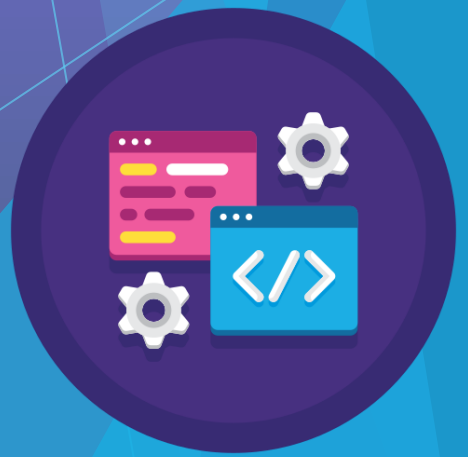
PROGRAMACIÓN

Ejemplo Práctico Integrador

P.U. PABLO AGÜERO – LIC. GABRIELA BARRAZA – LIC. JOSEFINA LOBO

Cátedra de Programación - PU_LI_II - FACET - UNT



Ejemplo Práctico Integrador:

- Estructuras de control
- Estructuras de iteración
- Funciones. Pasaje de parámetros.
- Arreglos
- Arreglos de caracteres. Funciones.
- Punteros
- Estructuras
- Asignación dinámica de memoria

Museo de historia – Voluntariado juvenil

El Museo de Historia organiza el **Programa de Voluntariado Juvenil 2024**, que permite a los jóvenes participar en distintas actividades de restauración y conservación. Cada actividad contará con un número variable de voluntarios.

Datos a tener en cuenta:

- **Nombre de la actividad:** restauración de documentos, limpieza de artefactos, visitas guiadas, catalogación de piezas, etc.
- **Área:** historia, arqueología, arte, conservación, etc.
- **Responsable a cargo:** persona encargada de supervisar la actividad.
- **Horario:** incluye hora de inicio y el lugar donde se desarrollará (depósito de piezas, sala de exposición, laboratorio).
- **Voluntarios:** incluye nombre, DNI y edad.

Informe/consulta requerida:

- Módulo que muestre la información de todas las actividades de voluntariado.
- Reporte que liste los voluntarios asignados a cada actividad.
- Informe que indique la edad promedio de los voluntarios por actividad.
- Reporte que muestre el total de actividades organizadas por un área que ingrese el usuario.

NOTA: Utilice asignación de memoria dinámica para el arreglo principal de actividades y para los arreglos de voluntarios.

Desarrollo:

-Análisis del problema

-Estructuras

-Comenzamos a programar

-Inicializaremos con valores de prueba

-Informes

-Liberar memoria antes de finalizar

Museo de historia – Voluntariado juvenil

El Museo de Historia organiza el **Programa de Voluntariado Juvenil 2024**, que permite a los jóvenes participar en distintas actividades de restauración y conservación. Cada actividad contará con un número variable de voluntarios.

Datos a tener en cuenta:

- **Nombre de la actividad:** restauración de documentos, limpieza de artefactos, visitas guiadas, catalogación de piezas, etc.
- **Área:** historia, arqueología, arte, conservación, etc.
- **Responsable a cargo:** persona encargada de supervisar la actividad.
- **Encuentro:** incluye hora de inicio y el lugar donde se desarrollará (depósito de piezas, sala de exposición, laboratorio).
- **Voluntarios:** incluye nombre, DNI y edad.

Informe/consulta requerida:

- Módulo que muestre la información de todas las actividades de voluntariado.
- Reporte que liste los voluntarios asignados a cada actividad.
- Informe que indique la edad promedio de los voluntarios por actividad.
- Reporte que muestre el total de actividades organizadas por un área que ingrese el usuario.

NOTA: Utilice asignación de memoria dinámica para el arreglo principal de actividades y para los arreglos de voluntarios.

Estructuras

Actividad

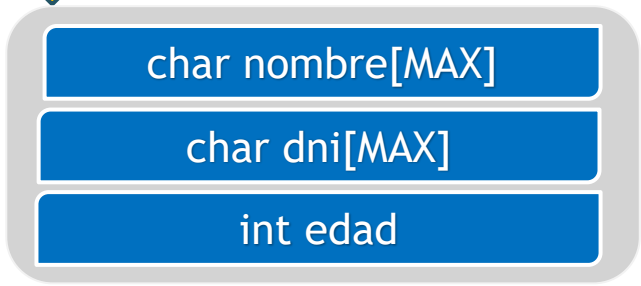
Cada actividad contará con un número variable de voluntarios.

