Лабораторна робота № 3

МОВА ПРОГРАМУВАННЯ Сі, КОМПІЛЯЦІЯ, ФУНКЦІЇ ВВОДУ/ВИВОДУ.

Мета роботи:

Вивчити поняття: компіляції, функцій вводу/виводу.

Обладнання та програмне забезпечення:

– IBM сумісна персональна обчислювальна машина;

– онлайн компілятор мови програмування Сі, доступний за посиланням https://repl.it/languages/c

Завдання до роботи:

Написати програму для з використанням мови програмування Сі, котра реалізує наступні функції:

– вивід стрічки “hello, world”;

– вивід довільної стрічки;

– форматований вивід довільної стрічки;

– форматований ввід та вивід набору символів.

Теоретичні відомості

Мова Сі була використана для створення широко поширених операційних систем - Linux, Windows, OS X, iOS і Android, оскільки мова надає вільний доступ до апаратних ресурсів комп'ютера. Порівнюючи її з іншими мовами програмування, такими як Python або Matlab, необхідно відзначити, що в мові Сі відсутня вбудована реалізація таких розвинених мовних засобів, як високорівневі функції для роботи з файлами, порівняння з шаблоном, матричні операції і взаємодія з графічним інтерфейсом користувача. Також в ній відсутні ресурси, що дозволяють виявити помилки при зверненні до пам'яті, помилки виходу за межі масиву. З іншого боку це є перевагою мови Сі котра дозволяє хакерам проникати в комп'ютерні системи, де використовується програмне забезпечення, розроблене без належної уваги до безпеки.

Наведемо першу програму написану мовою Сі:

```
#include <stdio.h>
int main(void)
{
    printf("Hello World\n");
    return 0;
}
```

Зазвичай, програма на мові Сі містить одну або кілька функцій. Кожна програма повинна мати функцію з ім'ям `main`, яка служить стартовою точкою для виконання програми і в кожній окремій програмі ця функція є єдина. Крім цієї головної функції, більшість програм на мові Сі містять набір функцій, які знаходяться в тексті програми і /

або в бібліотеках. Наш приклад hello.c складається з директиви включення в програму заголовного файлу `stdio.h`, оголошення функції `main` і її реалізації. Для підключення до програми стандартної бібліотеки використовується ключове слово `#include`, після якого вказується назва заголовного файлу даної бібліотеки, взята в трикутні дужки.

Заголовок: `#include <stdio.h>`

Заголовки складаються з оголошень функцій, потрібних програмі. У нашому випадку, програма використовує функцію `printf`, яка знаходиться в стандартній бібліотеці введення / виведення і оголошена в заголовному файлі `stdio.h.`

Робота програми починається з виконання операторів, що містяться в функції з ім'ям `main`. Набір цих операторів називається **тілом функції** `main`. Будь-яка програма на Сі повинна мати одну і тільки одну функцію `main`. Тіло функції містить послідовність операторів, кожний з яких має закінчуватися крапкою з комою. Ключове слово `int` вказує, що результат функції (або значення, що повертається) має цілочисельний тип. Значення, що повертається показує, чи була програма виконана успішно (якщо так, то функція main повертає нуль, якщо програма завершилася аварійно, то функція main повертає код помилки). По завершенню програми, результат `main` передається в те оточення, з якого вона була запущена.

Тіло функції: `printf("Hello World!\n");`

Тіло функції `main` з нашого прикладу містить єдиний виклик функції `printf`, що друкує рядок "Hello world!", яка закінчується символом переведення рядка "`\n`".

У загальних рисах, всі програми влаштовані подібно до нашого прикладу hello.c, з тією різницею, що складні програми можуть складатися з мільйонів рядків тексту, розбитих на безліч функцій і файлів.

Запуск Сі-програми.

