

Факультет _____ О _____
 _____ индекс факультета
 Выпускающая кафедра _____ О7 _____
 _____ индекс кафедры
 Группа _____ О711Б _____
 _____ индекс группы

Когай Егора Денисовича

Обучающегося по направлению	09.03.02	Информационные системы и технологии
	код	полное наименование направления

Должность обучающегося на практике: студент

 Подпись

 Вальштейн К.В.
 Фамилия ИО

« » _____ 2022 г.

САНКТ-ПЕТЕРБУРГ
2022 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Практическая работа №1	4
2 Практическая работа №2	10
3 Практическая работа №3	17
4 Практическая работа №4	21
5 Практическая работа №5	24
ЗАКЛЮЧЕНИЕ	31
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	32

ВВЕДЕНИЕ

В рамках учебной практики необходимо выполнить 5 практических работ:

- изучить особенности работы с набором компиляторов и утилит GNU Compiler Collection;
- ознакомиться с особенностями использования отладчика GDB;
- изучить процесс создания динамических библиотек при помощи набора компиляторов и утилит GCC и особенности их применения;
- изучить особенности работы с программой управления компиляцией на примере утилиты make;
- изучить особенности различных систем контроля версий.

1 Практическая работа №1

Тема: особенности использования набора компиляторов и утилит GCC.

Цель работы: изучить особенности работы с набором компиляторов и утилит GNU Compiler Collection (GCC).

Для выполнения первой части работы необходимо выполнить следующие три задачи:

1. Написать программу 1 в соответствии с вариантом при помощи любого текстового редактора.
2. Провести поэтапную компиляцию исходного текста написанной программы, разобраться в результатах, полученных на каждом этапе компиляции.
3. Провести оптимизацию кода написанной программы с помощью набора компиляторов GCC, пояснить внесённые для оптимизации кода изменения.

Задача 1: среди четырехзначных чисел выбрать те, у которых все четыре цифры различны.

Текст программы:

```
#include <stdio.h>

int main() {
    int a, b, c, d, i;
    for (i = 1000; i < 10000; ++i)
    {
        a = i / 1000;
        b = i / 100 % 10;
        c = i / 10 % 10;
        d = i % 10;
        if (a != b && b != c && c != d && a != c && a != d && b != d)
            printf("%d%d%d%d ", a,b,c,d);
    }
}
```

Результат работы программы представлен на рисунке 1.1.

```
C:\Users\ASUS>C:\stud\CP1\CP1.1\CP1.exe
1823 1824 1825 1826 1827 1828 1829 1832 1834 1835 1836 1837 1838 1839 1842 1843 1845 1846 1847 1848 1849 1852 1853 1854 1856 1857 1858 1859 1862 1863 1864 1865 1867 1868 1869 1872 1873 1874
230 1234 1235 1236 1237 1238 1239 1240 1243 1245 1246 1247 1248 1249 1250 1253 1254 1256 1257 1258 1259 1260 1263 1264 1265 1267 1268 1269 1270 1273 1274 1275 1276 1278 1279 1280 1283 1284 1
40 1342 1345 1346 1347 1348 1349 1350 1352 1354 1356 1357 1358 1359 1360 1362 1364 1365 1367 1368 1369 1370 1372 1374 1375 1376 1378 1379 1380 1382 1384 1385 1386 1387 1389 1390 1392 1394 13
0 1452 1453 1456 1457 1458 1459 1460 1462 1463 1465 1467 1468 1469 1470 1472 1473 1475 1476 1478 1479 1480 1482 1483 1485 1486 1487 1489 1490 1492 1493 1495 1496 1497 1498 1502 1503 1504 150
1562 1563 1564 1567 1568 1569 1570 1572 1573 1574 1576 1578 1579 1580 1582 1583 1584 1586 1587 1589 1590 1592 1593 1594 1596 1597 1598 1602 1603 1604 1605 1607 1608 1609 1620 1623 1624 1625
1672 1673 1674 1675 1678 1679 1680 1682 1683 1684 1685 1687 1689 1690 1692 1693 1694 1695 1697 1698 1702 1703 1704 1705 1706 1708 1709 1720 1723 1724 1725 1726 1728 1729 1730 1732 1734 1735
782 1783 1784 1785 1786 1789 1790 1792 1793 1794 1795 1796 1798 1802 1803 1804 1805 1806 1807 1809 1820 1823 1824 1825 1826 1827 1829 1830 1832 1834 1835 1836 1837 1839 1840 1842 1843 1845 1
92 1893 1894 1895 1896 1897 1898 1899 1902 1903 1904 1905 1906 1907 1908 1920 1923 1924 1925 1926 1927 1928 1930 1932 1934 1935 1936 1937 1938 1940 1942 1943 1945 1946 1947 1948 1950 1952 1953 1954 19
4 2015 2016 2017 2018 2019 2021 2024 2025 2026 2027 2028 2039 2041 2043 2045 2046 2047 2048 2049 2051 2053 2054 2056 2057 2058 2059 2061 2063 2064 2065 2067 2068 2069 2071 2073 2074 2075 207
2135 2136 2137 2138 2139 2140 2143 2145 2146 2147 2148 2149 2150 2153 2154 2156 2157 2158 2159 2160 2163 2164 2165 2167 2168 2169 2170 2173 2174 2175 2176 2178 2179 2180 2183 2184 2185 2186
2345 2346 2347 2348 2349 2350 2351 2354 2356 2357 2358 2359 2360 2361 2364 2365 2367 2368 2369 2370 2371 2374 2375 2376 2378 2379 2380 2381 2384 2385 2386 2387 2389 2390 2391 2394 2395 2396
453 2456 2457 2458 2459 2460 2461 2463 2465 2467 2468 2469 2470 2471 2473 2475 2476 2478 2479 2480 2481 2483 2485 2486 2487 2489 2490 2491 2493 2495 2496 2497 2498 2501 2503 2504 2506 2507 2
63 2564 2567 2568 2569 2570 2571 2573 2574 2576 2578 2579 2580 2581 2583 2584 2586 2587 2589 2590 2591 2593 2594 2596 2597 2598 2601 2603 2604 2605 2607 2608 2609 2610 2613 2614 2615 2617 26
3 2674 2675 2678 2679 2680 2681 2683 2684 2685 2687 2689 2690 2691 2693 2694 2695 2697 2698 2701 2703 2704 2705 2706 2708 2709 2710 2713 2714 2715 2716 2718 2719 2730 2731 2734 2735 2736 273
2784 2785 2786 2789 2790 2791 2793 2794 2795 2796 2798 2801 2803 2804 2805 2806 2807 2809 2810 2813 2814 2815 2816 2817 2819 2830 2831 2834 2835 2836 2837 2839 2840 2841 2843 2845 2846 2847
2884 2885 2886 2887 2901 2903 2904 2905 2906 2907 2908 2910 2913 2914 2915 2916 2917 2918 2920 2931 2934 2935 2936 2937 2938 2940 2941 2943 2945 2946 2947 2948 2950 2951 2953 2954 2956 2957
916 3017 3018 3019 3021 3024 3025 3026 3027 3028 3029 3041 3042 3045 3046 3047 3048 3049 3051 3052 3054 3056 3057 3058 3059 3061 3062 3064 3065 3067 3068 3069 3071 3072 3074 3075 3076 3078 3
26 3127 3128 3129 3140 3142 3145 3146 3147 3148 3149 3150 3152 3154 3156 3157 3158 3159 3160 3162 3164 3165 3167 3168 3169 3170 3172 3174 3175 3176 3178 3179 3180 3182 3184 3185 3186 3187 31
6 3247 3248 3249 3250 3251 3254 3256 3257 3258 3259 3260 3261 3264 3265 3267 3268 3269 3270 3271 3274 3275 3276 3278 3279 3280 3281 3284 3285 3286 3287 3289 3290 3291 3294 3295 3296 3297 329
3 3457 3458 3459 3460 3461 3462 3465 3467 3468 3469 3470 3471 3472 3475 3476 3478 3479 3480 3481 3482 3485 3486 3487 3489 3490 3491 3492 3495 3496 3497 3498 3501 3502 3504 3506 3507 3508 3509
3567 3568 3569 3570 3571 3572 3574 3576 3578 3579 3580 3581 3582 3584 3586 3587 3589 3590 3591 3592 3594 3596 3597 3598 3601 3602 3604 3605 3607 3608 3609 3610 3612 3614 3615 3617 3618 3619
675 3678 3679 3680 3681 3682 3684 3685 3687 3689 3690 3691 3692 3694 3695 3697 3698 3701 3702 3704 3705 3706 3708 3709 3710 3712 3714 3715 3716 3718 3719 3720 3721 3724 3725 3726 3728 3729 3
85 3786 3789 3790 3791 3792 3794 3795 3796 3798 3801 3802 3804 3805 3806 3807 3809 3810 3812 3814 3815 3816 3817 3819 3820 3821 3824 3825 3826 3827 3829 3840 3841 3842 3845 3846 3847 3849 38
5 3896 3897 3901 3902 3904 3905 3906 3907 3908 3910 3912 3914 3915 3916 3917 3918 3920 3921 3924 3925 3926 3927 3928 3940 3941 3942 3945 3946 3947 3948 3950 3951 3952 3954 3956 3957 3958 396
4018 4019 4021 4022 4025 4026 4027 4028 4029 4031 4032 4035 4036 4037 4038 4039 4041 4052 4053 4056 4057 4058 4059 4061 4062 4063 4065 4067 4068 4069 4071 4072 4073 4075 4076 4078 4079 4081
4128 4129 4130 4132 4135 4136 4137 4138 4139 4150 4152 4153 4156 4157 4158 4159 4160 4162 4163 4165 4167 4168 4169 4170 4172 4173 4175 4176 4178 4179 4180 4182 4183 4185 4186 4187 4189 4190
238 4239 4250 4251 4253 4256 4257 4258 4259 4260 4261 4263 4265 4267 4268 4269 4270 4271 4273 4275 4276 4278 4279 4280 4281 4283 4285 4286 4287 4289 4290 4291 4293 4295 4296 4297 4298 4301 4
8 4359 4360 4361 4362 4365 4367 4368 4369 4370 4371 4372 4375 4376 4378 4379 4380 4381 4382 4385 4386 4387 4389 4390 4391 4392 4395 4396 4397 4398 4501 4502 4503 4506 4507 4508 4509 4510 45
8 4569 4570 4571 4572 4573 4576 4578 4579 4580 4581 4582 4583 4586 4587 4589 4590 4591 4592 4593 4596 4597 4598 4601 4602 4603 4605 4607 4608 4609 4610 4612 4613 4615 4617 4618 4619 4620 462
4679 4680 4681 4682 4683 4685 4687 4689 4690 4691 4692 4693 4695 4697 4698 4701 4702 4703 4705 4706 4708 4709 4710 4712 4713 4715 4716 4718 4719 4720 4721 4723 4725 4726 4728 4729 4730 4731
4789 4790 4791 4792 4793 4795 4796 4798 4801 4802 4803 4805 4806 4807 4809 4810 4812 4813 4815 4816 4817 4819 4820 4821 4823 4825 4826 4827 4829 4830 4831 4832 4835 4836 4837 4839 4850 4851
897 4901 4902 4903 4905 4906 4907 4908 4910 4912 4913 4915 4916 4917 4918 4920 4921 4923 4925 4926 4927 4928 4930 4931 4932 4935 4936 4937 4938 4950 4951 4952 4953 4956 4957 4958 4960 4961 4
19 5021 5023 5024 5026 5027 5028 5029 5031 5032 5034 5036 5037 5038 5039 5041 5042 5043 5046 5047 5048 5049 5061 5062 5063 5064 5067 5068 5069 5071 5072 5073 5074 5076 5078 5079 5081 5082 50
9 5130 5132 5134 5136 5137 5138 5139 5140 5142 5143 5146 5147 5148 5149 5160 5162 5163 5164 5167 5168 5169 5170 5172 5173 5174 5176 5178 5179 5180 5182 5183 5184 5186 5187 5189 5190 5192 519
5240 5241 5243 5246 5247 5248 5249 5260 5261 5263 5264 5267 5268 5269 5270 5271 5273 5274 5276 5278 5279 5280 5281 5283 5284 5286 5287 5289 5290 5291 5293 5294 5296 5297 5298 5301 5302 5304
5360 5361 5362 5364 5367 5368 5369 5370 5371 5372 5374 5376 5378 5379 5380 5381 5382 5384 5386 5387 5389 5390 5391 5392 5394 5396 5397 5398 5401 5402 5403 5406 5407 5408 5409 5410 5412 5413
```

