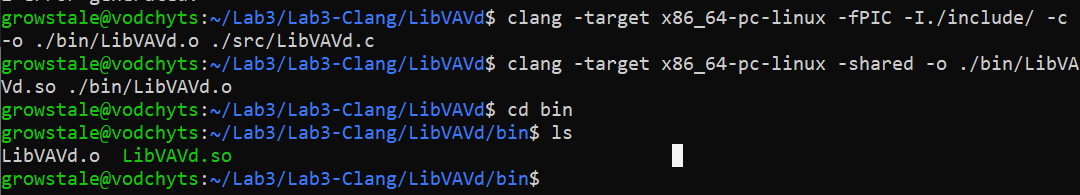
**LibVAVd.h:**

|  |
| --- |
| extern const int array[1024];  int bsearch\_vav(const int\* a, int n, int x);  int bsearch\_r\_vav(const int\* a, int x, int i, int j);  void \_\_attribute\_\_((constructor)) library\_open(void);  void \_\_attribute\_\_((destructor)) library\_close(void); |

**LibVAVd.c:**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <time.h>  #include <unistd.h>  #include "LibVAVd.h"  const int array[1024] = {      4, 13, 23, 32, 41, 51, 61, 70, 79, 89, 98, 108,      118, 127, 136, 146, 156, 165, 174, 184, 194, 203, 212, 222,      232, 241, 251, 260, 269, 279, 288, 298, 307, 317, 327, 336,      345, 355, 364, 374, 384, 393, 402, 412, 422, 431, 440, 450,      460, 469, 478, 488, 498, 507, 517, 526, 535, 545, 554, 564,      574, 583, 592, 602, 611, 621, 630, 640, 650, 659, 668, 678,      688, 697, 706, 716, 725, 735, 745, 754, 763, 773, 782, 792,      802, 811, 820, 830, 840, 849, 858, 868, 878, 887, 896, 906,      915, 925, 935, 944, 953, 963, 972, 982, 992, 1001, 1010, 1020,      1030, 1039, 1049, 1058, 1067, 1077, 1086, 1096, 1106, 1115, 1124, 1134,      1143, 1153, 1163, 1172, 1181, 1191, 1201, 1210, 1219, 1229, 1239, 1248,      1257, 1267, 1277, 1286, 1295, 1305, 1314, 1324, 1334, 1343, 1352, 1362,      1371, 1381, 1391, 1400, 1409, 1419, 1429, 1438, 1447, 1457, 1466, 1476,      1486, 1495, 1504, 1514, 1524, 1533, 1542, 1552, 1561, 1571, 1581, 1590,      1599, 1609, 1619, 1628, 1637, 1647, 1656, 1666, 1676, 1685, 1694, 1704,      1713, 1723, 1733, 1742, 1751, 1761, 1771, 1780, 1789, 1799, 1808, 1818,      1828, 1837, 1846, 1856, 1866, 1875, 1884, 1894, 1903, 1913, 1923, 1932,      1941, 1951, 1961, 1970, 1979, 1989, 1998, 2008, 2018, 2027, 2036, 2046,      2055, 2065, 2075, 2084, 2093, 2103, 2112, 2122, 2132, 2141, 2150, 2160,      2170, 2179, 2188, 2198, 2207, 2217, 2227, 2236, 2245, 2255, 2264, 2274,      2284, 2293, 2302, 2312, 2322, 2331, 2340, 2350, 2359, 2369, 2379, 2388,      2397, 2407, 2417, 2426, 2435, 2445, 2454, 2464, 2474, 2483, 2492, 2502,      2511, 2521, 2531, 2540, 2549, 2559, 2568, 2578, 2588, 2597, 2606, 2616,      2625, 2635, 2645, 2654, 2663, 2673, 2683, 2692, 2701, 2711, 2720, 2730,      2740, 2749, 2758, 2768, 2777, 2787, 2797, 2806, 2815, 2825, 2834, 2844,      2854, 2863, 2872, 2882, 2892, 2901, 2910, 2920, 2929, 2939, 2949, 2958,      2967, 2977, 2987, 2996, 3005, 3015, 3024, 3034, 3044, 3053, 3062, 3072,      3081, 3091, 3101, 3110, 3119, 3129, 3138, 3148, 3158, 3167, 3176, 3186,      3195, 3205, 3215, 3224, 3233, 3243, 3253, 3262, 3271, 3281, 3290, 3300,      3310, 3319, 3328, 3338, 3347, 3357, 3367, 3376, 3385, 3395, 3404, 3414,      3424, 3433, 3442, 3452, 3461, 3471, 3481, 3490, 3499, 3509, 3518, 3528,      3538, 3547, 3556, 3566, 3576, 3585, 3594, 3604, 3613, 3623, 3633, 3642,      3651, 3661, 