Moar brainz!

Este ejercicio propone implementar un programa en C++ para crear y gestionar una **horda de zombies**. El objetivo principal es practicar la gestión dinámica de memoria, la creación de objetos en un solo bloque de memoria, y trabajar con punteros.

Detalles del ejercicio:

1. **Directorio de entrega:** Los archivos necesarios se deben guardar en la carpeta ex01/.
2. **Archivos a entregar:**
    a. Makefile (para compilar el programa).
    b. main.cpp (para realizar pruebas).
    c. Zombie.h o Zombie.hpp (declaración de la clase Zombie).
    d. Zombie.cpp (definición de la clase Zombie).
    e. zombieHorde.cpp (definición de la función zombieHorde).
3. **Funciones prohibidas:** Ninguna. Puedes usar cualquier función estándar de C++.
4. **Tarea principal:** Crear una función zombieHorde que:
    a. Reciba dos parámetros:
        i. int N: El número de zombies a crear.
        ii. std::string name: El nombre que compartirán todos los zombies.
    b. Asigne dinámicamente memoria para N objetos de tipo Zombie en un solo bloque.
    c. Inicialice cada zombie con el nombre proporcionado.
    d. Devuelva un puntero al primer zombie de la horda.
5. **Requisitos adicionales:**
    a. Implementa tus propias pruebas en main.cpp para asegurarte de que la función zombieHorde funciona correctamente.
    b. Usa la función announce() para que cada zombie de la horda "se presente".
    c. Al final, elimina la memoria asignada a los zombies para evitar **fugas de memoria**.

Cómo afrontar este ejercicio:

Paso 1: Diseña la clase Zombie

- Crea una clase Zombie con los siguientes elementos:
    - Un atributo privado para almacenar el nombre.
    - Un constructor para inicializar el nombre del zombie.
    - Un método announce() para que el zombie imprima un mensaje como:

"Zombie [nombre]: BraiiiiiiinnnzzzZ...".

Paso 2: Implementa la función zombieHorde

- Define la función zombieHorde en zombieHorde.cpp. Asegúrate de que:
  - Asigne dinámicamente memoria para un arreglo de N zombies.
  - Use un bucle para inicializar cada zombie con el nombre proporcionado.
  - Devuelva un puntero al inicio del arreglo.

Paso 3: Crea pruebas en main.cpp

- Implementa pruebas para:
  - Crear una horda de zombies.
  - Llamar al método announce() para que cada zombie se presente.
  - Liberar la memoria asignada con delete[] para evitar fugas.

Paso 4: Maneja la memoria

- Utiliza herramientas como valgrind (en Linux) o el depurador de tu entorno de desarrollo para verificar que no hay fugas de memoria.

En resumen:

1. Diseña una clase Zombie con un constructor y un método announce().
2. Implementa la función zombieHorde que cree e inicialice una horda de zombies.
3. Prueba la función en main.cpp, asegurándote de que los zombies se anuncien de forma correcta.
4. Libera la memoria para evitar fugas.

Zombie.hpp

Este archivo se utiliza para declarar la estructura básica de la clase y de la función antes de implementarlas en otro archivo (Zombie.cpp).

```
#ifndef ZOMBIE_HPP
# define ZOMBIE_HPP
...
#endif
```

- Estas líneas evitan que el archivo se incluya más de una vez durante la compilación. En este sentido, #ifndef ZOMBIE_HPP verifica si el identificador ZOMBIE_HPP no ha sido definido anteriormente.
- #define ZOMBIE_HPP lo define para evitar futuras inclusiones repetidas.
- El bloque #endif marca el final de esta protección.

Inclusión de las bibliotecas necesarias.

```
#include <cstdlib>
#include <iostream>
#include <string.h>
```

Estas bibliotecas proporcionan funciones y tipos que la clase o las funciones relacionadas pueden necesitar:

- <cstdlib>: Funciones generales de C, como manejo de memoria o generación de números aleatorios.
- <iostream>: Entrada y salida estándar (por ejemplo, para usar std::cout).
- <string.h>: Funciones de manejo de cadenas estilo C.
- <stdio.h>: Funciones estándar de entrada y salida de C necesaria para stdin.

Declaración de la clase Zombie

Visibilidad pública (public:):

- Los elementos en esta sección son accesibles desde fuera de la clase.
- void setName(std::string name);
  Método para establecer el nombre del zombie. Se espera que reciba un std::string como argumento y lo almacene en el atributo name.
- const std::string& getName(void) const;
  Método para obtener el nombre del zombie. Devuelve una referencia constante al atributo name para evitar copias innecesarias y asegura que no se pueda modificar.
- void announce(void);
  Método que hará que el zombie "se presente". Este método probablemente imprimirá algo como "Zombie [nombre]: Braiiiiiiinnnzzz...".
- ~Zombie(void);
  Destructor de la clase. Se ejecuta automáticamente cuando un objeto de tipo Zombie es destruido (por ejemplo, cuando la memoria dinámica se libera).

Visibilidad privada (private:):

- Los elementos en esta sección son accesibles solo desde dentro de la clase.
- std::string name; Atributo que almacena el nombre del zombie.