Програми на мові Сі можуть виконуватися процесорами з різними системами команд. Переносимість програм - це ще одна перевага мови програмування Сі. Перед запуском програма повинна бути скомпільована Сі-компілятором. Існує кілька відмінних реалізацій Сі-компіляторів, включаючи cc (C компілятор) і gcc (GNU C компілятор). Компілятор gcc доступний для вільного використання. Він поставляється в складі всіх дистрибутивів операційної системи Linux і не вимагає додаткових дій по інсталяції. В операційній системі Windows може знадобитися установка пакета програм Cygwin. Процес підготовки файлу з програмою, його компіляція і виконання коду програми однакові для всіх програм на мові Сі.

Для використання хмарного середовища розробки:

1. Перейдіть за посиланням https://repl.it/languages/c .
2. У вікні браузера відкриється онлайн-середовище розробки із стандартним кодом першої програми “Hello World”.
3. Натисніть кнопку “Run”.
4. У вікні видач в правій частині екрану повинен з'явитися рядок "Hello World".

Функції вводу/виводу, `printf` та `scanf` з бібліотеки `stdio`

`printf` - вивід форматованого рядка в стандартний потік виведення.

синтаксис:

```
#include <stdio.h>
int printf (const char * format, ...);
```

аргументи:

format - вказівник на рядок з описом формату.

Значення, що повертається:

При успішному завершенні виведення повертається кількість виведених символів.
При помилці повертається негативне число.

Опис:

Функція printf виводить в стандартний потік виведення рядок, відформатований відповідно до правил, зазначених в рядку, на які вказує аргумент format. Правила задаються набором трьох типів директив:

1. Звичайні символи, які виводяться без зміни (крім '%' і '\');
2. Специфікатори формату;
3. Спеціальні символи.

Специфікатори формату.

Кожен специфікатор формату починається з символу '%' і має такий вигляд:

% [прапорці] [ширина] [точність] [модифікатори] [тип перетворення]

прапорці

прапорець	Значення прапорця
- (дефіс)	Результат перетворення вирівнюється по лівому краю (за замовчуванням - по правому краю)
+	Перед позитивними числами виводиться знак '+', а перед від'ємними - знак '-' (за замовчуванням виводиться тільки знак '-' перед відємними числами)
' ' (пробіл)	Якщо не вказано модифікатор '+', то перед позитивними числами, на місці знака числа, буде виводитися пробіл.
#	Використовувати альтернативну форму подання виведеного числа. При виведенні чисел в шістнадцятковому форматі (перетворювач 'x' або 'X') перед числом буде вказуватися 0x або 0X відповідно. При виведенні чисел в вісімковому форматі (перетворювач 'o') перед числом буде вказуватися 0. При виведенні чисел з плаваючою комою (перетворювачі e, E, f, g і G) завжди буде міститися десяткова крапка (за замовчуванням десяткова крапка виводиться тільки при ненульовий дробовій частини). При використанні перетворювачів g і G хвостові нулі НЕ будуть видалятися (за замовчуванням видаляються).
0	Якщо не вказано прапорець '-', то зліва від виведеного числа будуть виведені символи "0" для підгону числа до зазначеної ширини. Якщо для перетворювачів d, i, o, x або X вказана точність, то прапорець 0 ігнорується.

Специфікатор **[прапорець]** можна не вказувати.

Ширина

Специфікатор [**ширина**] задає мінімальний розмір виведеного числа в символах. Якщо кількість символів в виведеному числі менша зазначеної мінімальної ширини, то символи, котрих бракує заповнюється нулями або пропусками зліва чи справа в залежності від зазначених прапорців. Ширина вказується або цілим числом, або символом * з подальшим зазначенням імені змінної типу int, що містить значення ширини, перед аргументом до якого він відноситься. Якщо аргумент має від'ємне значення, то він еквівалентний відповідному позитивному значенню з прапорцем "-".
Специфікатор [**ширина**] можна не вказувати.