Datos a tener en cuenta:

- **Nombre de la actividad**
- **Área**
- **Responsable a cargo**
- **Encuentro:** hora de inicio y lugar donde se desarrollará
- **Voluntarios:** incluye nombre, DNI y edad.

NOTA: Utilice asignación de memoria dinámica para el arreglo principal de actividades y para los arreglos de voluntarios.

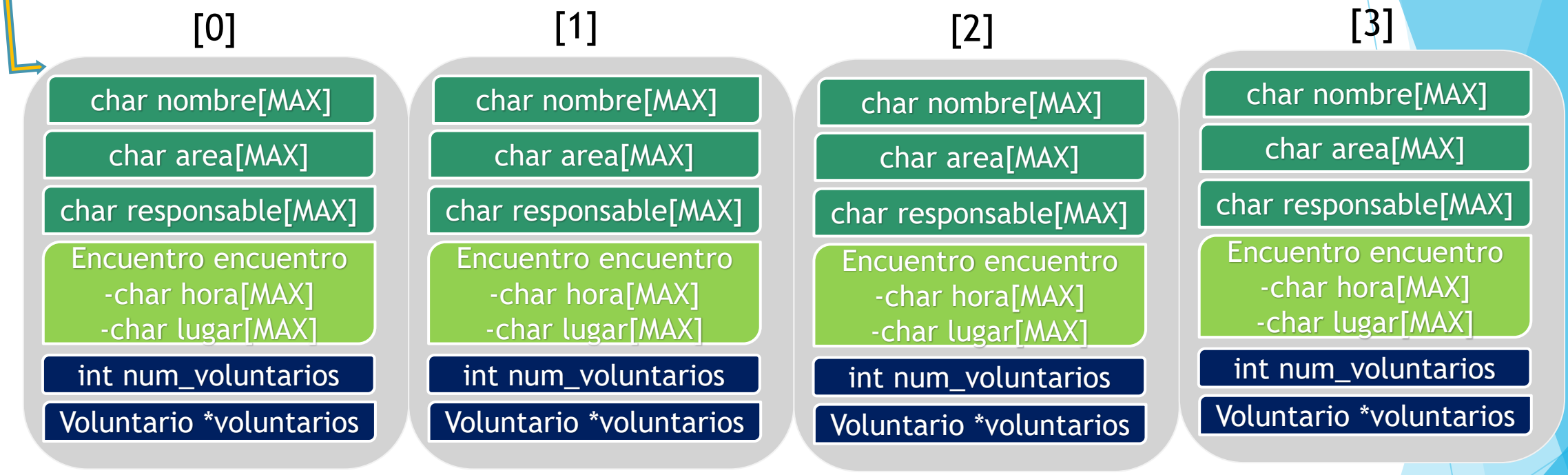
Voluntario



Estructuras

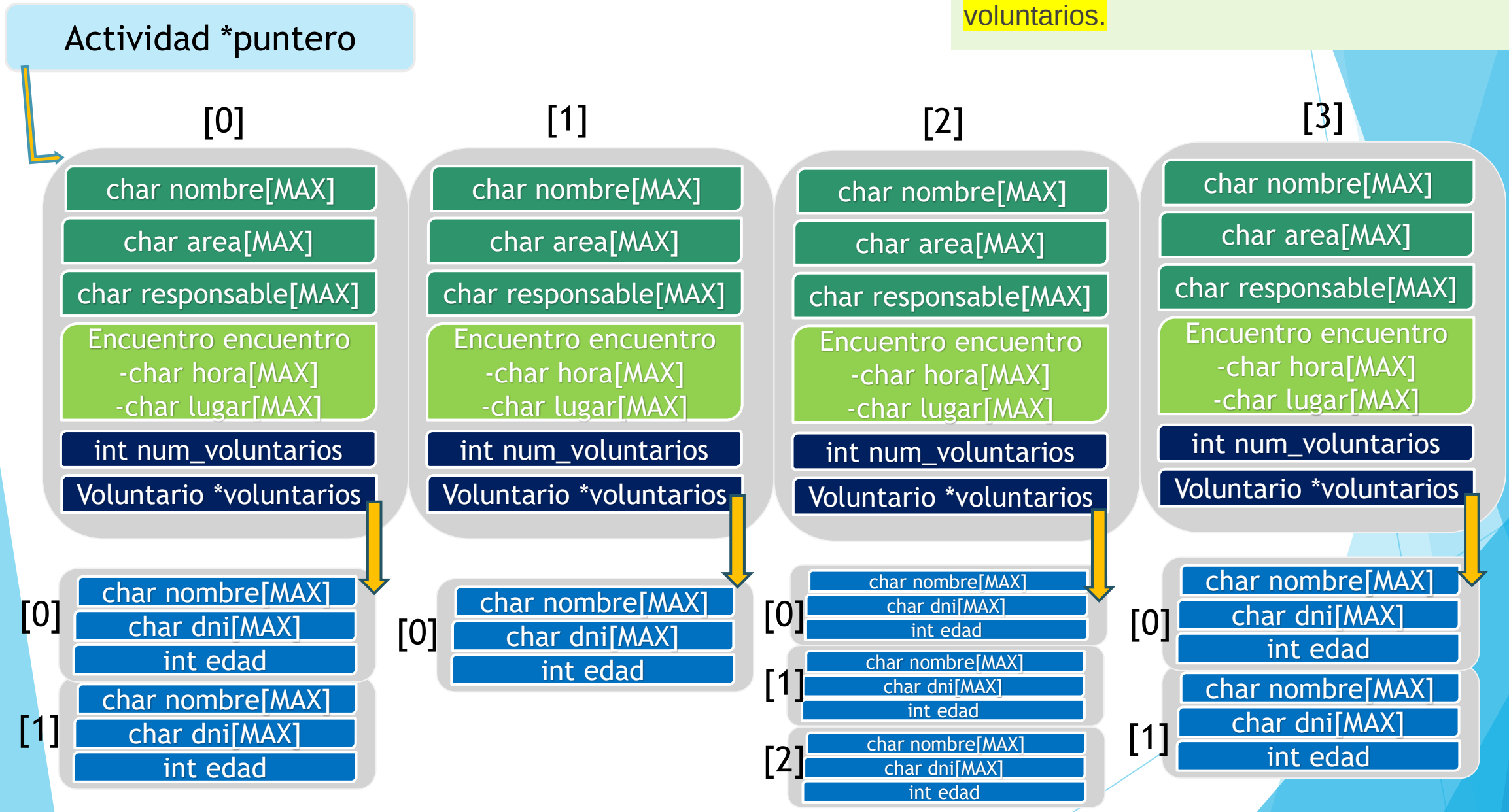
NOTA: Utilice asignación de memoria dinámica para el arreglo principal de actividades y para los arreglos de voluntarios.

Puntero principal



Estructuras

NOTA: Utilice asignación de memoria dinámica para el arreglo principal de actividades y para los arreglos de voluntarios.



Estructuras

```c
// Estructura para los encuentros
typedef struct {
    char horario[50];
    char lugar[50];
} Encuentro;


// Estructura para los voluntarios
typedef struct {
    char nombre[50];
    char dni[20];
    int edad;
} Voluntario;


// Estructura para las actividades
typedef struct {
    char nombre_actividad[100];
    char area[50];
    char responsable[50];
    Encuentro encuentro; // Estructura anidada para el horario y el lugar
    int num_voluntarios;
    Voluntario *voluntarios; // Puntero dinámico para los voluntarios
} Actividad;
```

Cada actividad contará con un número variable de voluntarios.

Datos a tener en cuenta:

- **Nombre de la actividad**
- **Área**
- **Responsable a cargo**
- **Encuentro:** hora de inicio y lugar donde se desarrollará
- **Voluntarios:** incluye nombre, DNI y edad.

NOTA: Utilice asignación de memoria dinámica para el arreglo principal de actividades y para los arreglos de voluntarios.

- Estructuras ✓
- Comenzamos a programar
- Inicializaremos con valores de prueba
- Informes
- Liberar memoria antes de finalizar

Estructuras

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

// Estructura para los encuentros
typedef struct {
    char horario[50];
    char lugar[50];
} Encuentro;

// Estructura para los voluntarios
typedef struct {
    char nombre[50];
    char dni[20];
    int edad;
} Voluntario;

