Ejercicio 1: Imprimir un Patrón de Asteriscos

Crea un programa que imprima un patrón de asteriscos en forma de cuadrado. La dimensión del cuadrado debe ser especificada por el usuario. Debes implementar una función llamada ImprimirPatron que acepte un parámetro entero que represente el tamaño del patrón. Esta función no debe tener valor de retorno (void).

Ejercicio 2: Saludar a una Persona

Desarrolla un programa que solicite al usuario ingresar su nombre y luego lo salude. Implementa una función llamada Saludar que acepte un parámetro de cadena (string) que represente el nombre de la persona. Esta función no debe tener valor de retorno (void).

Ejercicio 3: Convertir Grados Celsius a Fahrenheit
Escribe una función llamada ConvertirACelsius que tome una temperatura en grados Fahrenheit como parámetro y devuelva la temperatura equivalente en grados Celsius. Luego, crea un programa que solicite al usuario ingresar una temperatura en grados Fahrenheit, llame a la función y muestre la temperatura en grados Celsius.

Ejercicio 4: Menú de Opciones
Crea una función llamada MostrarMenu (no toma parámetros). Esta función debe mostrar un menú de opciones, por ejemplo, opciones numeradas para diferentes funciones o acciones. Luego, desarrolla un programa que llame a esta función y permita al usuario seleccionar una opción del menú.(Tampoco tiene return, es decir void)

Ejercicio 5: Calculadora de Experiencia
Desarrolla una función que tome un puntero a una variable de experiencia y un puntero a una variable de nivel en un juego. La función debe calcular el nivel en función de la experiencia y actualizar ambas variables.

Ejercicio 6:Guardar Puntuación Máxima
Desarrolla una función que tome un puntero a la puntuación actual del jugador y un puntero a la puntuación máxima registrada en un juego. La función debe actualizar la puntuación máxima si la puntuación actual es mayor.

Ejercicio 7: Pequeña Aventura
Queremos crear un pequeño juego de aventura de texto en la cual creará funciones que contemplen ciertas acciones del jugador, por ejemplo.

```cpp
// Función para tomar decisiones del jugador
char TomarDecision()
{
    char decision;
    std::cout << "¿Qué quieres hacer? (A)venturarte o (R)etirarte: ";
    std::cin >> decision;
    return decision;
}

int main()
{
    std::cout << "Bienvenido al juego de aventura de texto." << std::endl;

    bool jugando = true;

    while (jugando)
    {
        std::cout << "\nEstás en una encrucijada en el bosque misterioso." << std::endl;
        char decision = TomarDecision();

        if (decision == 'A' || decision == 'a')
        {
            std::cout << "Has decidido aventurarte más en el bosque." << std::endl;
            // Aquí puedes agregar más eventos o desafíos para la aventura.
        }
        else if (decision == 'R' || decision == 'r')
        {
            std::cout << "Decidiste retirarte del bosque. Fin del juego." << std::endl;
            jugando = false; // Termina el juego
        }
        else
        {
            std::cout << "Opción no válida. Intenta de nuevo." << std::endl;
        }
    }

    std::cout << "¡Gracias por jugar!" << std::endl;

}
```

Este ejercicio es integrador, implemente los tipos de datos, bucles, condicionales y funciones que conoce para lograr un programa sencillo y funcional.
(Opcional) Si desea puede incorporar el concepto de punteros, como en el siguiente ejemplo:

```cpp
// Función para tomar decisiones del jugador
char TomarDecision()
{
    char decision;
    std::cout << "¿Qué quieres hacer? (A)venturarte o (R)etirarte: ";
    std::cin >> decision;
    return decision;
}

// Función para actualizar la salud del jugador
void ActualizarSalud(int* salud, int cantidad)
{
    *salud += cantidad;
}

// Función para manejar el inventario del jugador
void GestionarInventario(std::string* inventario, const std::string& objeto)
{
    *inventario += objeto + " ";
}
```

```cpp
int main()
{
    std::cout << "Bienvenido al juego de aventura de texto." << std::endl;

    bool jugando = true;
    int salud = 100;
    std::string inventario;

    while (jugando)
    {
        std::cout << "\nEstás en una encrucijada en el bosque misterioso." << std::endl;
        char decision = TomarDecision();

        if (decision == 'A' || decision == 'a')
        {
            std::cout << "Has decidido aventurarte más en el bosque." << std::endl;

            // Simular un evento que afecta la salud del jugador
            int danio = rand() % 20; // Daño aleatorio entre 0 y 19
            ActualizarSalud(&salud, cantidad: -danio); // Reducir la salud del jugador

            // Simular encontrar un objeto y agregarlo al inventario
            std::string objetoEncontrado = "Poción";
            GestionarInventario(&inventario, objetoEncontrado);
        }
        else if (decision == 'R' || decision == 'r')
        {
            std::cout << "Decidiste retirarte del bosque. Fin del juego." << std::endl;
            jugando = false; // Termina el juego
        }
        else
        {
            std::cout << "Opción no válida. Intenta de nuevo." << std::endl;
        }

        // Mostrar la salud actual del jugador y su inventario
        std::cout << "Salud actual: " << salud << std::endl;
        std::cout << "Inventario: " << inventario << std::endl;

        if (salud <= 0)
        {
            std::cout << "Tu salud ha llegado a cero. Has perdido." << std::endl;
            jugando = false; // Termina el juego
        }
    }

    std::cout << "¡Gracias por jugar!" << std::endl;
```