Точність

Дія специфікатора [**точність**] залежить від типу виведеного числа:

- Для типів d, i, o, u, x, X визначає мінімальне число виведених цифр. Якщо кількість виведених цифр в числі менше, ніж зазначено в специфікатором [точність], то число, що виводиться буде доповнено нулями зліва. Наприклад, якщо при виведенні числа 126 вказати точність 4, то на екран буде виведено число 0126

- Для типів a, A, e, E, f, F визначає кількість виведених цифр після коми. Якщо в виведеному числі кількість значущих цифр після коми менше зазначеної точності, то відсутні символи виводяться нулями праворуч від числа. Якщо більше, то зайві цифри не виводяться. Наприклад, якщо при виведенні числа 126.345 вказати точність 2, буде виведено на екран число 126.34, а якщо вказати точність 5, то на екран буде виведено число 126.34500.

- Для типів g і G визначає максимальне виведене число цифр. Наприклад, якщо при виведенні числа 126.345 вказати точність 4, буде виведено на екран число 126.3. Якщо при виведенні числа 1242679.23 вказати точність 3, буде виведено на екран число 1.24e + 06.

Точність вказується у вигляді символу точка, за яким слідує десяткове число або символ "*", з подальшою вказівкою імені змінної типу int, що містить значення точності, перед аргументом до якого він відноситься.

Специфікатор [**точність**] можна не вказувати.

Модифікатор

Специфікатор [**модифікатор**] визначає розмір даних, що виводяться (char, short, long, longlong). Специфікатори використовуються для виведення чисел типу: char, short int, long int, long long int, long double або для явного перетворення даних, що виводяться. Наприклад, якщо є змінна типу int, а необхідно вивести її як short int. Доступні модифікатори наведені в таблиці 2.

Таблиця 2

модифікатор	Результат застосування
h	Для виводу числа типу short int або unsigned short int. Або для явного перетворення при виведенні цілочисленного числа до типу short int або unsigned short int. Використовується спільно з типами перетворення: d, i, o, u, x і X, n.
hh	Для виводу числа типу char або unsigned char. Або для явного перетворення при виведенні цілочисленного числа до типу char або unsigned char. Використовується спільно з типами перетворення: d, i, o, u, x і X, n.
l	Для виводу числа типу long int або unsigned long int. Або для явного перетворення при виведенні цілочисленного числа до типу long int або unsigned long int. Використовується спільно з типами перетворення: d, i, o, u, x і X, n.
ll	Для виводу числа типу long long int або unsigned long long int. Або для явного перетворення при виведенні цілочисленного числа до типу long long int або unsigned long long int. Використовується спільно з типами перетворення: d, i, o, u, x і X, n.
L	Для виводу числа типу long double. Або для явного перетворення при виведенні числа с плаваючою комою до типу long double. Використовується спільно з типами перетворення: e, E, f, g і G.

Специфікатор [**модифікатор**] можна не вказувати.

Тип перетворення

Специфікатор [**тип перетворення**] визначає як треба інтерпретувати і виводити число, наприклад як знакове цілочислене в десятковому вигляді, або беззнакове цілочислене в шістнадцятковому, або як число з плаваючою комою і так далі. Перелік доступних перетворень і їх опис зазначено в таблиці 3.

Таблиця 3

Тип перетворення	Результат
d, i	Вивід цілого числа зі знаком в десятковій системі числення. За замовчуванням виводиться число розміром sizeof (int), з правим вирівнюванням, зазначенням знака тільки для негативних чисел.
u	Вивід цілого числа без знака в десяткової системі числення. За замовчуванням виводиться число розміром sizeof (int), з правим вирівнюванням.
o	Вивід цілого числа без знака в вісімковій систем числення. За замовчуванням виводиться число розміром sizeof (int), з правим вирівнюванням.
x, X	Вивід цілого числа без знака в шістнадцятковій системі числення. Причому для перетворення x використовуються символи abcdef, а для X - символи ABCDEF. За замовчуванням виводиться число розміром sizeof (int), з правим вирівнюванням.

