Национальный исследовательский университет “Высшая школа экономики”.

Факультет компьютерных наук. Программная инженерия.

Архитектура вычислительных систем.

Домашнее задание №2 студента группы БПИ213 Абрамова Александра Сергеевича.

Для удобства переименовал файл с сортировкой Шелла в `2.c`.

1. На первый взгляд явных логических ошибок или ошибок во время выполнения я не обнаружил. Тем не менее запуск программы показывает, что что-то явно не так: на входных данных `5 4 3 2 1` программа выводит `0 1 2 3 4`.

   Для этого программа была собрана с помощью утилиты `gcc` командой `gcc 2.c` и запущена стандартным методом ОС Linux (Debian 10) с передачей входного массива с помощью параметров командной строки: `./a.out 5 4 3 2 1`

```
debian@vps-d19579e5:~/as$ gcc 2.c
debian@vps-d19579e5:~/as$ ./a.out 5 4 3 2 1
Output: 0 1 2 3 4 N
```

2. При входных данных `2 1 5 -12 3 -9` программа работает корректно:

```
debian@vps-d19579e5:~/as$ ./a.out 2 1 5 -12 3 -9
Output: -12 -9 1 2 3 5 N
```

3. Для отслеживания ошибки воспользуемся утилитой `gdb`. Для этого пересоберём программу с отладочными символами (`gcc -g 2.c`) и запустим её через отладчик (`gdb ./a.out`).

```
debian@vps-d19579e5:~/as$ gcc -g 2.c
debian@vps-d19579e5:~/as$ gdb ./a.out
GNU gdb (Debian 8.2.1-2+b3) 8.2.1
Copyright (C) 2018 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-linux-gnu".
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 ./a.out...done.
(gdb)
```

a. Под подозрением строки 19 и 22 - единственные, в которых происходит изменение элементов массива. Для отслеживания того, что происходит при их выполнении, установим точку остановки на начало функции `shell_sort` с помощью команды `break shell_sort`. Для отладки я буду использовать входные данные `3 2 1`: этот тест аналогичен приведённому в п.2, но чуть короче, из-за чего с помощью него удобнее отлаживать программу.

```
(gdb) break shell_sort
Breakpoint 1 at 0x1180: file 2.c, line 8.
(gdb)
```

b. С помощью команды `run 3 2 1` начнём выполнение. Для наблюдения за значениями воспользуемся `display` и закрепим все элементы массива, а также переменные `i` и `j` для отслеживания итерации, на которой происходит сбой:

```
display a[0]
display a[1]
display a[2]
display i
display j
```

Заметим, что значения `a[0], a[1], a[2]` верны, то есть ошибки в алгоритме ввода не произошло. Переменные `i, j` не были инициализированы, поэтому содержат произвольные значения.

```
(gdb) run 3 2 1
Starting program: /home/debian/as/a.out 3 2 1

Breakpoint 1, shell_sort (a=0x555555559260, size=3) at 2.c:8
8         int h = 1;
(gdb)
(gdb)
(gdb) display a[0]
1: a[0] = 3
(gdb) display a[1]
2: a[1] = 2
(gdb) display a[2]
3: a[2] = 1
(gdb) display i
4: i = 32767
(gdb) display j
5: j = -136083184
(gdb)
```

c. С помощью команд `step` и `until` выполним программу построчно. При первом объявлении переменной `v` также закрепим её с помощью `display v`.

```
(gdb) step
11          h = h * 3 + 1;
1: a[0] = 3
2: a[1] = 2
3: a[2] = 1
4: i = 32767
5: j = -136083184
(gdb) step
12        } while (h <= size);
1: a[0] = 3
2: a[1] = 2
3: a[2] = 1
4: i = 32767
5: j = -136083184
(gdb) until
15          h /= 3;
1: a[0] = 3
2: a[1] = 2
3: a[2] = 1
4: i = 32767
5: j = -136083184
(gdb) step
16          for (i = h; i < size; i++) {
1: a[0] = 3
2: a[1] = 2
3: a[2] = 1
4: i = 32767
5: j = -136083184
(gdb) step
17            int v = a[i];
1: a[0] = 3
2: a[1] = 2
3: a[2] = 1
4: i = 1
5: j = -136083184
(gdb) display v
6: v = 0
(gdb)
```

d. Заметим, что на четвёртой итерации (при `i = 3`) переменная `v` почему-то принимает значение `0`, то есть в этот момент `a[i] = 0`, что эквивалентно `a[3] = 0`. Но `a[3]` не существует: введённый массив содержит всего `3` элемента, то есть последний корректный индекс – `2`.

```
(gdb) step
17            int v = a[i];
1: a[0] = 1
2: a[1] = 2
3: a[2] = 3
4: i = 3
5: j = 0
6: v = 1
(gdb) step
18            for (j = i; j >= h && a[j - h] > v; j -= h) {
1: a[0] = 1
2: a[1] = 2
3: a[2] = 3
4: i = 3
5: j = 0
6: v = 0
(gdb)
```

e. Так как `i` принимает значения от `0` до `size`, проверим значение переменной `size` с помощью `print size` и заметим, что оно действительно почему-то равно `4`, хотя длина введённого массива – `3`. Ошибка обнаружена.

```
(gdb) print size
$1 = 4
(gdb)
```

f. Единственное место, где переменная `size` получает или изменяет значение – при вызове функции. Скорее всего, в функцию `shell_sort` было передано неверное значение. Перезапустим программу и посмотрим значение переменной `argc` (второй аргумент функции) перед вызовом.

g. Для этого отследим значение переменной `argc` в функции `main`. Установим точку останова в начале (`break main`), запустим программу (`run 3 2 1`) и закрепим значение переменной `argc` (`display argc`).

```
(gdb) q
A debugging session is active.

        Inferior 1 [process 27839] will be killed.

Quit anyway? (y or n) y
debian@vps-d19579e5:~/as$ gdb ./a.out
GNU gdb (Debian 8.2.1-2+b3) 8.2.1
Copyright (C) 2018 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-linux-gnu".
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 ./a.out...done.
(gdb) break main
Breakpoint 1 at 0x128c: file 2.c, line 32.
(gdb) run 3 2 1
Starting program: /home/debian/as/a.out 3 2 1

Breakpoint 1, main (argc=4, argv=0x7fffffffe588) at 2.c:32
32          a = (int *)malloc((argc - 1) * sizeof(int));
(gdb) display argc
1: argc = 4
(gdb)
```

h. Действительно, значение переменной в начале программы – `4`, которое без изменений передаётся в `shell_sort`.

i. На самом деле, в командной строке мы писали 4 «слова», но лишь 3 из них – наш массив. Обратим внимание, что при выделении памяти и её заполнении учитывается, что первый параметр командной строки, как и ожидается, – не часть массива. Значит, для исправления ошибки необходимо в функцию `shell_sort` передавать не `argc`, а `argc - 1`.

j. Исправим это, пересоберём и запустим программу, после чего получим верный ответ на рассмотренный тест, а также несколько других тестов, по чему можно судить о том, что теперь всё работает корректно. Таким образом, нужно было изменить `36` строку на

```
shell_sort(a, argc - 1);
```