Рисунок 1.1 – Результат работы первой программы

Команды для компиляции представлены на рисунке 1.2.

```
Microsoft Windows [Version 10.0.19044.1889]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\ASUS>cd %PATH%

C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -o C:\stud\CP1.exe C:\stud\CP1.c

C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -E C:\stud\CP1.i C:\stud\CP1.c
gcc: error: C:\stud\CP1.i: No such file or directory

C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -E -o C:\stud\CP1.i C:\stud\CP1.c

C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -S C:\stud\CP1.s C:\stud\CP1.c

C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -c -o C:\stud\CP1.o C:\stud\CP1.c

C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>_
```

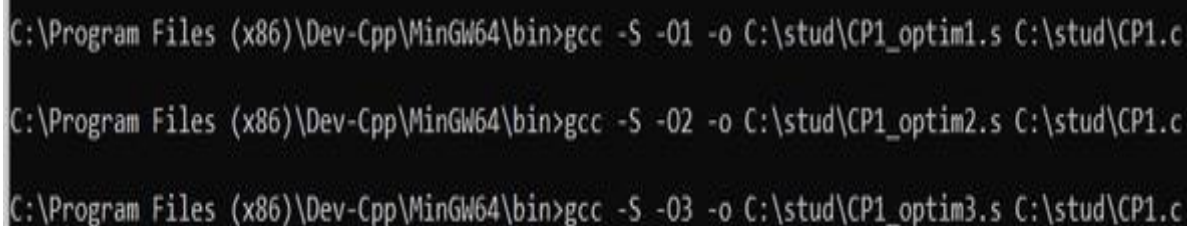
Рисунок 1.2 – Команды для компиляции

Команда `gcc -E -o C:\stud\CP1.i C:\stud\CP1.c`, создает файл `CP1.i`, содержащий исходный текст, обработанный препроцессором. В данный файл будет добавлено содержимое заголовочных файлов, будут удалены комментарии и раскрыты макросы.

Применив команду `gcc -S -o C:\stud\CP1.s C:\stud\CP1.c`, будет получен файл с ассемблерным кодом, соответствующим исходному тексту.

Результатом команды `gcc -c -o C:\stud\CP1.o C:\stud\CP1.c` будет объектный файл, содержащий блоки готового к исполнению машинного кода, блоки данных, а также список определенных в файле функций и внешних переменных.