f, F	Вивід числа з плаваючою комою в вигляді [-] dddd.ddd. За замовчуванням виводиться число з точністю 6, якщо число по модулю менше одиниці, то перед десятковою комою виводиться нуль, знак вказується тільки для негативних чисел, з правим вирівнюванням. Розмір за замовчуванням sizeof (double).
e, E	Вивід числа з плаваючою комою в експоненційній формі запису, у вигляді [-] dddd.ddde ± dd, причому для модифікатора e використовується символ e, а для модифікатора E - символ E. За замовчуванням виводиться число з точністю 6, якщо число по модулю менше одиниці, то перед десятковою комою виводиться нуль, знак вказується тільки для негативних чисел, з правим вирівнюванням. Після символу "e" (або "E") завжди виводиться дві цифри (вони дорівнюють 0, якщо аргумент дорівнює 0).
g, G	Вивід числа з плаваючою комою в формі залежній від величини числа. Наприклад число 345.26 буде виведено як 345.26, а число 1344527.434 як 1.34453e + 06. За замовчуванням виводиться 6 значущих цифр числа з округленням, якщо число по модулю менше одиниці, то перед десятковою комою виводиться нуль, знак вказується тільки для негативних чисел, з правим вирівнюванням.
a, A	Вивід числа з плаваючою комою в шістнадцятковому форматі, в експоненційній формі запису. Наприклад число 137.434 буде виведено як 0x1.12de353f7ced9p + 7. Експонента позначається символом p. Для модифікатора a використовується символ p, а для модифікатора A - символ P. По замовчуванням знак вказується тільки для негативних чисел, вирівнювання - праве.
c	Вивід символу, що відповідає числу вказаному в аргументі функції. За замовчуванням число приводиться до типу unsigned char.
s	Вивід рядка, на котрий посилається вказівник в аргументі функції printf. Рядок виводиться поки не зустрінеться символ кінець рядка (/0). За замовчуванням рядок повинен позначатися як char *. Якщо вказано модифікатор l, то рядок інтерпретується як wchar_t *. Для функції wprintf рядок за замовчуванням обробляється як wchar_t *.
S	Аналогічний перетворенню s з модифікатором l (ls).
p	Вивід вказівника. Результат виводу залежить від архітектури і використовуваного компілятрора. Наприклад, на 16 бітній платформі MS-DOS вивід буде мати вигляд типу FFAB: 1402, а на 32-бітної платформі з плоскою адресацією - 00FC0120.
n	Запис за адресою, вказаною в аргументі функції, кількості виведених символів функцією printf до зустрічі перетворювача %n. При обробці перетворювача % n ніякого виводу символів не проводиться.

Спеціальні символи

Для форматування виводу в функції printf передбачений набір спеціальних символів. Перелік спеціальних символів наведено в таблиці 4.

Таблиця 4.

Символ	Значення
\a	Видається звуковий сигнал.
\b	Видаляє останній виведений символ.
\f	Перехід на новий рядок. Новий символ буде надрукований на новому рядку, на позиції, наступній за останнім надрукованим символом на попередньому рядку.
\n	Новий рядок. Наступний символ буде надрукований з початку нового рядка.
\r	Повернення на початок рядка.
\t	Табуляція по горизонталі.
\v	Вертикальна табуляція.
\\	Вивід зворотного слеша.
\”	Вивід лапок.
\% або %%	Вивід відсотка.
\num	Вивід символу за його кодом. Наприклад при виведенні \ 123 буде надрукований символ 'S'.

