(/)

0x01. C - Variables, if, else, while

C

By Julien Barbier

Weight: 1

Project over - took place from Jul 5, 2022 to Jul 6, 2022 - you're done with 200% of tasks.

An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 70.0/70 mandatory & 21.0/21 optional
- **Altogether: 200.0%**
  - Mandatory: 100.0%
  - Optional: 100.0%
  - Calculation: 100.0% + (100.0% * 100.0%) == **200.0%**

Resources

Read or watch:

- Everything you need to know to start with C.pdf (/rltoken/PkAydT3D9u5pN3nPCAlNZQ) (*You do not have to learn everything in there yet, but make sure you read it entirely first and make sure you understand the slides: "comments", "Data types | Integer types", "Declaration", "Characters", "Arithmetic operators", "Variables assignments", "Comparisons", "Logical operators", "if, if…else", "while loops"*.)
- Keywords and identifiers (/rltoken/58ThnAAxwJv5s_ceKMMPhw)
- integers (/rltoken/2sXkmDiD7BF7pNIOxMQWFA)
- Arithmetic Operators in C (/rltoken/S-b9MN2iELhSEwCl093Vzw)
- If statements in C (/rltoken/usvxrTB3ko5kGTq48p5fSA)
- if…else statement (/rltoken/CU6mSX1qdZKOhDEgmToUGA)
- Relational operators (/rltoken/O1N-qacaTC-BHXm3Dp3eUA)
- Logical operators (/rltoken/ndmvlsrk_wLgwBs-Yma9ag)
- while loop in C (/rltoken/mwx2_bj3gIFEgCqdwdTp4w)
- While loop (/rltoken/MW4Ob-6JLWt7Zn6vZ0EsBw)

man or help:

- `ascii` (*You do not need to learn about `scanf`, `getc`, `getchar`, `EOF`, `EXIT_SUCCESS`, `time`, `rand`, `srand`, `RAND_MAX`, `for` loops, `do...while` loops, functions.*)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/qBrRkO5QsdzGun5FkaMooQ), **without the help of Google**:

General

- What are the arithmetic operators and how to use them
- What are the logical operators (sometimes called boolean operators) and how to use them
- What the the relational operators and how to use them
- What values are considered TRUE and FALSE in C
- What are the boolean operators and how to use them
- How to use the `if`, `if ... else` statements
- How to use comments
- How to declare variables of types `char`, `int`, `unsigned int`
- How to assign values to variables
- How to print the values of variables of type `char`, `int`, `unsigned int` with `printf`
- How to use the `while` loop
- How to use variables with the `while` loop
- How to print variables using `printf`
- What is the `ASCII` character set
- What are the purpose of the `gcc` flags `-m32` and `-m64`

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project
- There should be no errors and no warnings during compilation
- You are not allowed to use `system`

- Your code should use the `Betty` style. It will be checked using betty-style.pl (https://github.com/holbertonschool/Betty/blob/master/betty-style.pl) and betty-doc.pl (https://github.com/holbertonschool/Betty/blob/master/betty-doc.pl)

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Hide quiz)

Question #0

What is the size of the `unsigned int` data type?

- [ ] 1 byte
- [ ] 2 bytes
- [x] 4 bytes
- [ ] 8 bytes

Question #1

What is the size of the `char` data type?

- [x] 1 byte
- [ ] 2 bytes
- [ ] 4 bytes
- [ ] 8 bytes

Question #2

What is the size of the `float` data type?

- [ ] 1 byte
- [ ] 2 bytes
- [x] 4 bytes
- [ ] 8 bytes

Question #3

Which of the following are valid `if` statements in ANSI C and Betty-compliant? (Considering `a` and `b` two variables of type `int`)

Please select all correct answers

- [ ]
```
if a > b
{
  return (a);
}
```

- [x]
```
if (a > b)
{
  return (a);
}
```

- [ ]
```
if {a > b}
(
  return {a};
)
```

- [ ]
```
if ((((((a > b))))))
{
  return (a);
}
```

- [x]
```
if (a > b)
  return (a);
```

Question #4

Which of the following are valid `for` statements in ANSI C and Betty-compliant? (Considering `a` and `b` two variables of type `int`)