// Estructura para las actividades
typedef struct {
    char nombre_actividad[100];
    char area[50];
    char responsable[50];
    Encuentro encuentro; // Estructura anidada para el horario y el lugar
    int num_voluntarios;
    Voluntario *voluntarios; // Puntero dinámico para los voluntarios
} Actividad;


int main() {
```

Cada actividad contará con un número variable de voluntarios.

Datos a tener en cuenta:

- **Nombre de la actividad**
- **Área**
- **Responsable a cargo**
- **Encuentro:** hora de inicio y lugar donde se desarrollará
- **Voluntarios:** incluye nombre, DNI y edad.

NOTA: Utilice asignación de memoria dinámica para el arreglo principal de actividades y para los arreglos de voluntarios.

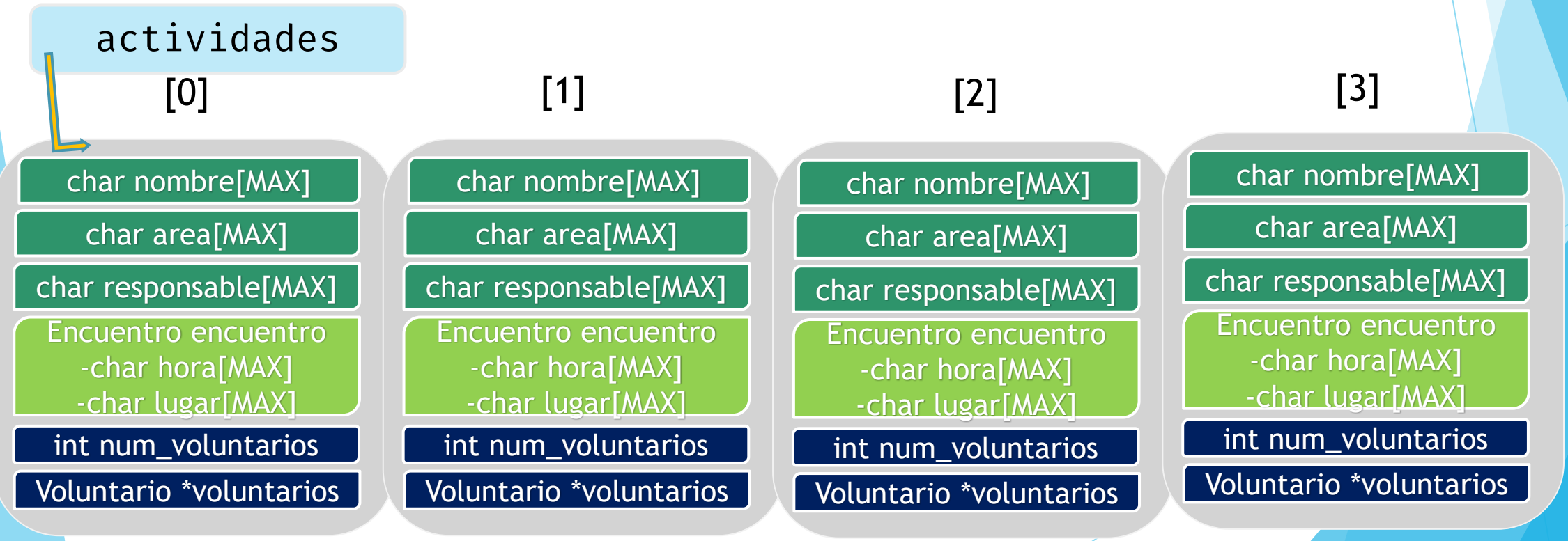
Inicializamos

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

NOTA: Utilice asignación de memoria dinámica para el arreglo principal de actividades y para los arreglos de voluntarios.