Приклад:

```
#include <stdio.h>

int main (void)
{
 printf ("1. Вивід простої стрічки\n");

 printf ("\n2. Вивід цілих чисел\n");

 printf ("2.1 Вивід числа 123 з форматом по замовчуванню:\n");
 { int d1=123;
   printf ("%d\n",d1);
 }
 printf ("\n2.2 Вивід чисел з вирівнюванням по правому краю:\n");
 { int d1=123, d2=42, d3=1543;
   printf ("%6d\n%6d\n%6d\n",d1,d2,d3);
 }
 printf ("\n2.3 Вивід чисел з вирівнюванням по лівому краю:\n");
 { int d1=123, d2=42, d3=1543;
   printf ("%-6d\n%-6d\n%-6d\n",d1,d2,d3);
 }
 printf ("\n2.4 Вивід числа з виводом 0 в символах котрих бракує:\n");
 { int d1=-123;
   printf ("%06d\n",d1);
 }
 printf ("\n2.5 Вивід числа 123 з заданою шириною 6 символів:\n");
 { int d1=123;
   printf ("%6d\n",d1);
 }
 printf ("\n2.6 Вивід числа 123 з заданою точністю 6 символів:\n");
 { int d1=123;
   printf ("%.6d\n",d1);
 }
```

```
printf ("\n2.7 Вивід числа типу char:\n");
{ char d1=123;
  printf ("%hhd\n",d1);
}
printf ("\n2.8 Вивід числа типу short int:\n");
{ short int d1=123;
  printf ("%hd\n",d1);
}
printf ("\n2.9 Вивід числа типу long int:\n");
{ long int d1=123;
  printf ("%ld\n",d1);
}
printf ("\n2.10 Вивід числа типу long long int:\n");
{ long long int d1=123;
  printf ("%lld\n",d1);
}
printf ("\n2.11 Вивід цілого беззнакового числа:\n");
{ unsigned int d1=123;
  printf ("%u\n",d1);
}
printf ("\n2.12 Вивід цілого числа в вісімковому форматі:\n");
{ unsigned int d1=123;
  printf ("%o\n",d1);
}
printf ("\n2.13 Вивід цілого числа в шістнадцятковому форматі:\n");
{ unsigned int d1=123;
  printf ("%x\n",d1);
}
printf ("\n2.14 Вивід цілого числа в шістнадцятковому форматі:\n");
{ unsigned int d1=123;
  printf ("%#x\n",d1);
}
printf ("\n3. Вивід чисел з плаваючою комою\n");
printf ("3.1 Вивід числа 123.456 з форматом по замовчуванню:\n");
{ double d1=123.456;
  printf ("%f\n",d1);
}
printf ("\n3.2 Вивід числа 123.456 з точністю 2:\n");
{ double d1=123.456;
  printf ("%.2f\n",d1);
}
printf ("\n3.3 Вивід числа 123.456 в експоненційній формі:\n");
{ double d1=123.456;
  printf ("%e\n",d1);
}
printf ("\n3.4 Використання перетворювача g:\n");
{ double d1=123.4567, d2=1234567.34567;
  printf ("%g\n%g\n",d1,d2);
}
printf ("\n3.5 Вивід числа 123.456 в шістнадцятковому форматі:\n");
{ double d1=123.456;
  printf ("%a\n",d1);
}
printf ("\n4. Вивід символа 'k':\n");
{ char d1='k';
  printf ("%c\n",d1);
}
printf ("\n5. Вивід стрічки 'abc':\n");
{ char d1[4]="abc";
```

```
   printf ("%s\n",d1);
 }
 printf ("\n6. Вивід вказівника:\n");
 { char d1[4]="abc";
   printf ("%p\n",d1);
 }
 printf ("\n7. Підрахунок числа виведених символів:\n");
 { int d1=0;
   fflush (stdout);
   printf ("Derived characters:%n",&d1);
   printf (" %d\n",d1);
 }

 //Примусовий вивід стрічки, не очікуючи заповнення буфера
 printf ("\n6. Вивід стрічки з буфера стрічки\n");
 { char d1[4]="abc";
   printf ("%s",d1);
   fflush (stdout);
 }
   return 0;
}
```

Форматований ввід - scanf

Функція `scanf` є аналогічною до `printf`, але тільки для вводу. Вона надає багато з тих самих можливостей перетворення, щоправда, в зворотному напрямку.

```
int scanf (char * format, arg1, arg2,...);
```

`scanf` зчитує знаки зі стандартного вводу, інтерпретуючи їх відповідно до специфікації, вказаної форматом (format), і зберігає результат за допомогою решти аргументів. Аргумент формату описано нижче; решта аргументів, кожен з яких повинен бути вказівником, вказує на те, де відповідний перетворений ввід потрібно зберегти.