3670, 3680, 3690, 3699, 3708, 3718, 3727, 3737, 3747, 3756,      3765, 3775, 3784, 3794, 3804, 3813, 3822, 3832, 3841, 3851, 3861, 3870,      3879, 3889, 3898, 3908, 3918, 3927, 3936, 3946, 3955, 3965, 3975, 3984,      3993, 4003, 4013, 4022, 4031, 4041, 4050, 4060, 4070, 4079, 4088, 4098,      4107, 4117, 4127, 4136, 4145, 4155, 4164, 4174, 4184, 4193, 4202, 4212,      4221, 4231, 4241, 4250, 4259, 4269, 4279, 4288, 4297, 4307, 4316, 4326,      4336, 4345, 4354, 4364, 4373, 4383, 4393, 4402, 4411, 4421, 4430, 4440,      4450, 4459, 4468, 4478, 4487, 4497, 4507, 4516, 4525, 4535, 4544, 4554,      4564, 4573, 4582, 4592, 4602, 4611, 4620, 4630, 4639, 4649, 4659, 4668,      4677, 4687, 4696, 4706, 4716, 4725, 4734, 4744, 4753, 4763, 4773, 4782,      4791, 4801, 4811, 4820, 4829, 4839, 4848, 4858, 4868, 4877, 4886, 4896,      4905, 4915, 4925, 4934, 4943, 4953, 4963, 4972, 4981, 4991, 5000, 5010,      5020, 5029, 5038, 5048, 5057, 5067, 5077, 5086, 5095, 5105, 5114, 5124,      5134, 5143, 5152, 5162, 5171, 5181, 5191, 5200, 5209, 5219, 5228, 5238,      5248, 5257, 5266, 5276, 5286, 5295, 5304, 5314, 5323, 5333, 5343, 5352,      5361, 5371, 5380, 5390, 5400, 5409, 5418, 5428, 5437, 5447, 5457, 5466,      5475, 5485, 5494, 5504, 5514, 5523, 5532, 5542, 5551, 5561, 5571, 5580,      5589, 5599, 5608, 5618, 5628, 5637, 5646, 5656, 5666, 5675, 5684, 5694,      5703, 5713, 5723, 5732, 5741, 5751, 5760, 5770, 5780, 5789, 5798, 5808,      5817, 5827, 5837, 5846, 5855, 5865, 5874, 5884, 5894, 5903, 5912, 5922,      5931, 5941, 5951, 5960, 5969, 5979, 5989, 5998, 6007, 6017, 6026, 6036,      6046, 6055, 6064, 6074, 6083, 6093, 6103, 6112, 6121, 6131, 6140, 6150,      6160, 6169, 6178, 6188, 6197, 6207, 6217, 6226, 6235, 6245, 6254, 6264,      6274, 6283, 6292, 6302, 6311, 6321, 6331, 6340, 6349, 6359, 6368, 6378,      6388, 6397, 6406, 6416, 6426, 6435, 6444, 6454, 6463, 6473, 6483, 6492,      6501, 6511, 6520, 6530, 6540, 6549, 6558, 6568, 6577, 6587, 6597, 6606,      6615, 6625, 6634, 6644, 6654, 6663, 6672, 6682, 6691, 6701, 6711, 6720,      6729, 6739, 6749, 6758, 6767, 6777, 6786, 6796, 6806, 6815, 6824, 6834,      6843, 6853, 6863, 6872, 6881, 6891, 6901, 6910, 6919, 6929, 6938, 6948,      6958, 6967, 6976, 6986, 6995, 7005, 7015, 7024, 7033, 7043, 7052, 7062,      7072, 7081, 7090, 7100, 7109, 7119, 7129, 7138, 7147, 7157, 7166, 7176,      7186, 7195, 7204, 7214, 7223, 7233, 7243, 7252, 7261, 7271, 7280, 7290,      7300, 7309, 7318, 7328, 7337, 7347, 7357, 7366, 7375, 7385, 7394, 7404,      7414, 7423, 7432, 7442, 7451, 7461, 7471, 7480, 7489, 7499, 7508, 7518,      7528, 7537, 7546, 7556, 7566, 7575, 7584, 7594, 7603, 7613, 7623, 7632,      7641, 7651, 7660, 7670, 7680, 7689, 7698, 7708, 7717, 7727, 7737, 7746,      7755, 7765, 7774, 7784, 7794, 7803, 7812, 7822, 7831, 7841, 7851, 7860,      7869, 