Программирование

Факультет безопасности информационных технологий
Университет ИТМО

Осень 2023

Лабораторная работа № 1
Поразрядные и логические операции

Разработать на языке C для ОС Linux программу, которая получает целое число указанного типа (Табл. 1), выводит его двоичное представление на экран (необходимо выводить все разряды числа, байты числа разделяются пробелами), выполняет преобразование в соответствии с вариантом (Табл. 2), затем выводит на экран двоичное представление результата преобразования.

Номер варианта лабораторной работы составлен из двух чисел (X-Y), где X — номер варианта из Табл. 1, а Y — номер варианта из Табл.2.

Программа должна представлять собой консольное приложение, настройка работы которого осуществляется путем передачи аргументов в строке запуска:

```
lab1abcNXXXXX [число]
```

Квадратные скобки здесь означают, что число — необязательный аргумент и может отсутствовать.

Имя программы должно начинаться на lab1, далее должен следовать уникальный для варианта суффикс. Уникальный суффикс составляется из первых букв имени, отчества (если есть) и фамилии студента, выполняющего лабораторную работу. Далее следует номер группы студента. Используются строчные латинские буквы и арабские (в традиционном понимании, т. е. 0..9) цифры. Например, если студента, выполняющего лабораторную, зовут Петр Сергеевич Иванов, его группа — N32451, то имя программы должно быть `lab1psiN32451`.

При запуске программы пользователь может указать число в десятичной системе счисления, в этом случае преобразование осуществляется над указанным числом. Например (тип числа — short, преобразование — инверсия всех битов):

```
$ ./lab1psiN32451 128
00000000 10000000
11111111 01111111
```

Если пользователь при запуске указывает вместо числа нужного формата строку, которая не является числом, программа должна сообщить об ошибке:

```
$ ./lab1psiN32451 щито
Ошибка: 'щито' не является числом.

$ ./lab1psiN32451 220v
Ошибка: '220v' не является числом.
```

Если пользователь не указывает число при запуске, программа должна получить случайное число с помощью функций стандартной библиотеки srand() и rand(). Например:

```
$ ./lab1psiN32451
00000001 01011001
11111110 10100110
```

Проект (исходные коды, заголовочные файлы, Makefile и прочие файлы, которые могут понадобиться для сборки) должен содержаться в отдельном каталоге с именем,

совпадающим с названием программы (lab1abcNXXXXX), и собираться с помощью стандартной утилиты make. Исходные файлы программы на языке C должны компилироваться с помощью gcc. Makefile должен поддерживать как минимум цели all и clean. Пример заготовки проекта ЛР № 1 содержится на гугл-диске в папке "лабораторные" (архив lab1abcNXXXXX.tar.gz, для распаковки можно использовать команду `tar -xzvf lab1abcNXXXXX.tar.gz`)

Порядок выполнения и сдачи лабораторной работы:

1. Скачать заготовку проекта, изменить название каталога на правильное (соответствующее вашей группе и ФИО), скорректировать содержимое Makefile'a.
2. Выполнить задание, подготовить все файлы проекта, скомпилировать программу и библиотеку с флагами `-Wall -Wextra -Werror` и устранить все предупреждения и ошибки.
3. Протестировать программу на различных входных данных, убедиться, что ошибок нет, в противном случае вернуться к пункту 2.
4. Удалить все исполняемые и промежуточные файлы из папки проекта (`make clean`). В архиве должны остаться только файлы *.c, *.h, Makefile, README.txt.
5. Заархивировать папку проекта, используя формат tar.gz. (`tar -czvf lab1abcNXXXXX.tar.gz lab1abcNXXXXX/`).
6. Подготовить отчет по лабораторной работе в формате pdf, на титульной странице отчета не забыть поставить подпись и указать номер варианта. Файл отчета должен иметь название NXXXXX_ФамилияИО_ЛР1.pdf. Состав отчета описан ниже.
7. Отправить архив и отчет в формате pdf на почту преподавателя, который ведет лабораторные, письмом с темой «*Программирование ЛР1 Фамилия Имя Отчество NXXXXX Вариант X-Y*».
8. Дождаться ответа по почте или на лабораторном занятии, устранить возможные замечания (повторить с пункта 1).
9. Получить некоторое количество вопросов от преподавателя по отчету и темам, связанным с лабораторной, и дать на них ответы (а может и не получить, если лабораторная выполнена на хорошем уровне и сомнений в знаниях студента у преподавателя не возникает). Получить от преподавателя подтверждение, что работа выполнена успешно и отчет принят.
10. Немного расслабиться и приступить к следующей лабораторной работе :-)

Отчет должен быть подготовлен в формате pdf и содержать:

- правильно оформленную титульную страницу (с подписью студента);
- задание;
- Make-файл;
- примеры работы программ на различных исходных данных (скриншоты);
- блок-схему алгоритма преобразования (нужна только в ЛР № 1);
- исходный текст програмы с комментариями.