`scanf` зупиняється, якщо вона вичерпає свій список формату, або коли ввід не збігається із контрольною специфікацією. Вона повертає як значення число елементів вводу, які були сприйняті і яких було присвоєно. Це можна використати, щоб дізнатися, скільки об’єктів було введено.

Керуюча стрічка, на яку вказує format, складається з символів трьох типів:

- Специфікатори формату
- Спеціальні символи
- Інші символи (не спеціальні)

Специфікатори формату слідують за символом % (процент) и вказують scanf() тип даних, які будуть зчитані наступними. Ці параметри є аналогічними до тих, що використовує printf().

Наприклад, %s зчитує стрічку, а %d зчитує десяткову змінну цілого типу.

Стрічка формату зчитується зліва направо, при цьому встановлюється відповідність між кодами формату і аргументами зі списку аргументів.

Всі змінні, які приймають значення від функції scanf(), повинні знаходитися за їхніми адресами. Це означає, що всі аргументи функції повинні бути вказівниками на змінні. Таким чином, Сі створює можливість передачі по посиланню, і це дозволяє функції змінювати вміст аргументу.

Наприклад, для того, щоб зчитати ціле число і присвоїти його значення змінній count, необхідно скористатися наступним зверненням до scanf():

```
scanf("%d", &count);
```

Стрічки є масивами символів, і ім'я масиву є адресою першого елемента масиву, тобто по суті є вказівником. Тому для зчитування стрічки в масив символів address, можна використати команду

```
scanf("%s", address);
```

В цьому випадку ім'я address уже є вказівником, і не потребує префіксу &.

Елементи даних вводу повинні розділятися пробілами, знаками табуляції або нового рядка. Розділові знаки (крапки, коми і т.д.) не вважаються розділювачами. Це означає, що оператором

```
scanf("%d%d", &r, &c);
```

послідовність 10 20 буде сприйнята, а послідовність 10,20 – ні.

Комбінація %*, розміщена перед специфікатором формату, зчитує дані вказаного типу, але блокує їхнє присвоювання. Т.ч., код

```
scanf ("%d%*c%d", &x, &y);
```

при вводі послідовності 10/20 присвоює значення 10 змінній *x*, відкидає символ / і присвоює значення 20 змінній *y*.

Модифікатор максимальної ширини поля є цілим числом, яке розміщується між знаком % і специфікатором формату. Він обмежує кількість зчитуваних символів для довільного поля. Наприклад, якщо необхідно зчитати не більше 20 символів в масив address, слід написати

```
scanf ("%20s", address);
```

Приклад:

```
#include <stdio.h>

int main (void)
{
   char str[80];
   int i;
   /* зчитування стрічки і цілого */
   scanf("%s%d", str, &i);
   /* зчитування до 79 символів у str */
   scanf ("%79s", str);
   /* пропуск цілого між двома стрічками */
   scanf (" %s%*d%s", str, &i, str);
   return 0;
}
```

ПОРЯДОК ВИКОНАННЯ РОБОТИ:

1. Опрацювати і засвоїти матеріал наведений в теоретичних відомостях.
2. Написати програму на мові програмування Сі, котра ілюструє роботу функції printf() з використанням основних специфікаторів. Вивести ціле число у десятковому, двійковому, вісімковому і шістнадцятковому форматах. Вивести дійсне число (з дробовою частиною) у формі з плаваючою комою, в експоненційній формі і в гнучкій формі (специфікатор g). Вивести символ, стрічку і вказівник.
3. Написати програму котра реалізує введення кількох стрічок символів та виведення їх у складі форматованого тексту (наприклад, таблиця списку групи з колонками: № п/п; Прізвище, Ініціали; Ел.пошта; Улюблений колір тощо).
4. В звіті навести копії екранів та написаний код.
5. Зробити висновки.

```
char Name[20];
printf("%s","What is your name? \t");
scanf("%s",Name);
printf("%s%s%s","\nHello, ", Name, "!\n");
return 0;
```