7879, 7888, 7898, 7908, 7917, 7926, 7936, 7946, 7955, 7964, 7974,      7983, 7993, 8003, 8012, 8021, 8031, 8040, 8050, 8060, 8069, 8078, 8088,      8097, 8107, 8117, 8126, 8135, 8145, 8154, 8164, 8174, 8183, 8192, 8202,      8211, 8221, 8231, 8240, 8249, 8259, 8268, 8278, 8288, 8297, 8306, 8316,      8325, 8335, 8345, 8354, 8363, 8373, 8382, 8392, 8402, 8411, 8420, 8430,      8439, 8449, 8459, 8468, 8477, 8487, 8496, 8506, 8516, 8525, 8534, 8544,      8553, 8563, 8573, 8582, 8591, 8601, 8611, 8620, 8629, 8639, 8648, 8658,      8668, 8677, 8686, 8696, 8705, 8715, 8725, 8734, 8743, 8753, 8762, 8772,      8782, 8791, 8800, 8810, 8819, 8829, 8839, 8848, 8857, 8867, 8876, 8886,      8896, 8905, 8914, 8924, 8933, 8943, 8953, 8962, 8971, 8981, 8991, 9000,      9009, 9019, 9028, 9038, 9048, 9057, 9066, 9076, 9085, 9095, 9105, 9114,      9123, 9133, 9142, 9152, 9162, 9171, 9180, 9190, 9199, 9209, 9219, 9228,      9237, 9247, 9256, 9266, 9276, 9285, 9294, 9304, 9313, 9323, 9333, 9342,      9351, 9361, 9371, 9380, 9389, 9399, 9408, 9418, 9428, 9437, 9446, 9456,      9465, 9475, 9485, 9494, 9503, 9513, 9522, 9532, 9542, 9551, 9560, 9570,      9579, 9589, 9599, 9608, 9617, 9627, 9636, 9646, 9656, 9665, 9674, 9684,      9693, 9703, 9713, 9722  };  int bsearch\_vav(const int\* a, int n, int x) {      int i = 0, j = n - 1;      while (i <= j) {          int k = i + ((j - i) / 2);          if (a[k] == x) {              return k;          }          else if (a[k] < x) {              i = k + 1;          }          else {              j = k - 1;          }      }      return -1;  }  int bsearch\_r\_vav(const int\* a, int x, int i, int j) {      if (j < i) {          return -1;      }      int k = i + ((j - i) / 2);      if (a[k] == x) {          return k;      }      else if (a[k] < x) {          return bsearch\_r\_vav(a, x, k + 1, j);      }      else {          return bsearch\_r\_vav(a, x, i, k - 1);      }  }  void \_\_attribute\_\_((constructor)) library\_open(void) {      FILE \*log\_handle = fopen("/home/growstale/Lab3/myfile.txt ", "a");      if (log\_handle != NULL) {          time\_t current\_time = time(NULL);          char \*time\_string = ctime(&current\_time);          size\_t len = strlen(time\_string);          time\_string[len - 1] = '\0';          fprintf(log\_handle, "Library Initialized: %s\n", time\_string);          fclose(log\_handle);      }  }  void \_\_attribute\_\_((destructor)) library\_close(void) {      FILE \*log\_handle = fopen("/home/growstale/Lab3/myfile.txt", "a");      if (log\_handle != NULL) {          time\_t exit\_time = time(NULL);          char \*time\_string = ctime(&exit\_time);          size\_t len = strlen(time\_string);          time\_string[len - 1] = '\0';          fprintf(log\_handle, "Library Exiting: %s\n", time\_string);          fclose(log\_handle);      }  } |

