Лабораторна робота № 7
ФУНКЦІЇ.

Мета роботи:
Вивчити поняття і застосування функцій.
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 на мові програмування Сі, котра використовує функції для:

- розв'язку квадратного рівняння;
- знаходження суми чисел рекурсивно;
- розв'язку рівняння методом половинного ділення.

Теоретичні відомості

Функції

специфікатор_типу імя_функції (список параметрів)
{
  тіло функції;
}

Специфікатор_типу визначає тип значення, що повертається функцією (результат функції) за допомогою оператора return. Якщо тип не вказано, передбачається, що функція повертає цілочисельні значення.

Список параметрів - це розділений комами список змінних, який одержує значення аргументів при виклику функції. Функція може бути без параметрів в такому випадку список параметрів містить ключове слово void.

Оператор return має два значення:

1. Вихід з функції;
2. Повернення значень викликаної функції;

Коли використовується оператор return в main(), програма повертає код завершення процесу що її викликав (операційній системі). Значення, що повертається повинно бути цілого типу. Більшість операційних систем, трактують 0 як нормальне завершення програми. Решта значення сприймаються як помилки.

Якщо не вказано значення, що повертається, то в залежності від конкретної платформи, з функції буде повернуто випадкове значення, що лежить в регістрі повернення (наприклад, регістр EAX на платформі x86). Тому краще використовувати оператор return.

Попереднє оголошення

```
#include <stdio.h>
#include <math.h>
double myfunc(); // попереднє оголошення без вказання кількості і типу аргументів
int main(void)
{
   printf ("%lf", myfunc (10.0));
   return 0;
}
```

```
double myfunc(double x)
{
   return sqrt (x) * 2.0;
}
```

Оголошення функції через прототип

```
#include <stdio.h>
#include <math.h>
double myfunc(double x); // прототип, котрий повністю описує функцію
int main(void)
{
   printf ("%lf", myfunc (10.0));
   return 0;
}

double myfunc(double x)
{
   return sqrt (x) * 2.0;
}
```

Правила видимості функцій

Змінні, визначені у функціях, називаються **локальними змінними**.

Локальні змінні створюються при вході в функцію і знищуються при виході з неї. Тому локальні змінні не можуть зберігати значення між викликами функцій. Єдиним винятком з цього правила є змінні, оголошені зі специфікатором static. Він змушує компілятор сприймати дану змінну як глобальну, але область видимості її, як і раніше є обмеженою.

Всі функції в Сі знаходяться на одному рівні видимості. Тобто неможливо визначити функцію вкладену в іншу функцію.

Аргументи функції

Якщо функція використовує аргументи, вона повинна оголошувати змінні, які отримують значення аргументів. Дані змінні називаються формальними параметрами функції. Вони поводять себе так само, як і звичайні локальні змінні, тобто створюються при вході в функцію і знищуються при виході з неї.

Зазвичай підпрограми можуть передавати аргументи двома способами.

Перший спосіб називається передачею через значення. Даний метод копіює вміст аргументу в формальний параметр функції. Зміни, зроблені в параметрі, не впливають на значення змінної, використовуваної при виклику.

Другий спосіб – передача за вказівником. В даному методі копіюється адреса аргументу. У підпрограмі адреса використовується для доступу до цього аргументу, що використовується при виклику. Тобто, зміни, зроблені в параметрі, впливають на вміст змінної, використовуваної при виклику.

```
#include <stdio.h>
int sqr (int x);

int main(void)
{
   int y = 10;
   printf("sqr(y)=%d; y=%d", sqr(y), y);
```

```
   return 0;
}

int sqr (int x)
{
   x = x*x;
   return x;
}
```

Вказівники передаються в функції як і звичайні значення. Нижче наведено функцію swap (), що обмінює значення двох цілочисельних аргументів:

```
void swap (int *x, int *y)
{
   int temp;
   temp = *x; //збереження значення за адресою x
   *x = *y; // запис y в x
   *y = temp; // запис x в y
}
#include <stdio.h>
void swap (int *x, int *y);
int main(void)
{
   int x=10, y=20; //В функцію swap потрібно передавати адреси x та y !!!
   swap(&x, &y); // &x- бере адресу x
   printf("x= %d; y=%d", x, y);
   return 0;
}
```

Рекурсія

В Сі функції можуть викликати самі себе. Функція є рекурсивною, якщо оператор в тілі функції викликає функцію, яка містить даний оператор.

Рекурсія є процесом визначення чого-небудь з використанням самої себе.

Простим прикладом є функція factr(), що обчислює факторіал цілого числа. Факторіал числа N є добутком чисел від 1 до N. Наприклад, факторіал 3 дорівнює 1 * 2 * 3 = 6.

```
int factr(int n) // рекурсивно
{
   int result;
   if(n<=1) return(1);
   result = n*factr(n-1); // рекурсивний виклик для (n-1)
   return(result);
}
```

Метод половинного ділення

Якщо на границях інтервалу [a..b] значення функції міняє знак, то в даному інтервалі міститься принаймні один розв'язок рівняння $f(x)=0$;

Наступним кроком знаходимо знак функції в точці $f((a+b)/2)$;

Беремо інтервал на границях котрого значення функції мають різний знак.

Процес проводимо до тих пір доки не буде знайдено точний розв'язок, або інтервал, який містить розв'язок, не зменшиться до заданої точності: $(b–a)<\varepsilon$, де, наприклад, $\varepsilon = 0.001$.

ПОРЯДОК ВИКОНАННЯ РОБОТИ

1. Опрацювати і засвоїти матеріал наведений в теоретичних відомостях.
2. Створити функцію, яка перевіряє, чи передане число просте.
3. Написати програму котра рекурсивно вираховує суму чисел від Вашого порядкового номера в журналі до 100.
4. Написати програму котра знаходить розв'язок рівняння $N*x+A = 0$ методом половинного ділення. N – Ваш порядковий номер в журналі, A – Ваш вік (повних років). Розв'язок знаходиться в інтервалі [–100..100].
5. В звіті навести копії екранів та написаний код.
6. Зробити висновки.