Замечание 1. При выполнении лабораторной работы следует использовать только функции стандартной библиотеки C и системные вызовы операционной системы. Использовать C++, ввод-вывод в стиле C++ (классы ifstream/ofstream/…), контейнеры и алгоритмы STL (<string>, <vector>, <map>, ...) **запрещено**.

Замечание 2. Преобразование должно осуществляться **только** с помощью арифметических, логических и побитовых операций. Использование стандартных массивов и VLA, битовых полей структур и объединений или контейнеров STL для представления битов и байтов числа в заданном вариантом задания алгоритме **запрещено**.

Замечание 3. В программе должна присутствовать обработка ошибок: в случаях, если пользователь передал некорректные аргументы, программа должна выдавать диагностическое сообщение на консоль (в стандартный поток ошибок), прежде чем завершиться.

Замечание 4. В результате преобразования должно измениться исходное число (переменная), недостаточно просто вывести результат преобразования на экран.

Замечание 5. Если операция предусматривает использование одного или нескольких случайных параметров (например, сдвиг на случайное число битов), их значения также следует выводить на экран.

Замечание 6. Если для облегчения понимания того, как выполняется преобразование, нужно выводить дополнительную информацию, её желательно выводить с помощью переменной окружения LAB1DEBUG (см. примеры на гугл-диске).

Замечание 7. Для преобразования строки в число рекомендуется использовать функцию sscanf() с правильно выбранным спецификатором формата и проверкой возвращаемого значения этой функции и переменной errno.

Замечание 8. Для подготовки блок-схемы рекомендуется использовать сервис https://draw.io.

Замечание 9. Программа должна успешно компилироваться и выполняться в 64-разрядной ОС Linux с ядром версии >= 5.0, glibc версии >= 2.0, gcc версии >= 10.0.

Таблица 1. Тип числа, над которым требуется выполнять преобразование

№ варианта (X)	Тип числа
1	unsigned short
2	unsigned int
3	unsigned long
4	unsigned long long
5	uint16_t
6	uint32_t
7	uint64_t

Таблица 2. Преобразование числа

№ варианта (Y)	Задание	Пример преобразования
1	Изменить порядок следования битов в числе на обратный.	11010011 → 11001011 *
2	Изменить порядок следования нечетных битов в числе на обратный.	11110000 → 01011010 *
3	Сдвинуть биты в каждом байте циклически вправо на случайное число N из диапазона 0..7.	0xDEADBEEF → 0xB76BAFFB (N = 2)
4	Если в числе встречается последовательность битов 11011, то заменить её на 01110. В преобразованном числе исходная последовательность встречаться не должна.	0xDEADBEEF → 0x5CACBAAE
5	Каждую младшую тетраду каждого байта сдвинуть циклически влево на число, содержащееся в двух старших битах старшей тетрады.	11010011 → 11011001 *
6	Старшую тетраду каждого байта числа заменить результатом операции «стрелка Пирса» старшей и младшей тетрад, а младшую тетраду – результатом операции «штрих Шеффера» старшей и младшей тетрад исходного байта.	11010011 → 00001110 *
7	Если в числе встречается последовательность битов 000, заменить её на 0110 (лишние разряды сдвигать	11010001 → 10101101 *