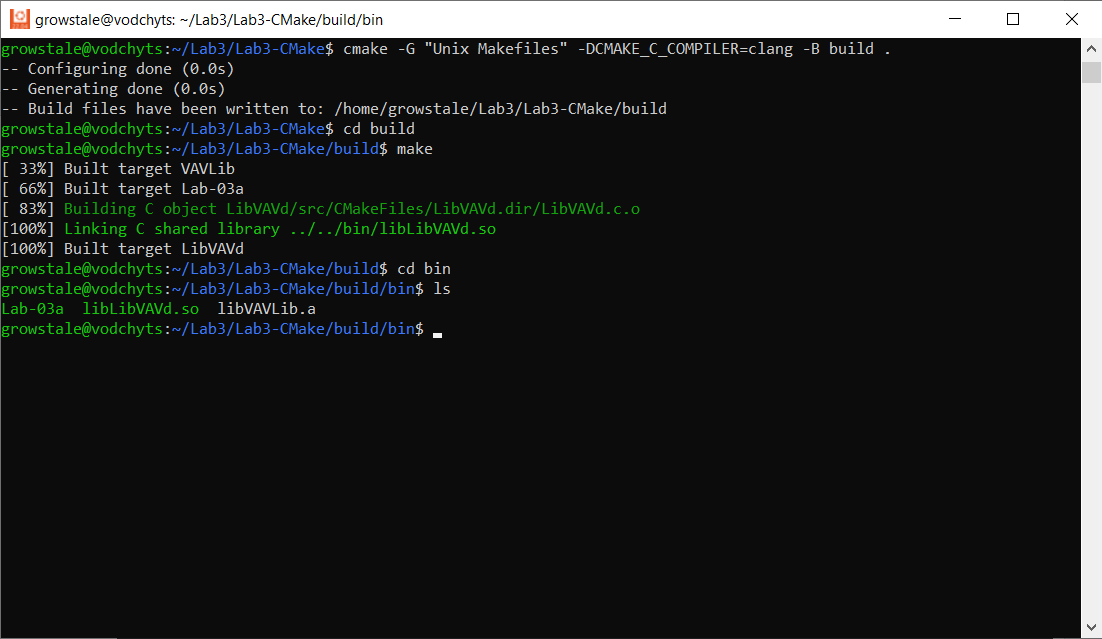
**Clang**

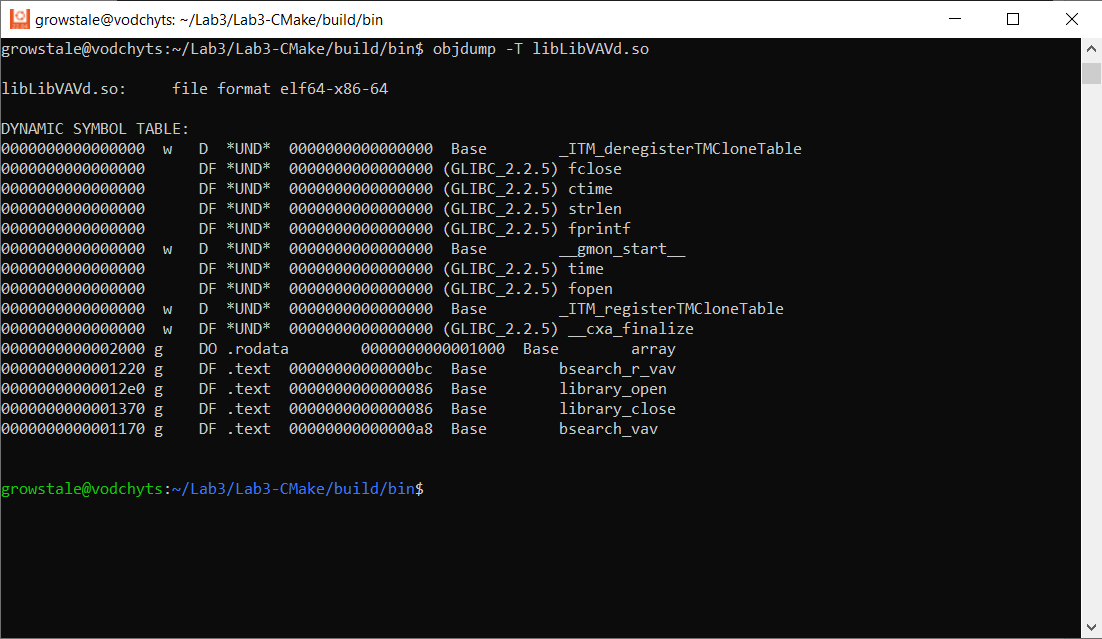
****

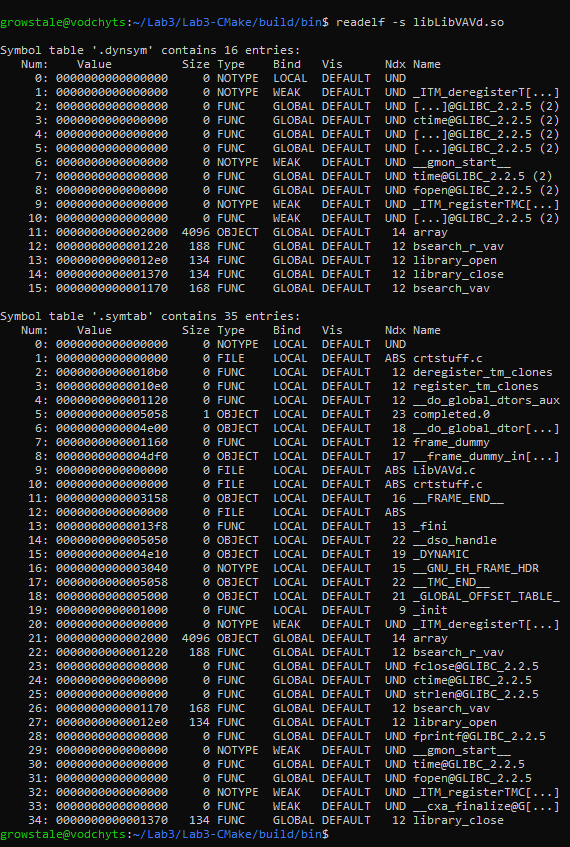
**CMake**

**CmakeLists.txt:**

|  |
| --- |
| cmake\_minimum\_required(VERSION 3.20)  include\_directories(${CMAKE\_SOURCE\_DIR}/LibVAVd/include)  add\_library(LibVAVd SHARED LibVAVd.c) |

****

****

****

objdump -T — показывает таблицу символов динамической линковки (Dynamic Symbol Table). Используется для анализа экспортируемых/импортируемых символов в ELF.

readelf -s — показывает все символы (символьную таблицу), включая как статические, так и динамические. То есть более полная информация, чем -T.

**Проект Lab-03c:**

Разработать клиентское приложение которое сможет **явно** подключать разделяемую библиотеку. Приложение должно быть консольным и может быть вызывано в виде:

***Lab-03с <Подключаемая библиотека> <Имя или номер вызываемой функции> [Искомое число]***

**Информация!** Параметры «Подключаемая библиотека» и «Имя вызываемой функции» должны быть непосредственно значениями передаваемыми в функции **dlopen** и **dlsym**.