Команды для оптимизации представлены на рисунке 1.3.



```
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -S -O1 -o C:\stud\CP1_optim1.s C:\stud\CP1.c  
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -S -O2 -o C:\stud\CP1_optim2.s C:\stud\CP1.c  
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -S -O3 -o C:\stud\CP1_optim3.s C:\stud\CP1.c
```

Рисунок 1.3 – Команды для компиляции

После выполнения команд `gcc -S -O1 -o C:\stud\CP1_optim1.s C:\stud\CP1.c`, `gcc -S -O2 -o C:\stud\CP1_optim2.s C:\stud\CP1.c` и `gcc -S -O3 -o C:\stud\CP1_optim3.s C:\stud\CP1.c` создадутся три файла с ассемблерным кодом. `CP1_optim1.s`, `CP1_optim2.s` и `CP1_optim3.s` – файлы с различной степенью оптимизации.

Для выполнения второй части работы необходимо выполнить следующие четыре задачи:

1. Написать программы 2 и 3 в соответствии с вариантом при помощи любого текстового редактора. Функции для работы с массивом вынести в отдельные файлы: в одном файле – для ввода/вывода массива, в другом – для обработки массива. В обеих программах должны использоваться одни и те же функции для ввода/вывода массивов, описанные в одном из этих файлов.
2. Провести отдельную компиляцию написанных файлов.
3. Скомпилировать обе программы, используя созданные объектные файлы; обе программы должны использовать один и тот же объектный файл с функциями для ввода/вывода массива.
4. Создать статическую библиотеку для ввода/вывода и обработки массива и продемонстрировать возможности по ее подключению.

Формулировка задачи 2: удалить из массива В (50) все элементы, кратные трем или пяти.

Текст программы:

Файл CP2.c:

```
int main()
```

```

{
    int B[50];
    int i, count = 0;
    srand(time(0));
    func_in_list (B, 50, &count);
    int A[count], j;
    count = 0;
    func_exec1 (A, B, 50, &count);
    func_out_list(A, count);
    return 0;
}

```

Файл func_exec.c:

```

#include "func_exec.h"
void func_exec1 (int *d, int *a, int b, int *c)
{
    int j;
    for (j = 0; j < b; j++)
    {
        if (a[j] % 2 == 0 && a[j] % 3 == 0)
        {
            d[*c] = a[j];
            *c = *c + 1;
        }
    }
}
void func_exec2 (int *a, int b, int *c, int d)
{
    int i;
    *c = a[0];
    for (i = 1; i < b; i++){
        if (a[i] == d) {
            *c = d;
            break;
        }
        else if (*c > abs(d - a[i])) {
            *c = a[i];
        }
    }
}

```

Файл func_in_out.c:

```

#include "func_exec.h"
void func_in_list (int *a, int b, int *c)
{
    int i;
    for (i = 0; i < b; i++) {
        a[i] = rand();
        if (a[i] % 2 == 0 && a[i] % 3 == 0) {
            *c = *c + 1;
        }
    }
}
void func_out_list (int *a, int b) {
    int i;
    for (i = 0; i < b; i++) {
        printf("%d ", a[i]);
    }
}

```

Результат работы второй программы показан на рисунке 1.4.

```
C:\Users\ASUS>C:\stud\CP1\CP1.2\CP2.exe
23682 11862 3786 29832 366 17034 18894 2604 7956 20790
```

Рисунок 1.4 – Результат работы второй программы

Задача 3: определить если в массив Q (10) заданное число X, и если нет, то найти ближайшее к нему.

Текст программы:

Файл CP3.c:

```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "func_in_out.h"
#include "func_exec.h"
int main()
{
    int Q[10];
    int i, x, temp, d=0;
    scanf("%d", &x);
    srand(time(0));
    func_in_list (Q, 10, &d);
    func_exec2 (Q, 10, &temp, x);
    func_out_list (Q, 10);

    printf("\n%d", temp);

    return 0;
}
```

Результат работы третьей программы показан на рисунке 1.5.

```
C:\Users\ASUS>C:\stud\CP1\CP1.2\CP3.exe
555
290 26237 14392 16019 30218 30848 7489 30343 28743 7611
290
```

Рисунок 1.5 – Результат работы третьей программы

Раздельная компиляция показана на рисунке 1.6.

```
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -c -o C:\stud\CP2.o C:\stud\CP2.c
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -c -o C:\stud\CP3.o C:\stud\CP3.c
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -c -o C:\stud\func_exec.o C:\stud\func_exec.c
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -c -o C:\stud\func_in_out.o C:\stud\func_in_out.c
C:\stud\func_in_out.c: In function 'func_out_list':
C:\stud\func_in_out.c:15:2: warning: incompatible implicit declaration of built-in function 'printf'
    printf("%d ", a[i]);
    ^
```

Рисунок 1.6 – Результат работы третьей программы

С помощью команды `gcc -o C:\stud\CP2.exe C:\stud\CP2.o C:\stud\func_exe.o C:\stud\func_in_out.o` и `gcc -o C:\stud\CP3.exe CP3.o C:\stud\func_exe.o C:\stud\func_in_out.o` создаются исполняемые файлы.

На рисунке 1.7 показано создание исполняемого файла.

```
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -o C:\stud\CP3.exe C:\stud\CP3.o C:\stud\func_exec.o C:\stud\func_in_out.o
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -o C:\stud\CP2.exe C:\stud\CP2.o C:\stud\func_exec.o C:\stud\func_in_out.o
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>
```

Рисунок 1.7 – Раздельная компиляция

Для создания статической библиотеки потребуется команда `ar crs libCP1.a C:\stud\func_exec.o C:\stud\func_in_out.o` (рисунок 1.8).

```
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -c C:\stud\func_exec.c
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -c C:\stud\func_in_out.c
C:\stud\func_in_out.c: In function 'func_out_list':
C:\stud\func_in_out.c:15:2: warning: incompatible implicit declaration of built-in function 'printf'
  printf("%d ", a[i]);
  ^
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>ar crs libCP1.a C:\stud\func_exec.o C:\stud\func_in_out.o
```

Рисунок 1.8 – Создание статической библиотеки

Для компиляции с использованием библиотеки используется команда `gcc -o <outputfile> main.o -L. -l<name_of_.a_file> [1]`.

Компиляция с использованием библиотеки показана на рисунке 1.9.

```
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -o C:\stud\main C:\stud\CP3.o -L. -lCP1
```

Рисунок 1.9 – Компиляция с помощью статической библиотеки

2 Практическая работа №2

Тема: особенности использования отладчика GDB.

Цель работы: изучить особенности работы с отладчиком GDB.

Для выполнения первой части работы необходимо выполнить следующие четыре задачи:

1. Написать программу 1 в соответствии с вариантом при помощи любого текстового редактора.
2. Скомпилировать программу с добавлением в файл отладочной информации.
3. Используя отладчик GDB, проверить значение вычисляемого в цикле выражения на каждом шаге цикла. При использовании нескольких циклов проверить все значения.
4. В отчете привести используемые команды отладчика и полученный результат.

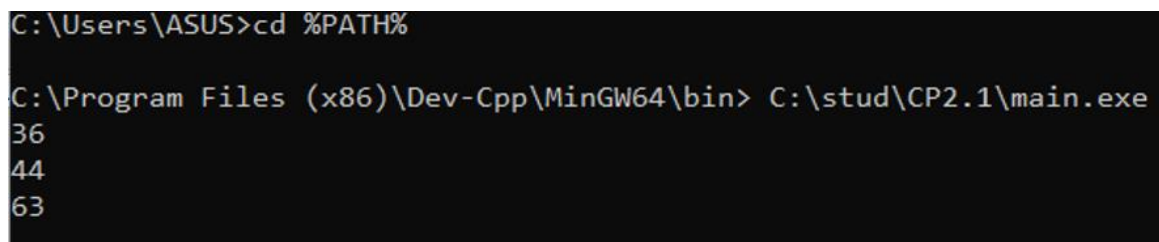
Задача 1: составить программу для определения, в каких двузначных числах удвоенная сумма цифр равна их произведению.