Declaración de la función zombieHorde

Esta función está declarada fuera de la clase Zombie.
Propósito: Crear y devolver una horda de zombies.
Parámetros:

int N: El número de zombies a crear.
std::string name: El nombre que compartirán todos los zombies.

Valor de retorno:

Devuelve un puntero al primer zombie de la horda, que estará alojado en un bloque de memoria dinámica.

Zombie.cpp

Este archivo implementa las funciones de la clase Zombie:

1. **Destructor (~Zombie):**
   - Muestra un mensaje cuando el zombie es destruido.
2. **setName(std::string name):**
   - Establece el nombre del zombie.
3. **getName() const:**
   - Devuelve el nombre del zombie sin modificar el objeto.
4. **announce():**
   - Hace que el zombie se presente con un mensaje personalizado.

Con esta implementación:

- Puedes crear zombies, darles nombres, y hacer que se anuncien.
- Cuando destruyas un zombie (o una horda de zombies), sabrás que fueron eliminados porque el destructor imprimirá un mensaje.

main.cpp

Este código corresponde al archivo principal del programa (main.cpp) y realiza las siguientes acciones:

- Solicita al usuario el nombre y el número de zombies a crear.
- Crea dinámicamente una horda de zombies usando la función zombieHorde.
- Hace que cada zombie de la horda "se presente".
- Libera la memoria asignada dinámicamente para evitar fugas de memoria.

Inclusión del archivo de cabecera

```
#include "Zombie.hpp"
```

Este archivo incluye las declaraciones de la clase Zombie y la función zombieHorde, que se utilizarán en este archivo.

Declaración de variables

```
std::string zombies_name;
int nbr_zombies;
```

- zombies_name: Variable para almacenar el nombre que compartirán todos los zombies.
- nbr_zombies: Variable para almacenar el número de zombies que el usuario desea crear.

Solicitar el nombre de los zombies

- Se utiliza un bucle infinito (while(1)) para asegurarse de que el usuario ingrese un nombre válido.
- getline(std::cin, zombies_name): Permite al usuario ingresar un nombre completo (con espacios si es necesario). El valor se guarda en zombies_name.
- std::cin.eof(): Verifica si se alcanzó el fin de la entrada (EOF), lo cual ocurre si el usuario presiona Ctrl+D (en Linux/macOS) o Ctrl+Z (en Windows).
- Si esto ocurre, se limpia el estado del flujo de entrada con std::cin.clear() y clearerr(stdin) para permitir que el programa continúe funcionando.
- zombies_name.empty(): Si el nombre ingresado está vacío, se muestra un mensaje pidiendo un nombre válido y el bucle se repite.
- Si el usuario ingresa un nombre válido, el bucle se interrumpe con break.

Solicitar el número de zombies

```
std::cout << " 🧟 Enter the number of zombies: ";
std::cin >>nbr_zombies;
```

Se solicita al usuario que ingrese un número entero (nbr_zombies), que representa la cantidad de zombies que se van a crear.

Crear la horda de zombies

```
Zombie *zombies_war = zombieHorde(nbr_zombies, zombies_name);
```

- Llama a la función zombieHorde, que crea dinámicamente un arreglo de nbr_zombies zombies, todos con el nombre zombies_name.
- zombies_war: Es un puntero que apunta al inicio del arreglo de zombies creado.

Los zombies se anuncian

```
For (int i = 0; i < nbr_zombies; i++)
        zombies_war[i].announce();
```

Recorre el arreglo de zombies utilizando un bucle for.

En cada iteración, se llama al método announce() del zombie en la posición i para que "se presente".
Por ejemplo, si hay 3 zombies llamados "George", la salida será algo como:

George 🧟 BraiiiiiiinnnzzzZ...
George 🧟 BraiiiiiiinnnzzzZ...
George 🧟 BraiiiiiiinnnzzzZ...

Liberar memoria dinámica

- Delete [] zombies_war;

- Delete [] se usa para liberar el bloque de memoria asignado dinámicamente por la función zombieHorde.

Esto destruye todos los objetos Zombie del arreglo y llama automáticamente al destructor de cada uno, imprimiendo mensajes como:

Zombie 🧟 George is dead 💤
Zombie 🧟 George is dead 💤
Zombie 🧟 George is dead 💤

Finalizar el programa

return (0);

Devuelve 0 para indicar que el programa se ejecutó correctamente.

En resumen:

- Este programa realiza los siguientes pasos:
- Solicita al usuario el nombre de los zombies (validando que no esté vacío).
- Solicita al usuario el número de zombies.
- Crea una horda de zombies con el nombre y número especificados.
- Cada zombie de la horda se presenta llamando a su método announce().
- Libera la memoria dinámica, destruyendo a todos los zombies, lo que activa su destructor.

Ejemplo de ejecución:

Entrada del usuario:
Enter the name of zombies 🧟 : George
🧟 Enter the number of zombies: 3

Salida del programa:
George 🧟 BraiiiiiiinnnzzzZ...
George 🧟 BraiiiiiiinnnzzzZ...
George 🧟 BraiiiiiiinnnzzzZ...

Zombie 🧟 George is dead 💤
Zombie 🧟 George is dead 💤
Zombie 🧟 George is dead 💤