Если искомое число не задано, то оно запрашивается для ввода с консоли (для наглядности работы динамической библиотеки поместите этот код непосредственно перед вызовом функции поиска). В результате выполнения этого приложения должен быть выведен один из следующих результатов:

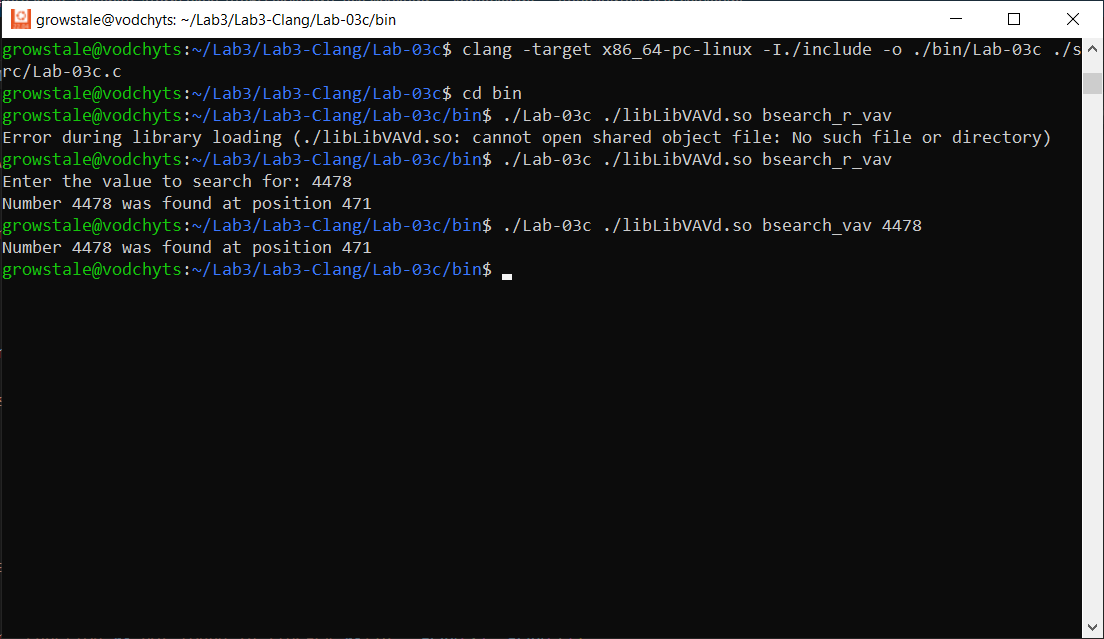
* Не указана загружаемая библиотека!
* Не указана вызываемая функция!
* Загружаемая библиотека не найдена!
* Искомая функция <Имя вызываемой функции> не найдена в библиотеке <Имя загруженной библиотеки>!
* <Имя вызванной функции>: Заданное число не найдено!
* <Имя вызванной функции>: Число <Х> найдено на позиции <Y>!

Просмотреть список импортируемых символов через утилиту **objdump**/**readelf** и предоставить скриншот с его содержимым!

Lab-03c.c:

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <dlfcn.h>  #include <string.h>  typedef int (\*IterativeBSearchFunc)(int\*, int, int);  typedef int (\*RecursiveBSearchFunc)(int\*, int, int, int);  int main(int argc, char\* argv[]) {      if (argc < 3) {          printf("Incorrect number of arguments entered\n");          printf("Lab-03c <library> <function> [value]\n");          return 1;      }      void\* handle = dlopen(argv[1], RTLD\_NOW);      if (!handle) {          fprintf(stderr, "Error during library loading (%s)\n", dlerror());          return 1;      }      const int\* array = dlsym(handle, "array");      if (array == NULL) {          fprintf(stderr, "Error during getting array: (%s)\n", dlerror());          dlclose(handle);          return 1;      }      int x;      if (argc < 4) {          printf("Enter the value to search for: ");          if (scanf("%d", &x) != 1) {              printf("Invalid input.\n");              dlclose(handle);              return 1;          }      } else {          x = atoi(argv[3]);      }      if (strcmp(argv[2], "bsearch\_vav") == 0) {          IterativeBSearchFunc func = dlsym(handle, argv[2]);          if (!func) {              fprintf(stderr, "Error: Function %s not found in library %s!\n", argv[2], argv[1]);              dlclose(handle);              return 1;          }          int index = func((int\*)array, 1024, x);          if (index == -1) {              printf("The number was not found\n");          }          else {              printf("Number %d was found at position %d\n", x, index);          }      }      else if (strcmp(argv[2], "bsearch\_r\_vav") == 0) {          RecursiveBSearchFunc func = dlsym(handle, argv[2]);          if (!func) {              fprintf(stderr, "Error during library loading (%s)\n", argv[2], argv[1]);              dlclose(handle);              return 1;          }          int index = func((int\*)lib\_array, x, 0, 1023);          if (index == -1) {              printf("The number was not found\n");          }          else {              printf("Number %d was found at position %d\n", x, index);          }      }      else {          printf("Functions that can be used: bsearch\_vav, bsearch\_r\_vav\n");          dlclose(handle);          return 1;      }      dlclose(handle);      return 0;  } |