Текст программы:

```
#include <stdio.h>

int main(void) {
    int i, j;
    int result;
    for (i=1; i <=9; ++i) {
        for (j=0; j <= 9; ++j) {
            if (2*(i+j) == i*j) {
                result = 10*i+j;
                printf("%d\n", result);
            }
        }
    }
    return 0;
}
```

Результаты работы программы представлены на рисунках 2.1 и 2.2.



```
C:\Users\ASUS>cd %PATH%

C:\Program Files (x86)\Dev-Cpp\MinGW64\bin> C:\stud\CP2.1\main.exe
36
44
63
```

Рисунок 2.1 – Результат работы первой программы

```
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gdb C:\stud\CP2.1\main.exe
GNU gdb (GDB) 7.8.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-w64-mingw32".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
```

Рисунок 2.2 – Запуск отладчика

С помощью команды `gdb C:\stud\CP2.1\main.exe` запускается отладчик `gdb`. Команды установки точек останова представлены на рисунке 2.3.

```
Type "apropos word" to search for commands related to "word"...
Reading symbols from C:\stud\CP2.1\main.exe...done.
(gdb) b 5
Breakpoint 1 at 0x40153d: file C:\stud\CP2.1\main.c, line 5.
(gdb) b 10
Breakpoint 2 at 0x40157b: file C:\stud\CP2.1\main.c, line 10.
(gdb) r
Starting program: C:\stud\CP2.1\main.exe
[New Thread 5876.0x69f8]
[New Thread 5876.0x3a40]
```

Рисунок 2.3 – Точки останова

Устанавливаются точки останова на 5 и 10 строках с помощью команд `b 5` и `b 10`. Затем с помощью команды `r` запускается программа.

Отладка представлена на рисунке 2.4.

```
Breakpoint 1, main () at C:\stud\CP2.1\main.c:6
6   for (i=1; i <=9; ++i) {
(gdb) display result
1: result = 0
(gdb) c
Continuing.

Breakpoint 2, main () at C:\stud\CP2.1\main.c:10
10   printf("%d\n", result);
1: result = 36
(gdb) display result
2: result = 36
(gdb) c
Continuing.
36

Breakpoint 2, main () at C:\stud\CP2.1\main.c:10
10   printf("%d\n", result);
2: result = 44
1: result = 44
(gdb) c
Continuing.
44

Breakpoint 2, main () at C:\stud\CP2.1\main.c:10
10   printf("%d\n", result);
2: result = 63
1: result = 63
(gdb)
```

Рисунок 2.4 – Отладка

Команды, использованные в первой части практической работы, следующие:

- `b <число>` – устанавливает точку останова выполнения программы;
- `r` – начать исполнение программы;
- `display < имя_переменной >` – наблюдение за значением переменной программы;
- `continue/c` – продолжить исполнение программы до следующей точки останова;
- `quit/q` – завершение работы отладчика [2].

Для выполнения второй части работы необходимо выполнить следующие пять задач:

1. Написать программу 2 в соответствии с вариантом при помощи любого текстового редактора.
2. Для ввода и вывода строки использовать отдельные функции.
3. Скомпилировать программу с добавлением в файл отладочной информации.
4. Используя отладчик GDB, проверить содержимое стека при входе в функции ввода и вывода строки и выходе из них.
5. В отчете привести содержимое стека и используемые команды.

Задача 2: дана символьная строка. Оставить в ней только слова, содержащие хотя бы одну букву «А».

Текст программы:

Файл `main.c`:

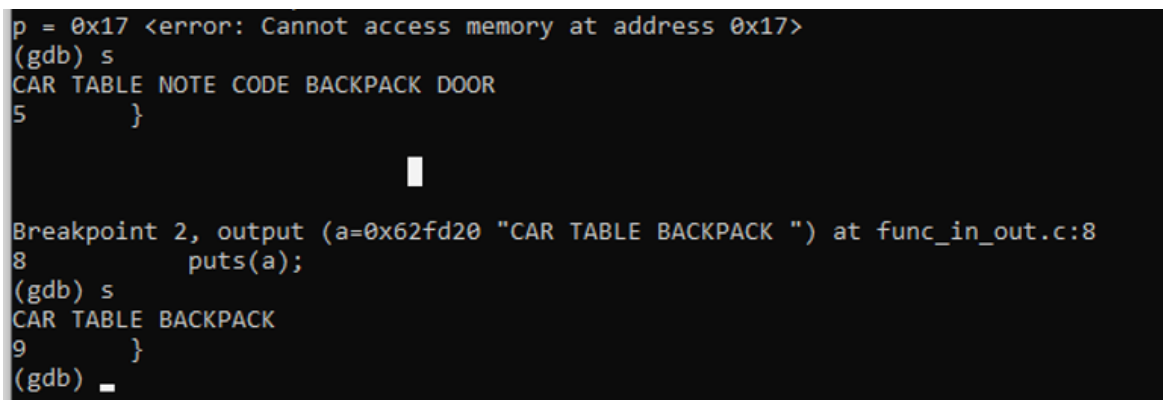
```
#include <stdio.h>
#include "func_in_out.h"
#include <locale.h>
#include <string.h>
int main() {
    char *locale = setlocale(LC_ALL, "");
    char result[100];
    char answer[100] = {};
    char *p;
    input(result);
    p = strtok(result, " ");
    while (p != NULL) {
        int i;
        for (i = 0; i < strlen(p); i++) {
```


После запуска программы происходит переход в функцию `input(char *a)`. С помощью команды `backtrace` или же просто `bt` проверяется поток и все существующие `frame`'ы. Также можно передвигаться между `frame`'ми с помощью команды `frame/f <frame's id>`. При использовании команды `info locals` проверяются только актуальные переменные для текущего `frame`'а (рисунок 2.8).

Продолжается исполнение программы и происходит переход в функцию `output(char *a)`.

Команды, использованные во второй части практической работы, следующие:

- `b <число>` – устанавливает точку останова выполнения программы;
- `r` – начать исполнение программы;
- `s` – перейти на следующую строчку исходного текста программы;
- `bt/backtrace` – вывод содержимого стектрейса;
- `frame <id>/f <id>` – переход между фреймами;
- `info locals` – актуальные переменные для текущего фрейма.



```
p = 0x17 <error: Cannot access memory at address 0x17>
(gdb) s
CAR TABLE NOTE CODE BACKPACK DOOR
5      }

Breakpoint 2, output (a=0x62fd20 "CAR TABLE BACKPACK ") at func_in_out.c:8
8      puts(a);
(gdb) s
CAR TABLE BACKPACK
9      }
(gdb) _
```

Рисунок 2.8 – Отладка

Для выполнения третьей части работы необходимо выполнить следующие пять задач:

1. Написать программу 3 в соответствии с вариантом при помощи любого текстового редактора.
2. Для ввода и вывода строки использовать отдельные функции, помещенные в статическую библиотеку.
3. Скомпилировать программу с добавлением в файл отладочной информации.

4. Используя отладчик GDB, проверить содержимое стека при входе в функции ввода и вывода строки и выходе из них.

5. В отчете привести содержимое стека и используемые команды.

Задача 3: дана строка, состоящая из букв и цифр. Преобразовать ее так, чтобы сначала шли буквы, а потом – все цифры исходной строки.

