ЛАБОРАТОРНА РОБОТА №3

Перша частина (базове завдання)

ЦИКЛІЧНІ ОПЕРАТОРИ. ФУНКЦІЇ

Мета: Навчитися працювати із операторами циклу, а також створювати власні функції.

Вхідні дані: Значення кількості предметів та значення балу по кожному предмету для обрахунку середнього балу успішності студента.

Вихідні дані: Обраховане значення середнього балу успішності студента, який виводиться на екран монітора.

Виконання лабораторної роботи передбачає опанування використання циклічних операторів, а також написання власних функцій, які реалізують учбове завдання.
В лабораторній роботі використовуються циклічні оператори: **for, while, do/while**.

Циклічний оператор **`for`** має наступний вигляд:

```
for( ініціалізація; перевірка; оновлення ) {
    оператори;
}
```

Три вирази, що записані в дужках після ключового слова **`for`** відділяються один від одного крапкою з комою.

Вираз *"ініціалізація"* виконується лише один раз до того як буде виконаний хоча б один із операторів в тілі циклу.

Вираз *"перевірка"* — логічний вираз, який впливає на виконання тіла циклу. Якщо результат логічного виразу **істинний**, тоді тіло циклу виконується (виконується ітерація циклу). Якщо результат логічного виразу **хибний**, то відбувається вихід із циклу.

Вираз *"оновлення"* передбачає модифікацію змінної, яка управляє циклом і використовується у виразі *"перевірка"*. Після виконання виразу *"оновлення"* знову обраховується вираз *"перевірка"*, і якщо результат ***істинний***, то виконується тіло циклу (виконується ітерація), якщо результат ***хибний*** — відбувається вихід із циклу.

***Цикл* `do/while`** — цикл з постумовою. Спочатку виконується тіло циклу (виконується ітерація), і лише після цього виконується перевірка умови, і в залежності від результату логічного виразу, який записується в умові, відбувається продовження циклу, або ж відбувається вихід із циклу.

***Цикл* `while`** — цикл з передумовою. Спочатку виконується перевірка умови, і в залежності від результату логічного виразу, який записаний в умові, відбувається виконання тіла циклу, або ж вихід із циклу.

Крім циклічних операторів, в лабораторній роботі необхідно також написати власні функції, які реалізують певні алгоритми.

При використанні власної функції необхідно записати прототип функції, а також виконати опис функції.

Функції, які мають бути реалізовані в лабораторній роботі, поділяються на функції, що ***повертають*** значення, та функції, які ***не повертають*** значення (це **функції типу `void`**).

В лабораторній роботі необхідно реалізувати функцію, яка підраховує середній бал успішності. Ця функція буде називатися `averageGrade`. При цьому функція отримує аргумент — це кількість предметів для яких буде підрахований середній бал. Отримавши цей аргумент, передбачається введення з клавіатури балів по кожному предмету. На етапі введення балів, необхідно передбачити можливість перевірки потрапляння введеного балу в проміжок **60…100**, і якщо введений бал виходить за ці рамки, необхідно забезпечити можливість користувачу ввести правильний бал повторно. Після введених балів, функція обраховує середній бал успішності і повертає це значення в точку виклику, де повернений середній бал присвоюється відповідній змінній.

Інша функція, яка має бути реалізована в програмі — це функція типу `void`, яка отримує в якості аргументу середній бал успішності (ця функція буде називатися `printGrade`). Функція повинна виводити середній бал на екран монітора.

Враховуючи, що в програмі необхідно буде використати бібліотечні функції по введенню та виведенню значень, а також будуть використані бібліотечні функції `system` та `exit`, тому необхідно за допомогою директиви `include` вставити в програму відповідні заголовочні файли:

```
#include <stdio.h>
#include <stdlib.h>
```

Також необхідно вставити прототипи тих двох власних функцій, які будуть описуватися в програмі і виконувати вище зазначені дії. Для цих функцій обрано імена `averageGrade` та `printGrade`. Вказані функції будуть отримувати відповідні параметри. Прототипи для цих двох функцій можуть бути записані так:

```
double averageGrade(unsigned int number_of_subjects);
void printGrade( double ave_grade );
```

Функція `averageGrade` отримує один параметр типу `unsigned int`. Цей параметр буде відповідати за кількість предметів, для кожного з яких, буде передбачатися введення балу. Функція буде розраховувати середній бал і повертати це значення в місце виклику функції в програмі.

Функція `printGrade` має тип даних `void`. Ця функція не повертає ніяких значень. Функція отримує параметр типу `double`, і цей параметр містить середній бал, який був раніше обрахований в програмі. Функція `printGrade` повинна вивести на екран значення середнього балу, також супроводжувати виведення значення певними коментарями. Виведення середнього балу відбувається із одним знаком після коми.

Виклик функцій `averageGrade` та `printGrade` виконується у функції `main`.

Також у функції `main` оголошуються змінні `subjects` та `mean`.