	влево). В преобразованном числе исходная последовательность встречаться не должна.	
8	Старшую тетраду в нечетных байтах числа заменить результатом операции «исключающее ИЛИ» старшей и младшей тетрад, а младшую тетраду в четных байтах – побитовым отрицанием результата операции «исключающее ИЛИ» старшей и младшей тетрад.	0xDEADBEEF → 0x3EA85EEE
9	Назовем сверткой байта порядка N операцию циклического сдвига старшей тетрады на N битов вправо, а младшей тетрады на N битов влево. Выполнить свертку всех байтов на случайное число из диапазона 0..3.	0xDEADBEEF → 0x7BA7EBBF (N = 2)
10	Назовем рангом байта значение трех его старших битов. Инвертировать в числе байты с самым низким рангом.	0xDEADBEEF → 0xDE5241EF
11	Назовем характеристикой байта количество единичных битов. Инвертировать в байтах с самой маленькой характеристикой старшую тетраду, а в байтах с самой большой характеристикой — младшую тетраду.	0xDEADBEEF → 0xDE5DBEE0
12	Назовем триплетом группу из трех битов. Если соседние триплеты одинаковы, инвертировать в младшем старший бит, а в старшем триплете – младший.	0xDEADBEEF → 0xDEAD7EEF
13	В четных байтах переместить нулевые биты в старшие биты, а в нечетных байтах – в младшие.	0xDEADBEEF → 0xFC1FFC7F
14	Назовем триплетом группу из трех битов. В каждом третьем триплете, начиная с младшего, изменить порядок следования битов на обратный (2,5,8,…).	0xDEADBEEF → 0xDBAF3FAF
15	Назовем симметричным байт, в котором нулевой бит имеет такое же значение, что и седьмой, а первый – такое же, что и шестой. Изменить порядок следования симметричных байтов в числе на обратный.	0xDEADBEEF → 0xDEEFBEAD
16	Назовем рангом тетрады модуль разности количества нулевых и единичных битов в ней. Сделать тетрады с самым высоким рангом старшими в числе, сохранив относительный порядок остальных тетрад.	0xDEADBEEF → 0xFDEADBEE
17	Назовем сверткой байта его арифметический сдвиг вправо на количество содержащихся в нем нулевых битов. Выполнить свертку всех байтов числа.	0xDEADBEEF → 0xF7F5EFF7
18	Назовем характеристикой байта разность между значениями младшей и старшей тетрад. В байтах с наибольшим значением характеристики инвертировать нечетные биты.	0xDEADBEEF → 0xDE0714EF
19	Назовем симметричным байт, в котором значения разрядов младшей тетрады обратны значениям разрядов старшей тетрады. Если в числе есть симметричные байты, сделать их несимметричными.	0xDEADBEE1 → 0xDEADBEE2
20	Случайным образом изменить порядок следования тетрад в числе.	0xDEADBEEF → 0xEDDEFABE
21	Назовем серединой байта его биты 2..5. Выполнить циклический свиг середины нечетных по порядку следования байтов влево, а четных – вправо на случайное число из диапазона 0..3.	1010110111 → 1010011111 * (N = 2)
22	Выполнить операцию исключающее или (XOR) самой большой по значению тетрады в числе со	0xDEADBEEF → 0xD15DB11F

	всеми тетрадами, содержащими четное число.	
23	Назовем дублетом группу из двух битов. Расположить дублеты числа в порядке возрастания.	101101001101 → 111110010100
24	Сдвинуть биты в каждом байте циклически вправо на число, содержащееся в двух старших битах байта.	0xDEADBEEF → 0xDB6BAFFD
25	Циклически сдвинуть нечетные биты в числе влево, а четные – вправо на случайное число из диапазона 0..3.	10110101 → 11000111 * (N=2)
26	Случайным образом изменить порядок следования битов в каждом байте числа.	0xDEADBEEF → 0x7B9EE7BF
27	Найти в числе непрерывную последовательность из 10 битов, имеющую наименьшее значение, и изменить в ней порядок следования битов на обратный.	0xDEADBEEF → 0xDFB53EEF
28	Назовем характеристикой байта количество нулевых битов. Инвертировать четные биты во всех байтах, кроме байтов с самой маленькой и самой большой характеристиками.	0xDEADBEEF → 0x8BADEBEF
29	Назовем рангом байта результат побитовой операции исключающее или (XOR) младшей и старшей тетрад. Инвертировать в байтах с самым низким рангом нечетные биты.	0xDEADBEEF → 0xDEADBE45
30	Назовем дублетом группу из двух битов. Заменить наиболее часто встречающийся дублет в числе на его инвертированное значение.	0xDEADBEEF → 0x12A18220
31	Расположить в каждом байте числа тетрады по возрастанию.	0xDEADBEEF → 0xEDDAEBFE
32	Найти в числе непрерывную последовательность из 6 битов, имеющую наибольшее значение, и изменить в ней порядок следования битов на обратный.	0xDEADBEEF → 0xDEAD9FEF

* – в данных примерах показано преобразование одного байта числа

Версия 0.4 от 01.11.2023

Список изменений:
0.3: Исключено требование проверки того, попадает ли введенное число в заданный диапазон.
0.4: Исправлен пример в Табл. 2, вар. 17.