Текст программы:

Файл main.c:

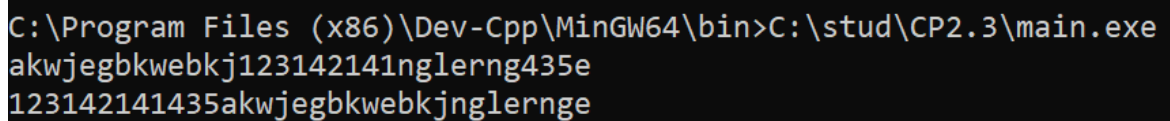
```
#include <string.h>
#include "func_in_out.h"

int main() {
    char str[100];
    char result[100] = {};
    input(str);
    int i;
    for (i = 0; i < strlen(str); i++) {
        if (str[i] >= '0' && str[i] <= '9') {
            result[strlen(result)] = str[i];
        }
    }
    for (i = 0; i < strlen(str); i++) {
        if (str[i] >= 'A' && str[i] <= 'z') {
            result[strlen(result)] = str[i];
        }
    }
    output(result);
    return 0;
}
```

Файл func_in_out.c:

```
#include <stdio.h>
void input(char *a) {
    fgets(a, 100 * sizeof(char), stdin);
}
void output(char *a) {
    puts(a);
}
```

Результат работы программы приведен на рисунке 2.9.



```
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>C:\stud\CP2.3\main.exe
akwjegbkwebkj123142141nglerng435e
123142141435akwjegbkwebkjnglernge
```

Рисунок 2.9 – Результат работы третьей программы

Описание команд показано на рисунке 2.10.


```

C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -o C:\stud\CP2.3\main.exe -g C:\stud\CP2.3\main.c C:\stud\CP2.3\func_in_out.c

C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>C:\stud\CP2.3\main.exe
akwjegbkwebkj123142141nglerng435e
123142141435akwjegbkwebkjnglernge
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gdb C:\stud\CP2.3\main.exe
GNU gdb (GDB) 7.8.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-w64-mingw32".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from C:\stud\CP2.3\main.exe...done.
(gdb) b func_in_out.c:4
No source file named func_in_out.c.
Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) b func_in_out.c:4
Breakpoint 1 at 0x40165c: file C:\stud\CP2.3\func_in_out.c, line 4.
(gdb) b func_in_out.c:8
Breakpoint 2 at 0x401689: file C:\stud\CP2.3\func_in_out.c, line 8.

```

Рисунок 2.10 – Отладка

Создается исполняемый файл и ставятся точки останова (рисунок 2.11).

```

(gdb) r
Starting program: C:\stud\CP2.3\main.exe
[New Thread 25768.0x1bbc]
[New Thread 25768.0x3c44]

Breakpoint 1, input (a=0x62fda0 "") at C:\stud\CP2.3\func_in_out.c:4
4      fgets(a, 100 * sizeof(char), stdin);
(gdb) list, 8
1      #include <stdio.h>
2
3      void input(char *a) {
4          fgets(a, 100 * sizeof(char), stdin);
5      }
6
7      void output(char *a) {
8          printf("%s", a);
(gdb) s
akwjegbkwebkj123142141nglerng435e
5      }
(gdb) c
Continuing.

Breakpoint 2, output (a=0x62fd30 "123142141435akwjegbkwebkjnglernge") at C:\stud\CP2.3\func_in_out.c:8
8      printf("%s", a);
(gdb) s
123142141435akwjegbkwebkjnglernge9      }

```

Рисунок 2.11 – Продолжение отладки

Команды, использованные в третьей части практической работы, следующие:

- b <число> – устанавливает точку останова выполнения программы;
- r – начать исполнение программы;
- s – перейти на следующую строчку исходного текста программы;
- c – продолжить исполнение программы.

3 Практическая работа №3

Тема: создание библиотек при помощи набора компилятора и утилит GCC и их применение.

Цель работы: изучить процесс создания динамической библиотеки.

Для выполнения практической работы необходимо выполнить следующие шесть задач:

1. Написать программу в соответствии с вариантом.
2. Массив и матрицу заполнять случайными числами от -50 до 50.
3. Функции для работы с массивами и матрицами поместить в две отдельные динамические библиотеки.
4. При запуске программы пользователь должен увидеть меню, в котором можно выбрать, с чем будет проходить работа: с матрицей или с массивом.
5. В зависимости от выбора пользователя загружается одна или другая динамическая библиотека.
6. Библиотеки должны быть скомпилированы с учетом возможного использования в ОС семейств Linux или Windows.

Задача: заменить все четные по значению элементы массива A (20) и матрицы B (5x6) на их квадраты.

Текст программы:

Файл main.c:

```
#include "load.h"
#include <stdio.h>

int main(void)
{
    int a=0,b=1;
    printf("Choose library:\n1-first.\n2-second,\n3-quit\n");
    while(b)
    {
        scanf("%d",&a);
        if(a==1) {
            LoadRun("lib1.dll");
        }
        if(a==2) {
            LoadRun("lib2.dll");
        }
        if(a==3) {
            b=0;
        }
    }
}
```

```

    }
    return 0;
}

```

Файл array.c:

```

#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include "fun.h"

void func(void) {
    srand(time(0));
    int i, arr[20];
    for (i = 0; i < 20; i++) {
        arr[i] = rand() % 101 - 50;
        if (arr[i] % 2 == 0) {
            int temp = arr[i];
            arr[i] = temp * temp;
        }
    }
    for(i = 0; i < 20; i++) {
        printf("%d ", arr[i]);
    }
}

```

Файл matrix.c:

```

#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include "fun.h"

void func(void) {
    srand(time(0));
    int i, j, mat[5][6];
    for (i = 0; i < 5; i++) {
        for (j = 0; j < 6; j++) {
            mat[i][j] = rand() % 101 - 50;
            if (mat[i][j] % 2 == 0) {
                int temp = mat[i][j];
                mat[i][j] = temp * temp;
            }
        }
    }

    for (i = 0; i < 5; i++) {
        for (j = 0; j < 6; j++) {
            printf("%d ", mat[i][j]);
        }
        printf("\n");
    }
}

```

Файл load.c:

```

#include "load.h"
#include <stdio.h>
#include <windows.h>
#include "fun.h"

void LoadRun(const char * const s) {
    void * lib;
    void (*fun)(void);
    lib = LoadLibrary(s); //загрузка библиотеки в память;
    if (!lib) {

```

```

        printf("cannot open library '%s'\n", s);
        return;
    }
    fun = (void (*)(void))GetProcAddress((HINSTANCE)lib, "func"); //получение
    указателя на функцию из библиотеки;
    if (fun == NULL) {
        printf("cannot load function func\n");
    } else {
        fun();
    }
    FreeLibrary((HINSTANCE)lib); //выгрузка библиотеки;
}

```

Результат работы программы представлен на рисунке 3.1.

```

C:\Users\ASUS>C:\stud\CP3\main.exe
Choose library:
1-first.
2-second,
3-quit
1
-45 16 -47 484 47 1156 1764 35 100 23 1024 25 3 1156 -3 23 -11 1444 -45 35
2
-39 37 256 41 784 1764
484 39 1936 49 39 144
-1 2116 -45 13 7 -21
-19 676 35 -37 2304 784
-41 45 36 21 47 -23
3

```

Рисунок 3.1 – Результат работы программы

Файлы *.c скомпилируются в объектные файлы, при этом используется параметр -fPIC (рисунок 3.2, 3.3).

```

C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -c -o C:\stud\CP3\main.o -fPIC C:\stud\CP3\main.c
C:\stud\CP3\main.c:1:0: warning: -fPIC ignored for target (all code is position independent)
#include "load.h"
^

C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -c -o C:\stud\CP3\array.o -fPIC C:\stud\CP3\array.c
C:\stud\CP3\array.c:1:0: warning: -fPIC ignored for target (all code is position independent)
#include <stdio.h>
^

C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -c -o C:\stud\CP3\matrix.o -fPIC C:\stud\CP3\matrix.c