Змінна `subjects` зберігає введене з клавіатури значення кількості предметів. Для заданої кількості предметів необхідно буде ввести бали і розрахувати середнє значення. Змінна `subjects` буде мати тип даних `unsigned int`.

Змінна `mean` буде зберігати значення середнього балу. Змінна `mean` буде отримувати значення, яке буде повертатися функцією `averageGrade`. Змінна `mean` буде мати тип даних `double`.

Необхідно передбачити, що якщо користувач на запит ввести кількість предметів для яких буде обраховуватися середній бал (це значення зберігається в змінній `subjects`) введе значення нуль, то програма має завершитися.

Враховуючи, що при виведенні текстової інформації в командне вікно (в консоль) можуть некоректно виводитися літери українського алфавіту, тому текстові коментарі можна записувати на трансліті (написання українських слів літерами англійського алфавіту) або ж виводити текстові повідомлення англійською мовою.

Таким чином, функція `main` може мати такий вигляд:

```
int main()
{
    unsigned int subjects;
    double mean;

    printf("Enter number of subjects. This value must be greater than ZERO.");
    printf("\nOr enter ZERO to exit this program");

    printf("\n\nsubjects= ");
    scanf("%u", &subjects);

    if(subjects == 0 )
        exit(0);

    mean = averageGrade( subjects ); // виклик функції averageGrade. Функція повертає
                                     // обраховане середнє значення

    printGrade( mean );              // виклик функції printGrade. Функція має тип
                                     // даних void. Функція виводить на екран
                                     // значення розрахованого раніше середнього балу

    return 0;
}
```

Тепер необхідно виконати опис двох власних функцій.

Перша функція `averageGrade` буде мати наступний заголок:

```
double averageGrade(unsigned int number_of_subjects)
```

Коли функція `averageGrade` буде викликана, відбудеться створення локальної змінної на ім’я `number_of_subjects` типу `unsigned int`, і ця змінна отримає значення параметру, який стоїть у виклику функції `averageGrade` всередині функції `main`. Таким чином, змінна `number_of_subjects` отримує ніби як копію значення змінної `subjects`, яка виступає аргументом функції `averageGrade` при її виклику у функції `main`.

Передбачається, що бали, які будуть вводитися мають бути в діапазоні **60…100**. Якщо користувач буде намагатися вводити значення, що виходять за цей діапазон, програма повинна повторити запит на введення балу. Це може бути реалізовано за допомогою оператора циклу **do/while**:

```
do{
    printf("\ngrade #%u: ", i);
    scanf("%u", &grade);
}while( grade < 60  ||  grade > 100 );
```

де змінна `i` зберігає номер предмету для якого вводиться бал. Змінна `i` має тип даних `unsigned int`.

А змінна `grade` зберігає значення балу. Ця змінна також має тип даних `unsigned int`.

Даний частина програмного коду буде призводити до виведення на екран повідомлення наступного виду (в наведеному прикладі вважається, що змінна `i=2`) :

```
grade #2:
```

та буде очікуватися введення балу з клавіатури. Якщо користувач введе значення, що **виходить** за діапазон **60…100**, тоді відбудеться повторний запит, наприклад:

```
grade #2: 52
grade #2: 102
grade #2:
```

Після введення правильного значення, відбудеться перехід до введення балу по наступному предмету, наприклад:

```
grade #2: 52
grade #2: 102
grade #2: 88
grade #3:
```

Кількість предметів зберігається в змінній number_of_subjects. Таким чином, введення балів по кожному предмету необхідно реалізувати за допомогою циклічного оператора. Наприклад, за допомогою циклу **for**.

Для підрахунку середнього балу, необхідно знайти суму всіх введених балів. Для збереження цієї суми використовується змінна sum типу unsigned int. Ця змінна оголошується всередині функції averageGrade. Частина програмного коду, яка буде відповідати за правильність введення балів і підрахунок суми балів, буде мати наступний вигляд:

```
for( i=1;   i <= number_of_subjects;   i++ ){

     do{
         printf("\ngrade #%u: ", i);
         scanf("%u", &grade);
     }while( grade < 60  ||  grade > 100 );

     sum = sum + grade;
 }
```

Після обрахунку суми балів необхідно знайти середній бал. Середній бал буде обраховуватися як сума балів, яка поділена на кількість предметів.

Обраховане середнє значення буде зберігатися в змінній mean_value. Ця змінна буде мати тип даних double. Це буде локальна змінна, що оголошується всередині функції averageGrade. Враховуючи що сума балів та кількість предметів зберігаються в змінних типу unsigned int, то щоб при діленні була збережена дробова частина, необхідно виконати явне перетворення типів даних у виразі, в якому знаходиться середній бал:

```
mean_value = (double)sum / (double)number_of_subjects;
```

Значення змінної mean_value повертається за допомогою оператора return в місце виклику функції averageGrade в функції main. Повернене значення у функції main присвоюється змінній mean.