Please select all correct answers

- [x]
```
for (a = 0; a < b; a++)
{
    printf("%d\n", a);
}
```

- [x]
```
for (a = 0; a < b; a++)
    printf("%d\n", a);
```

(/)

```
for (int a = 0; a < b; a++)
{
    printf("%d\n", a);
}
```

```
a = 0;
for (a < b;;)
{
    printf("%d\n", a++);
}
```

```
a = 0;
for (; a < b;)
{
    printf("%d\n", a++);
}
```

Question #5

Which of the following are valid `while` or `do/while` statements in ANSI C and Betty-compliant? (Considering `a` and `b` two variables of type `int`)

Please select all correct answers

```
while (a = 0; a < b; a++)
{
    printf("%d\n", a);
}
```

```
a = 0;
while (a < b)
{
    printf("%d\n", a);
    a++;
}
```

```
a = 0;
do {
    printf("%d\n", a);
    a++;
} while (a < b);
```

```
a = 0;
while (a < b)
    printf("%d\n", a++);
```

(/)

```
a = 0;
while (a < b)
(
    printf("%d\n", a);
    a++;
)
```

```
a = 0;
do while (a < b)
{
    printf("%d\n", a);
    a++;
}
```

Tasks

0. Positive anything is better than negative nothing

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

This program will assign a random number to the variable `n` each time it is executed. Complete the source code in order to print whether the number stored in the variable `n` is positive or negative.

- You can find the source code here (/rltoken/rrqNDWjrCWdARnWFLPExPw)
- The variable `n` will store a different value every time you will run this program
- You don't have to understand what `rand`, `srand`, `RAND_MAX` do. Please do not touch this code
- The output of the program should be:
  - The number, followed by
    - if the number is greater than 0: `is positive`
    - if the number is 0: `is zero`
    - if the number is less than 0: `is negative`
  - followed by a new line

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 0-positive_or_n
egative.c -o 0-positive_or_negative
julien@ubuntu:~/0x01$ ./0-positive_or_negative
-520693284 is negative
julien@ubuntu:~/0x01$ ./0-positive_or_negative
-973398895 is negative
julien@ubuntu:~/0x01$ ./0-positive_or_negative
-199220452 is negative
julien@ubuntu:~/0x01$ ./0-positive_or_negative
561319348 is positive
julien@ubuntu:~/0x01$ ./0-positive_or_negative
561319348 is positive
julien@ubuntu:~/0x01$ ./0-positive_or_negative
266853958 is positive
julien@ubuntu:~/0x01$ ./0-positive_or_negative
-48147767 is negative
julien@ubuntu:~/0x01$ ./0-positive_or_negative
0 is zero
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `0-positive_or_negative.c`

☑ Done! | Help | Check your code | >_ Get a sandbox | QA Review

1. The last digit

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

This program will assign a random number to the variable `n` each time it is executed. Complete the source code in order to print the last digit of the number stored in the variable `n`.

- You can find the source code here (/rltoken/5HWhPDsq3jq1yCRQFrLI4Q)
- The variable `n` will store a different value every time you run this program
- You don't have to understand what `rand`, `srand`, and `RAND_MAX` do. Please do not touch this code
- The output of the program should be:
  - The string `Last digit of`, followed by
  - `n`, followed by
  - the string `is`, followed by
    - if the last digit of `n` is greater than 5: the string `and is greater than 5`
    - if the last digit of `n` is 0: the string `and is 0`

- if the last digit of `n` is less than 6 and not 0: the string `and is less than 6 and not 0`
  - followed by a new line

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 1-last_digit.c
-o 1-last_digit
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of 629438752 is 2 and is less than 6 and not 0
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of -748255693 is -3 and is less than 6 and not 0
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of -1052791662 is -2 and is less than 6 and not 0
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of -284805734 is -4 and is less than 6 and not 0
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of -284805734 is -4 and is less than 6 and not 0
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of 491506926 is 6 and is greater than 5
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of 954249937 is 7 and is greater than 5
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of 652334952 is 2 and is less than 6 and not 0
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of -729688197 is -7 and is less than 6 and not 0
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of -729688197 is -7 and is less than 6 and not 0
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of 45528266 is 6 and is greater than 5
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of 45528266 is 6 and is greater than 5
julien@ubuntu:~/0x01$ ./1-last_digit
Last digit of 809065140 is 0 and is 0
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `1-last_digit.c`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

2. I sometimes suffer from insomnia. And when I can't fall asleep, I play what I call the alphabet game

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a program that prints the alphabet in lowercase, followed by a new line.