```

Рисунок 3.2 – Компиляция, часть 1

```

C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -c -o C:\stud\CP3\load.o -fPIC C:\stud\CP3\load.c

```

Рисунок 3.3 – Компиляция, часть 2

Создаются динамические библиотеки для массива и матрицы с помощью параметра -shared (рисунок 3.4, 3.5) [3].

```
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -shared -o C:\stud\CP3\lib1.dll C:\stud\CP3\array.o  
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc -shared -o C:\stud\CP3\lib2.dll C:\stud\CP3\matrix.o
```

Рисунок 3.4 – Создание библиотек, часть 1

```
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc.exe -shared -o C:\stud\CP3\load.dll C:\stud\CP3\load.o
```

Рисунок 3.5 – Создание библиотек, часть 2

Создается исполняемый файл с использованием динамических библиотек (рисунок 3.6).

```
C:\Program Files (x86)\Dev-Cpp\MinGW64\bin>gcc.exe -o C:\stud\CP3\main.exe C:\stud\CP3\main.o -L./-l C:\stud\CP3\load.dll
```

Рисунок 3.6 – Создание исполняемого файла

4 Практическая работа №4

Тема: особенности работы с программой управления компиляцией.

Цель работы: научиться работать с программой для управления компиляцией.

Задание к практической работе: написать makefile для всех программ, написанных в предыдущих практических работах с возможностью выбора требуемой в зависимости от введенной цели. Учесть, что файлы с исходным текстом программы будут лежать в подкаталогах, в которые также должны быть помещены полученные объектные и исполняемые файлы [4].

Текст программы:

Файл Makefile:

```
all: CP1.1 CP1.2 CP2.1 CP2.2 CP2.3 CP3

CP1.1:
    gcc -E -o C:\stud\CP4\CP1.1\CP1.i C:\stud\CP4\CP1.1\CP1.c
    gcc -S -o C:\stud\CP4\CP1.1\CP1.s C:\stud\CP4\CP1.1\CP1.c

    gcc -S -O1 -o C:\stud\CP4\CP1.1\CP1_optim1.s
C:\stud\CP4\CP1.1\CP1.c
    gcc -S -O2 -o C:\stud\CP4\CP1.1\CP1_optim2.s
C:\stud\CP4\CP1.1\CP1.c
    gcc -S -O3 -o C:\stud\CP4\CP1.1\CP1_optim3.s
C:\stud\CP4\CP1.1\CP1.c

    gcc -o C:\stud\CP4\CP1.1\CP1.exe C:\stud\CP4\CP1.1\CP1.c

    @echo All files are compiled

CP1.2:
    gcc -c -o C:\stud\CP4\CP1.2\CP2.o C:\stud\CP4\CP1.2\CP2.c
    gcc -c -o C:\stud\CP4\CP1.2\func_exec.o
C:\stud\CP4\CP1.2\func_exec.c
    gcc -c -o C:\stud\CP4\CP1.2\func_in_out.o
C:\stud\CP4\CP1.2\func_in_out.c
    gcc -o C:\stud\CP4\CP1.2\main.exe C:\stud\CP4\CP1.2\CP2.o
C:\stud\CP4\CP1.2\func_exec.o C:\stud\CP4\CP1.2\func_in_out.o

    @echo All files are compiled

CP2.1:
    gcc -g -o C:\stud\CP4\CP2.1\main.exe C:\stud\CP4\CP2.1\main.c

    @echo All files are compiled

CP2.2:
    gcc -c -o C:\stud\CP4\CP2.2\func_in_out.o
C:\stud\CP4\CP2.2\func_in_out.c
    gcc -c -o C:\stud\CP4\CP2.2\main.o C:\stud\CP4\CP2.2\main.c
    gcc -o C:\stud\CP4\CP2.2\main.exe C:\stud\CP4\CP2.2\main.o
C:\stud\CP4\CP2.2\func_in_out.o

    @echo All files are compiled

CP2.3:
    gcc -c C:\stud\CP4\CP2.3\func_in_out.c
```

```

ar crs libfunc_in_out.a C:\stud\CP4\CP2.3\func_in_out.o
gcc -c -o C:\stud\CP4\CP2.3\main.o C:\stud\CP4\CP2.3\main.c
gcc -o C:\stud\CP4\CP2.3\main.exe -g C:\stud\CP4\CP2.3\main.c -L.
-lfunc_in_out

@echo All files are compiled
CP3:
gcc -c -o C:\stud\CP4\CP3\main.o -fPIC C:\stud\CP4\CP3\main.c
gcc -c -o C:\stud\CP4\CP3\array.o -fPIC C:\stud\CP4\CP3\array.c
gcc -c -o C:\stud\CP4\CP3\matrix.o -fPIC C:\stud\CP4\CP3\matrix.c
gcc -c -o C:\stud\CP4\CP3\load.o -fPIC C:\stud\CP4\CP3\load.c

gcc -shared -o C:\stud\CP4\CP3\lib1.dll C:\stud\CP4\CP3\array.o
gcc -shared -o C:\stud\CP4\CP3\lib2.dll C:\stud\CP4\CP3\matrix.o
gcc.exe -shared -o C:\stud\CP4\CP3\load.dll C:\stud\CP4\CP3\load.o

gcc.exe -c -fPIC C:\stud\CP4\CP3\main.c
gcc.exe -o C:\stud\CP4\CP3\main.exe C:\stud\CP4\CP3\main.o -L./-l
C:\stud\CP4\CP3\load.dll

```

@echo All files are compiled

Компиляция с помощью makefile показана на рисунке 4.1.