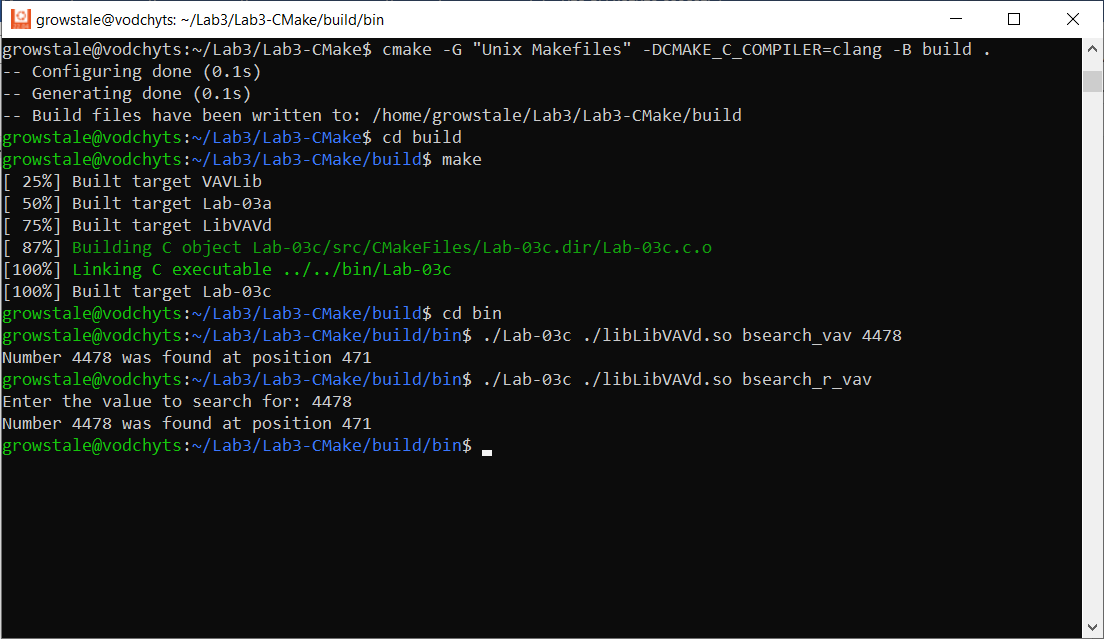
**Clang**

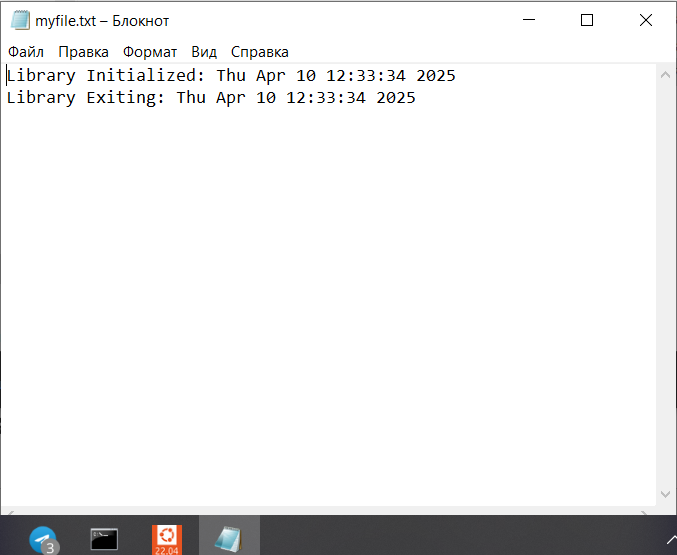
****

**CMake**

**CmakeLists.txt:**

|  |
| --- |
| cmake\_minimum\_required(VERSION 3.20)  add\_executable(Lab-03c Lab-03c.c) |