- You can only use the `putchar` function (every other function (`printf` , `puts` , etc...) is forbidden)
- All your code should be in the `main` function
- You can only use `putchar` twice in your code

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 2-print_alphabe
t.c -o 2-print_alphabet
julien@ubuntu:~/0x01$ ./2-print_alphabet
abcdefghijklmnopqrstuvwxyz
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `2-print_alphabet.c`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

3. alphABET

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a program that prints the alphabet in lowercase, and then in uppercase, followed by a new line.

- You can only use the `putchar` function (every other function (`printf` , `puts` , etc...) is forbidden)
- All your code should be in the `main` function
- You can only use `putchar` three times in your code

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 3-print_alphabe
ts.c -o 3-print_alphabets
julien@ubuntu:~/0x01$ ./3-print_alphabets | cat -e
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ$
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `3-print_alphabets.c`

Done! Help Check your code >_ Get a sandbox QA Review

4. When I was having that alphabet soup, I never thought that it would pay off

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a program that prints the alphabet in lowercase, followed by a new line.

- Print all the letters except `q` and `e`
- You can only use the `putchar` function (every other function (`printf` , `puts` , etc...) is forbidden)
- All your code should be in the `main` function
- You can only use `putchar` twice in your code

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 4-print_alphab
t.c -o 4-print_alphabt
julien@ubuntu:~/0x01$ ./4-print_alphabt
abcdfghijklmnoprstuvwxyz
julien@ubuntu:~/0x01$ ./4-print_alphabt | grep [eq]
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `4-print_alphabt.c`

Done! Help Check your code >_ Get a sandbox QA Review

5. Numbers

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a program that prints all single digit numbers of base 10 starting from `0` , followed by a new line.

- All your code should be in the `main` function

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 5-print_number
s.c -o 5-print_numbers
julien@ubuntu:~/0x01$ ./5-print_numbers
0123456789
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `5-print_numbers.c`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

6. Numberz

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a program that prints all single digit numbers of base 10 starting from `0`, followed by a new line.

- You are not allowed to use any variable of type `char`
- You can only use the `putchar` function (every other function (`printf`, `puts`, etc...) is forbidden)
- You can only use `putchar` twice in your code
- All your code should be in the `main` function

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 6-print_number
z.c -o 6-print_numberz
julien@ubuntu:~/0x01$ ./6-print_numberz
0123456789
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `6-print_numberz.c`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

7. Smile in the mirror

mandatory

(/)

Score: 100.00% (*Checks completed: 100.00%*)

Write a program that prints the lowercase alphabet in reverse, followed by a new line.

- You can only use the `putchar` function (every other function (`printf` , `puts` , etc...) is forbidden)
- All your code should be in the `main` function
- You can only use `putchar` twice in your code

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 7-print_tebahpl
a.c -o 7-print_tebahpla
julien@ubuntu:~/0x01$ ./7-print_tebahpla
zyxwvutsrqponmlkjihgfedcba
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `7-print_tebahpla.c`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

8. Hexadecimal

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a program that prints all the numbers of base 16 in lowercase, followed by a new line.

- You can only use the `putchar` function (every other function (`printf` , `puts` , etc...) is forbidden)
- All your code should be in the `main` function
- You can only use `putchar` three times in your code

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 8-print_base16.
c -o 8-print_base16
julien@ubuntu:~/0x01$ ./8-print_base16
0123456789abcdef
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`

- File: `8-print_base16.c`

(/)

Done! | Help | Check your code | >_ Get a sandbox | QA Review

9. Patience, persistence and perspiration make an unbeatable combination for success

mandatory

Score: 100.00% (*Checks completed: 100.00%*)

Write a program that prints all possible combinations of single-digit numbers.

- Numbers must be separated by `,` , followed by a space
- Numbers should be printed in ascending order
- You can only use the `putchar` function (every other function (`printf` , `puts` , etc...) is forbidden)
- All your code should be in the `main` function
- You can only use `putchar` four times maximum in your code
- You are not allowed to use any variable of type `char`

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 9-print_comb.c
-o 9-print_comb
julien@ubuntu:~/0x01$ ./9-print_comb | cat -e
0, 1, 2, 3, 4, 5, 6, 7, 8, 9$
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `9-print_comb.c`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

10. Inventing is a combination of brains and materials. The more brains you use, the less material you need

#advanced

Score: 100.00% (*Checks completed: 100.00%*)

Write a program that prints all possible different combinations of two digits.