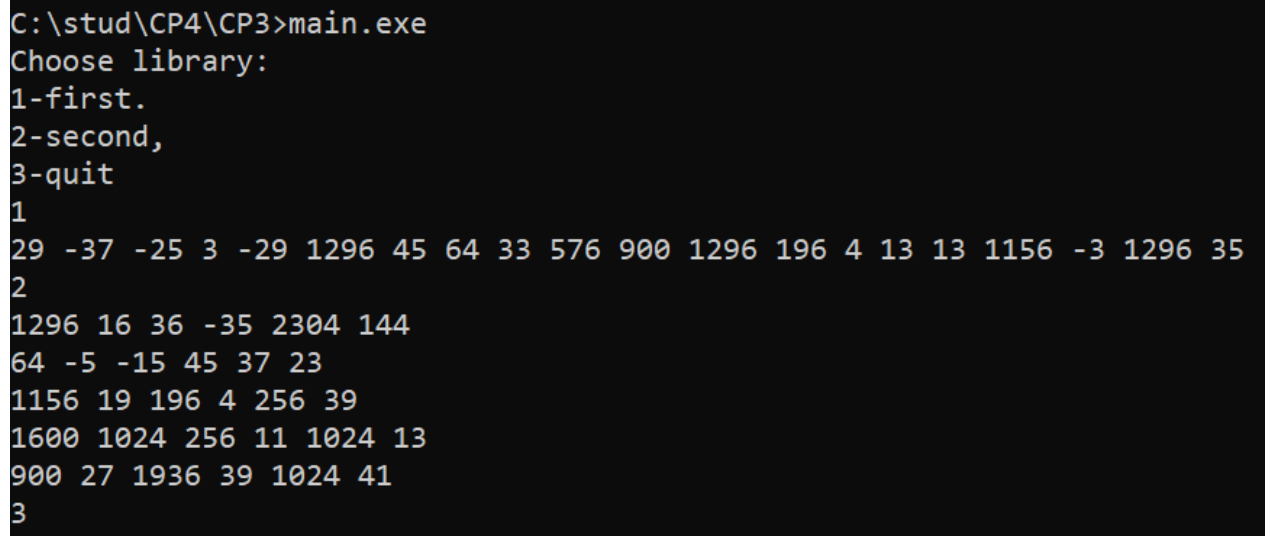
```

C:\stud\CP4\CP1.1>mingw32-make -f Makefile.txt
gcc -E -o C:\stud\CP4\CP1.1\CP1.i C:\stud\CP4\CP1.1\CP1.c
gcc -S -o C:\stud\CP4\CP1.1\CP1.s C:\stud\CP4\CP1.1\CP1.c
gcc -S -O1 -o C:\stud\CP4\CP1.1\CP1_optim1.s C:\stud\CP4\CP1.1\CP1.c
gcc -S -O2 -o C:\stud\CP4\CP1.1\CP1_optim2.s C:\stud\CP4\CP1.1\CP1.c
gcc -S -O3 -o C:\stud\CP4\CP1.1\CP1_optim3.s C:\stud\CP4\CP1.1\CP1.c
gcc -o C:\stud\CP4\CP1.1\CP1.exe C:\stud\CP4\CP1.1\CP1.c
All files are compiled
gcc -c -o C:\stud\CP4\CP1.2\CP2.o C:\stud\CP4\CP1.2\CP2.c
gcc -c -o C:\stud\CP4\CP1.2\func_exec.o C:\stud\CP4\CP1.2\func_exec.c
gcc -c -o C:\stud\CP4\CP1.2\func_in_out.o C:\stud\CP4\CP1.2\func_in_out.c
C:\stud\CP4\CP1.2\func_in_out.c: In function 'func_out_list':
C:\stud\CP4\CP1.2\func_in_out.c:15:2: warning: incompatible implicit declaration of built-in function 'printf'
printf("%d ", a[i]);
^
gcc -o C:\stud\CP4\CP1.2\main.exe C:\stud\CP4\CP1.2\CP2.o C:\stud\CP4\CP1.2\func_exec.o C:\stud\CP4\CP1.2\func_in_out.o
All files are compiled
gcc -g -o C:\stud\CP4\CP2.1\main.exe C:\stud\CP4\CP2.1\main.c
All files are compiled
gcc -c -o C:\stud\CP4\CP2.2\func_in_out.o C:\stud\CP4\CP2.2\func_in_out.c
gcc -c -o C:\stud\CP4\CP2.2\main.o C:\stud\CP4\CP2.2\main.c
gcc -o C:\stud\CP4\CP2.2\main.exe C:\stud\CP4\CP2.2\main.o C:\stud\CP4\CP2.2\func_in_out.o
All files are compiled
gcc -c C:\stud\CP4\CP2.3\func_in_out.c
ar crs libfunc_in_out.a C:\stud\CP4\CP2.3\func_in_out.o
gcc -c -o C:\stud\CP4\CP2.3\main.o C:\stud\CP4\CP2.3\main.c
gcc -o C:\stud\CP4\CP2.3\main.exe -g C:\stud\CP4\CP2.3\main.c -L. -lfunc_in_out
All files are compiled
gcc -c -o C:\stud\CP4\CP3\main.o -fPIC C:\stud\CP4\CP3\main.c
C:\stud\CP4\CP3\main.c:1:0: warning: -fPIC ignored for target (all code is position independent)
#include "load.h"
^
gcc -c -o C:\stud\CP4\CP3\array.o -fPIC C:\stud\CP4\CP3\array.c
C:\stud\CP4\CP3\array.c:1:0: warning: -fPIC ignored for target (all code is position independent)
#include <stdio.h>
^
gcc -c -o C:\stud\CP4\CP3\matrix.o -fPIC C:\stud\CP4\CP3\matrix.c
C:\stud\CP4\CP3\matrix.c:1:0: warning: -fPIC ignored for target (all code is position independent)
#include <stdio.h>
^
gcc -c -o C:\stud\CP4\CP3\load.o -fPIC C:\stud\CP4\CP3\load.c
C:\stud\CP4\CP3\load.c:1:0: warning: -fPIC ignored for target (all code is position independent)
#include "load.h"
^
gcc -shared -o C:\stud\CP4\CP3\lib1.dll C:\stud\CP4\CP3\array.o
gcc -shared -o C:\stud\CP4\CP3\lib2.dll C:\stud\CP4\CP3\matrix.o
gcc.exe -shared -o C:\stud\CP4\CP3\load.dll C:\stud\CP4\CP3\load.o
gcc.exe -c -fPIC C:\stud\CP4\CP3\main.c
C:\stud\CP4\CP3\main.c:1:0: warning: -fPIC ignored for target (all code is position independent)
#include "load.h"
^
gcc.exe -o C:\stud\CP4\CP3\main.exe C:\stud\CP4\CP3\main.o -L./-l C:\stud\CP4\CP3\load.dll
All files are compiled

```

Рисунок 4.1 – Компиляция

Запуск программы, скомпилированной при помощи makefile, показана на рисунке 4.2.



```
C:\stud\CP4\CP3>main.exe
Choose library:
1-first.
2-second,
3-quit
1
29 -37 -25 3 -29 1296 45 64 33 576 900 1296 196 4 13 13 1156 -3 1296 35
2
1296 16 36 -35 2304 144
64 -5 -15 45 37 23
1156 19 196 4 256 39
1600 1024 256 11 1024 13
900 27 1936 39 1024 41
3
```

Рисунок 4.2 – Запуск программы

5 Практическая работа №5

Тема: особенности использования систем контроля версий.

Цель работы: овладеть навыками работы с системой контроля версий.

Для выполнения первой части работы необходимо выполнить следующие семь задач:

1. Используя систему контроля версий git, создать локальный репозиторий для программы, написанной в третьей работе.
2. Используя команду commit, поместить текущее состояние файлов с исходными текстами программы и библиотек в созданный репозиторий.
3. Создать две ветви программы в репозитории: в одной убрать из программы поддержку ОС семейства Windows, в другой ОС семейства Linux.
4. Сравнить ветви и показать найденные конфликты.
5. Просмотреть историю изменений.
6. Рассмотреть команды.
7. Восстановить изначальную версию программы из репозитория.

На рисунке 5.1 показано создание локального репозитория.

```
C:\Users\ASUS>cd C:\stud\CP3
C:\stud\CP3>git status
fatal: not a git repository (or any of the parent directories): .git
C:\stud\CP3>git init
Initialized empty Git repository in C:/stud/CP3/.git/
C:\stud\CP3>git add .
warning: in the working copy of 'CMakeLists.txt', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'cmake-build-debug/.ninja_log', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'cmake-build-debug/CMakeFiles/clion-environment.txt', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'cmake-build-debug/CMakeFiles/clion-log.txt', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'main.c', LF will be replaced by CRLF the next time Git touches it
C:\stud\CP3>git commit -m "Initial commit"
[master (root-commit) cef7139] Initial commit
53 files changed, 3580 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/CP3.iml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 CMakeLists.txt
create mode 100644 array.c
create mode 100644 array.o
```

Рисунок 5.1 – Создание репозитория

Команда git status показывает состояние файлов в текущем репозитории.

Команда git init инициализирует репозиторий для текущей директории.

Команда git add . добавляет все файлы из текущей директории в репозиторий.

Команда git commit сохраняет данные во внутреннюю базу данных.