****

****

**Дополнительно (совершенно не обязательно, но огромный плюс):**

Написать примеры внедрения кода с использованием динамических библиотек как на Windows, так и на Linux (примеры из лекций сдавать как свои работы бесполезно ^\_^).

**Требования к выполненной работе:**

* Для генерации и сборки проекта использовать как средство CMake, так и clang/clang++ напрямую;
* Для обоих случаев выучить команды (консольные или в CMakeLists.txt) которые используются для сборки библиотек или приложений использующих библиотеки!
* Для обоих случаев подготовить серию скриншотов со сборкой библиотек и приложений!

При использовании CMake должно быть выполнено следующее:

* Сборка должна быть «out-of-source»;
* Для Windows использовать генератор «Visual Studio»/«Ninja» для Linux – «Unix Makefiles», для macOS – «XCode»;
* При сборке проектов всегда должен использоваться clang;
* Все проекты для обеих ОС являются частью одного большого проекта. Добавить в конфигурационные файлы условия сборки отдельных проектов для разных ОС. Конечные файлы генерируются в общий каталог!

**Вопросы для контроля:**

1. Что такое библиотека?
2. Что такое статическая библиотека?
3. Как подключить статическую библиотеку?
4. На каком этапе сборки происходит непосредственно загрузка кода?
5. Что такое динамическая библиотека?
6. Какой механизм лежит в основе работы динамических библиотек?
7. Назовите два способа подключения динамической библиотеки? Кратко поясните порядок подключений.
8. Что такое библиотека импорта?
9. Для чего нужен extern “C”?
10. Функции жиненного цикла динамических библиотек в Windows и Linux?

Noname Data:

**MyDll.c:**

|  |
| --- |
| #include <windows.h>  int sharedDataVariable = 123;  BOOL APIENTRY DllMain( HMODULE hModule,                         DWORD  ul\_reason\_for\_call,                         LPVOID lpReserved                       )  {      switch (ul\_reason\_for\_call)      {      case DLL\_PROCESS\_ATTACH:      case DLL\_THREAD\_ATTACH:      case DLL\_THREAD\_DETACH:      case DLL\_PROCESS\_DETACH:          break;      }      return TRUE;  } |

**MyDll.def:**

|  |
| --- |
| LIBRARY MyDll  EXPORTS  sharedDataVariable DATA @1 NONAME |

**main.c:**

|  |
| --- |
| #include <windows.h>  #include <stdio.h>  int main() {      HINSTANCE hDll = LoadLibraryA("mydll.dll");      DWORD lastError;      if (!hDll) {          lastError = GetLastError();          fprintf(stderr, "Error: Could not load mydll.dll. Code: %lu\n", lastError);          return 1;      }      printf("mydll.dll loaded successfully.\n");      // Access data variable exported by 1, NONAME      int\* dataPtr = (int\*)GetProcAddress(hDll, MAKEINTRESOURCE(1));      if (dataPtr) {          printf("Accessing sharedDataVariable (ordinal 1): Value = %d\n", \*dataPtr);      } else {          lastError = GetLastError();          fprintf(stderr, "Error: Could not get address of data at ordinal 1. Code: %lu\n", lastError);      }      void\* failedPtr = GetProcAddress(hDll, "sharedDataVariable");      if (!failedPtr) {           DWORD nameError = GetLastError();           printf("As expected, failed to get data address using name 'sharedDataVariable' due to NONAME. (Error: %lu)\n", nameError);      } else {           fprintf(stderr, "Unexpected: Succeeded in getting data address using name 'sharedDataVariable'!\n");      }      FreeLibrary(hDll);      printf("mydll.dll unloaded.\n");      return 0;  } |