- Numbers must be separated by `, `, followed by a space
- The two digits must be different
- `01` and `10` are considered the same combination of the two digits `0` and `1`
- Print only the smallest combination of two digits
- Numbers should be printed in ascending order, with two digits
- You can only use the `putchar` function (every other function (`printf`, `puts`, etc...) is forbidden)
- You can only use `putchar` five times maximum in your code
- You are not allowed to use any variable of type `char`
- All your code should be in the `main` function

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 100-print_comb
3.c -o 100-print_comb3
julien@ubuntu:~/0x01$ ./100-print_comb3
01, 02, 03, 04, 05, 06, 07, 08, 09, 12, 13, 14, 15, 16, 17, 18, 19, 23, 24, 25, 26,
27, 28, 29, 34, 35, 36, 37, 38, 39, 45, 46, 47, 48, 49, 56, 57, 58, 59, 67, 68, 69,
78, 79, 89
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `100-print_comb3.c`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

11. The success combination in business is: Do what you do better... and: do more of what you do...

#advanced

Score: 100.00% (*Checks completed: 100.00%*)

Write a program that prints all possible different combinations of three digits.

- Numbers must be separated by `, `, followed by a space
- The three digits must be different
- `012`, `120`, `102`, `021`, `201`, `210` are considered the same combination of the three digits `0`, `1` and `2`
- Print only the smallest combination of three digits
- Numbers should be printed in ascending order, with three digits
- You can only use the `putchar` function (every other function (`printf`, `puts`, etc...) is forbidden)
- You can only use `putchar` six times maximum in your code
- You are not allowed to use any variable of type `char`

- All your code should be in the `main` function

(/)

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 101-print_comb
4.c -o 101-print_comb4
julien@ubuntu:~/0x01$ ./101-print_comb4
012, 013, 014, 015, 016, 017, 018, 019, 023, 024, 025, 026, 027, 028, 029, 034, 035,
036, 037, 038, 039, 045, 046, 047, 048, 049, 056, 057, 058, 059, 067, 068, 069, 078,
079, 089, 123, 124, 125, 126, 127, 128, 129, 134, 135, 136, 137, 138, 139, 145, 146,
147, 148, 149, 156, 157, 158, 159, 167, 168, 169, 178, 179, 189, 234, 235, 236, 237,
238, 239, 245, 246, 247, 248, 249, 256, 257, 258, 259, 267, 268, 269, 278, 279, 289,
345, 346, 347, 348, 349, 356, 357, 358, 359, 367, 368, 369, 378, 379, 389, 456, 457,
458, 459, 467, 468, 469, 478, 479, 489, 567, 568, 569, 578, 579, 589, 678, 679, 689,
789
julien@ubuntu:~/0x01$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `101-print_comb4.c`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

12. Software is eating the World

#advanced

Score: 100.00% (*Checks completed: 100.00%*)

Write a program that prints all possible combinations of two two-digit numbers.

- The numbers should range from `0` to `99`
- The two numbers should be separated by a space
- All numbers should be printed with two digits. `1` should be printed as `01`
- The combination of numbers must be separated by comma, followed by a space
- The combinations of numbers should be printed in ascending order
- `00 01` and `01 00` are considered as the same combination of the numbers `0` and `1`
- You can only use the `putchar` function (every other function (`printf` , `puts` , etc...) is forbidden)
- You can only use `putchar` eight times maximum in your code
- You are not allowed to use any variable of type `char`
- All your code should be in the `main` function

```
julien@ubuntu:~/0x01$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 102-print_comb
5.c -o 102-print_comb5
julien@ubuntu:~/0x01$ ./102-print_comb5
00 01, 00 02, 00 03, 00 04, 00 05, 00 06, 00 07, 00 08, 00 09, 00 10, 00 11, [...] 4
0 91, 40 92, 40 93, 40 94, 40 95, 40 96, 40 97, 40 98, 40 99, 41 42, 41 43, 41 44, 4
1 45, 41 46, 41 47, 41 48, 41 49, 41 50, 41 51, 41 52, 41 53 [...] 93 95, 93 96, 93
97, 93 98, 93 99, 94 95, 94 96, 94 97, 94 98, 94 99, 95 96, 95 97, 95 98, 95 99, 96
97, 96 98, 96 99, 97 98, 97 99, 98 99
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x01-variables_if_else_while`
- File: `102-print_comb5.c`

Done! | Help | Check your code | >_ Get a sandbox | QA Review