Опис функції averageGrade, яка підраховує середнє значення, а також містить текстові коментарі, які повинні дати зрозуміти користувачу, що від нього очікується, може мати наступний вигляд:

```
double averageGrade(unsigned int number_of_subjects)
{
    unsigned int i, grade, sum;
    double mean_value;


    system("cls");

    printf("Enter grades for each subject. Grade must be between 60...100.");
    printf("\nTotal number of grades equals to %u\n", number_of_subjects);

    sum = 0;

    for( i=1;   i <= number_of_subjects;   i++ ){

        do{
            printf("\ngrade #%u: ", i);
            scanf("%u", &grade);
        }while( grade < 60  ||  grade > 100 );

        sum = sum + grade;
    }


    mean_value = (double)sum / (double)number_of_subjects;

    return mean_value;
}
```

Друга функція — printGrade. Ця функція виводить значення середнього балу на екран. Вона має наступний заголовок:

```
void printGrade( double ave_grade )
```

При виклику функції printGrade буде створена локальна змінна ave_grade, і цій змінній буде присвоєно значення аргументу, який записується у виклику функції printGrade всередині функції main. У виклику функції printGrade всередині функції main в якості аргументу виступає змінна mean. Таким чином, змінній ave_grade буде присвоєно значення змінної mean.

Функція `printGrade` має тип даних `void`, тобто ця функція **не повертає** ніякого значення в місце виклику.

Функція `printGrade` може мати наступний вигляд:

```
void printGrade( double ave_grade )
{
    system("cls");
    printf("\n");

    printf("Average grade: %.1lf\n", ave_grade);
}
```

Функція очищує екран за допомогою команди: `system("cls");`
Переводить курсор на новий рядок за допомогою команди: `printf("\n");`
Виводить на екран середнє значення балу. При цьому значення виводиться із одним знаком після коми. Виведення на екран середнього балу виконується за допомогою команди: `printf("Average grade: %.1lf\n", ave_grade);`

Загалом, весь текст програми може мати наступний вигляд:

```
#include <stdio.h>
#include <stdlib.h>

double averageGrade(unsigned int number_of_subjects);
void printGrade( double ave_grade );

int main()
{
    unsigned int subjects;
    double mean;

    printf("Enter number of subjects. This value must be greater than ZERO.");
    printf("\nOr enter ZERO to exit this program");

    printf("\n\nsubjects= ");
    scanf("%u", &subjects);

    if(subjects == 0 )
        exit(0);

    mean = averageGrade( subjects );

    printGrade( mean );

    return 0;
}
```

```
//------------ Опис функції, яка знаходить середній бал ------------

double averageGrade(unsigned int number_of_subjects)
{
    unsigned int i, grade, sum;
    double mean_value;


    system("cls");

    printf("Enter grades for each subject. Grade must be between 60...100.");
    printf("\nTotal number of grades equals to %u\n", number_of_subjects);

    sum = 0;

    for( i=1;   i <= number_of_subjects;   i++ ){

        do{
            printf("\ngrade #%u: ", i);
            scanf("%u", &grade);
        }while( grade < 60  ||  grade > 100 );

        sum = sum + grade;

    }


    mean_value = (double)sum / (double)number_of_subjects;

    return mean_value;

}


//-------------- Опис функції, яка виводить середній бал на екран -----------------


void printGrade( double ave_grade )
{
    system("cls");
    printf("\n");

    printf("Average grade: %.1lf\n", ave_grade);

}
```

Завдання на лабораторну роботу

1. Запустити програму, перевірити правильність виконання обрахунків.
2. Внести зміни в програму, щоб значення середнього балу виводилося з трьома знаками після коми.
3. В функції `averageGrade` цикл `for` замінити на цикл `while`.

4. Внести зміни в функцію `printGrade`, щоб крім виведення числового значення середнього балу, також виводився текстовий еквівалент оцінки, в залежності від того, в який проміжок потрапить значення середнього балу:

60…64 — Достатньо (Sufficient — *на англійській мові*)
65…74 — Задовільно (Satisfactory — *на англійській мові*)
75…84 — Добре (Good — *на англійській мові*)
85…94 — Дуже добре (Very good — *на англійській мові*)
95…100 — Відмінно (Excellent — *на англійській мові*)

Наприклад, якщо середній бал становив значення 92.125, то на екрані мало б бути повідомлення наступного виду:

```
Average grade: 92.125 - Very good
```

Якщо, наприклад, середній бал становив 62.500, то на екрані мало б бути повідомлення:

```
Average grade: 62.500 - Sufficient
```

Для реалізації даного пункта завдання необхідно використати умовний оператор **`if/else`**.

5. Виконати пукнт 4, але **замість** оператора **`if/else`**, використати умовний оператор **`if.`**

6. Написати власну функцію типу void, яка не отримує жодних параметрів, і виводить текстове повідомлення — прізвище та ім’я студента на початку роботи програми перед введенням з клавіатури вхідних даних щодо кількості предметів для якої буде знаходитися середній бал успішності. Прототип такої функції може бути наступний:

```
void printName (void);
```

7. Підготувати звіт.