На рисунке 5.2 показано создание двух веток в репозитории, добавление изменений и сравнение веток.

```
C:\stud\CP3>git branch -m master windows

C:\stud\CP3>git checkout -b linux
Switched to a new branch 'linux'

C:\stud\CP3>git branch
* linux
  windows

C:\stud\CP3>git add main.c
warning: in the working copy of 'main.c', LF will be replaced by CRLF the next time Git touches it

C:\stud\CP3>git commit -m "Linux supporting libraries"
[linux 28246ea] Linux supporting libraries
1 file changed, 1 insertion(+)

C:\stud\CP3>git checkout windows
Switched to branch 'windows'

C:\stud\CP3>git branch
  linux
* windows

C:\stud\CP3>git add main.c

C:\stud\CP3>git commit -m "Windows supoorting libraries"
[windows 180f421] Windows supoorting libraries
1 file changed, 1 insertion(+)

C:\stud\CP3>git diff linux
diff --git a/main.c b/main.c
index 2d86afa..bb98f8e 100644
--- a/main.c
+++ b/main.c
@@ -4,7 +4,7 @@
 int main(void)
 {
     int a=0,b=1;
-    printf("Linux OC");
+    printf("Windows OC");
     printf("Choose library:\n1-first.\n2-second,\n3-quit\n");
     while(b)
     {
```

Рисунок 5.2 – Создание веток, их изменение и сравнение

Команда git branch перечисляет ветки, создает новые, переименовывает и удаляет их.

Команда git diff показывает разницу между git деревьями.

На рисунке 5.3 показана история изменений репозитория.

```

C:\stud\CP3>git log
commit 180f4216c44958893aeb0998946634eefc6a280 (HEAD -> windows)
Author: Egor <kogayegor@gmail.com>
Date:   Sun Oct 23 15:53:03 2022 +0300

    Windows supoorting libraries

commit cef71390a9d07724e9e6acd508cb82621e6cb350
Author: Egor <kogayegor@gmail.com>
Date:   Sun Oct 23 15:47:39 2022 +0300

    Initial commit

C:\stud\CP3>git checkout linux
Switched to branch 'linux'

C:\stud\CP3>git log
commit 28246ea037d9a27511b79d7c1a2865683fb62b07 (HEAD -> linux)
Author: Egor <kogayegor@gmail.com>
Date:   Sun Oct 23 15:51:38 2022 +0300

    Linux supporting libraries

commit cef71390a9d07724e9e6acd508cb82621e6cb350
Author: Egor <kogayegor@gmail.com>
Date:   Sun Oct 23 15:47:39 2022 +0300

    Initial commit

```

Рисунок 5.3 – История изменений

Команда git log показывает историю изменений.

На рисунке 5.4 показано восстановление изначальной версии программы из репозитория.

```

C:\stud\CP3>git reset --hard HEAD~1
HEAD is now at cef7139 Initial commit

C:\stud\CP3>git diff windows
diff --git a/main.c b/main.c
index bb98f8e..104c521 100644
--- a/main.c
+++ b/main.c
@@ -4,7 +4,6 @@
 int main(void)
 {
     int a=0,b=1;
-    printf("Windows OC");
     printf("Choose library:\n1-first.\n2-second,\n3-quit\n");
     while(b)
     {

```

Рисунок 5.4 – Восстановление изначальной версий

Команда git reset используется для отмены изменений.

Для выполнения второй части работы необходимо выполнить следующие три задачи:

1. Самостоятельно изучить различия основных систем контроля версий – CVS, SVN, Git.
2. Рассмотреть доступные глобальные репозитории для любой из систем контроля версий.
3. Описать процесс регистрации и работы с одним из рассмотренных глобальных репозиториях.

CVS появилась в 1980-х и до сих пор популярна как у разработчиков коммерческих продуктов, так и у open-source разработчиков [5].

Преимущества: испытанная временем технология, которая удерживается на рынке десятки лет.

Недостатки:

- переименование или перемещение файлов не отражается в истории;
- риски безопасности, связанные с символическими ссылками на файлы;
- нет поддержки атомарных операций, что может привести к повреждению кода;
- операции с ветками программного кода дорогостоящие, так как эта система контроля не предназначена для долгосрочных проектов с ветками кода.

SVN создавалась как альтернатива CVS с целью исправить недостатки CVS и в то же время обеспечить высокую совместимость с ней.

Преимущества:

- система на основе CVS;
- допускает атомарные операции;
- операции с ветвлением кода менее затратны;
- широкий выбор плагинов IDE;

- не использует пиринговую модель.

Недостатки:

- все еще сохраняются ошибки, связанные с переименованием файлов и директорий;
- неудовлетворительный набор команд для работы с репозиторием;
- сравнительно небольшая скорость.

Git. Эта система была создана для управления разработкой ядра Linux и использует подход, который в корне отличается от CVS и SVN [5].

Преимущества:

- прекрасно подходит для тех, кто ненавидит CVS/SVN;
- значительное увеличение быстродействия;
- дешевые операции с ветками кода;
- полная история разработки доступная оффлайн и онлайн;
- распределенная, пиринговая модель.

Недостатки:


- высокий порог вхождения (обучения) для тех, кто ранее использовал SVN;
- ограниченная поддержка Windows (по сравнению с Linux).

На рисунке 5.5 показано создание глобального репозитория в системе GitHub.

Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner * **Repository name ***

 KogaiEgor ▾ /

Great repository names are short and memorable. Need inspiration? How about [redesigned-goggles?](#)

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.


☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

 You are creating a public repository in your personal account.

[Create repository](#)

Рисунок 5.5 – Создание глобального репозитория

На рисунке 5.6 показан перенос локального репозитория в глобальный.

```

C:\stud\CP3>git remote add origin https://github.com/KogaiEgor/practice.git
error: remote origin already exists.

C:\stud\CP3>git remote set-url origin git@github.com:KogaiEgor/practice.git

C:\stud\CP3>git branch
* linux
  windows

C:\stud\CP3>git push -u origin linux
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

C:\stud\CP3>git push -u origin linux
Enumerating objects: 65, done.
Counting objects: 100% (65/65), done.
Delta compression using up to 12 threads
Compressing objects: 100% (60/60), done.
Writing objects: 100% (65/65), 134.27 KiB | 928.00 KiB/s, done.
Total 65 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (6/6), done.
To github.com:KogaiEgor/practice.git
 * [new branch]      linux -> linux
branch 'linux' set up to track 'origin/linux'.

C:\stud\CP3>git push -u origin windows
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'windows' on GitHub by visiting:
remote:   https://github.com/KogaiEgor/practice/pull/new/windows
remote:
To github.com:KogaiEgor/practice.git
 * [new branch]      windows -> windows
branch 'windows' set up to track 'origin/windows'.

```

Рисунок 5.6 – Перенос репозитория

ЗАКЛЮЧЕНИЕ

В ходе выполнения учебной практики были изучены:

- особенности работы с набором компиляторов и утилит GNU Compiler Collection;
- особенности использования отладчика GDB;
- процессы создания динамических библиотек при помощи набора компиляторов и утилит GCC и особенности их применения;
- особенности работы с программой управления компиляцией на примере утилиты make;
- особенности различных систем контроля версий.

Все поставленные задачи учебной практики выполнены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Скулябина О.В. Компьютерный практикум. – СПб.: БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова, 2021. – 12 с.
2. Шпаргалка полезных команд GDB. – URL: <https://habr.com/ru/post/535960/> (дата обращения 18.09.2022).
3. Static and Dynamic libraries. – URL: <https://www.geeksforgeeks.org/static-vs-dynamic-libraries/> (дата обращения 25.09.2022).
4. GNU make. – URL: <https://www.gnu.org/software/make/manual/make.html> (дата обращения 3.10.2022).
5. Система контроля версий (cvc) – Сравниваем: GIT, SVN, Mercurial. – URL: <https://biz30.timedoctor.com/ru/система-контроля-версий/?ysclid=l9wp7gut4z523431172> (дата обращения 18.10.2022).