```
int main() {
    int num_actividades = 4; // Número fijo de activid
    Actividad *actividades = (Actividad *)malloc(num_actividades * sizeof(Actividad));
```

- Puntero actividades de tipo Actividad
- Arreglo dinámico de Actividad (tamaño 4 para precargar datos)



Inicializamos

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

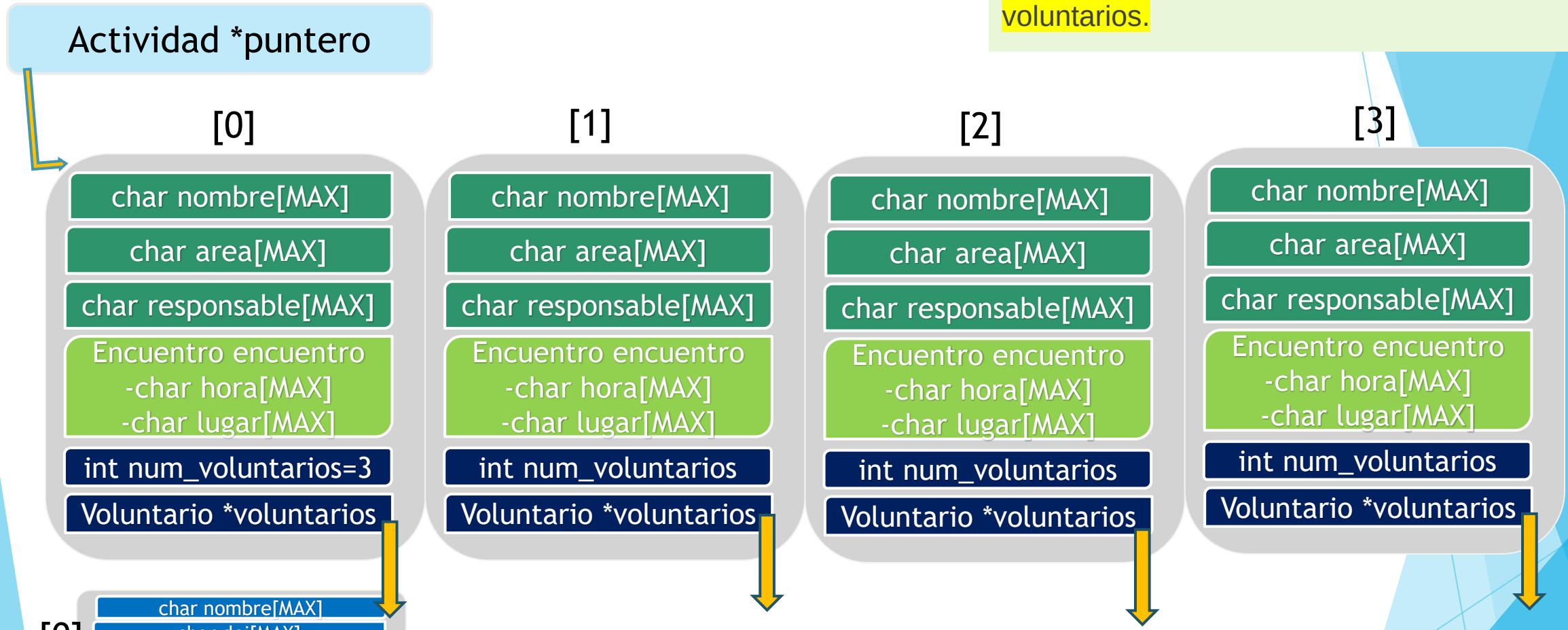
```c
int main() {
    int num_actividades = 4; // Número fijo de actividades precargadas
    Actividad *actividades = (Actividad *)malloc(num_actividades * sizeof(Actividad));

    // Precarga de datos con inicialización estática
    actividades[0] = (Actividad){
        "Restauración de documentos históricos",
        "Conservación",
        "Ana Pérez",
        {"10:00", "Sala de restauración"},
        3,
        (Voluntario *)malloc(3 * sizeof(Voluntario))
        };
```

- Puntero actividades
- Arreglo dinámico de Actividad (tamaño 4)
- Inicializamos el primer elemento del arreglo
- Creamos un arreglo dinámico de voluntarios (tamaño 3)

```c
typedef struct {
    char horario[50];
    char lugar[50];
} Encuentro;


typedef struct {
    char nombre[50];
    char dni[20];
    int edad;
} Voluntario;


typedef struct {
    char nombre_actividad[100];
    char area[50];
    char responsable[50];
    Encuentro encuentro;
    int num_voluntarios;
    Voluntario *voluntarios;
} Actividad;
```

Estructucturas

NOTA: Utilice asignación de memoria dinámica para el arreglo principal de actividades y para los arreglos de voluntarios.




```
actividades[0].voluntarios[0] = (Voluntario){"Lucas Martínez", "12345678", 21};
actividades[0].voluntarios[1] = (Voluntario){"María López", "87654321", 25};
actividades[0].voluntarios[2] = (Voluntario){"Julián Gómez", "11223344", 23};
```

```c
int main() {
    int num_actividades = 4; // Número fijo de actividades precargadas
    Actividad *actividades = (Actividad *)malloc(num_actividades * sizeof(Actividad));

    // Precarga de datos con inicialización estática
    actividades[0] = (Actividad){
         "Restauración de documentos históricos",
         "Conservación",
         "Ana Pérez",
         {"10:00", "Sala de restauración"},
        3,
        (Voluntario *)malloc(3 * sizeof(Voluntario))
        };

    actividades[0].voluntarios[0] = (Voluntario){"Lucas Martínez", "12345678", 21};
    actividades[0].voluntarios[1] = (Voluntario){"María López", "87654321", 25};
    actividades[0].voluntarios[2] = (Voluntario){"Julián Gómez", "11223344", 23};
```

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

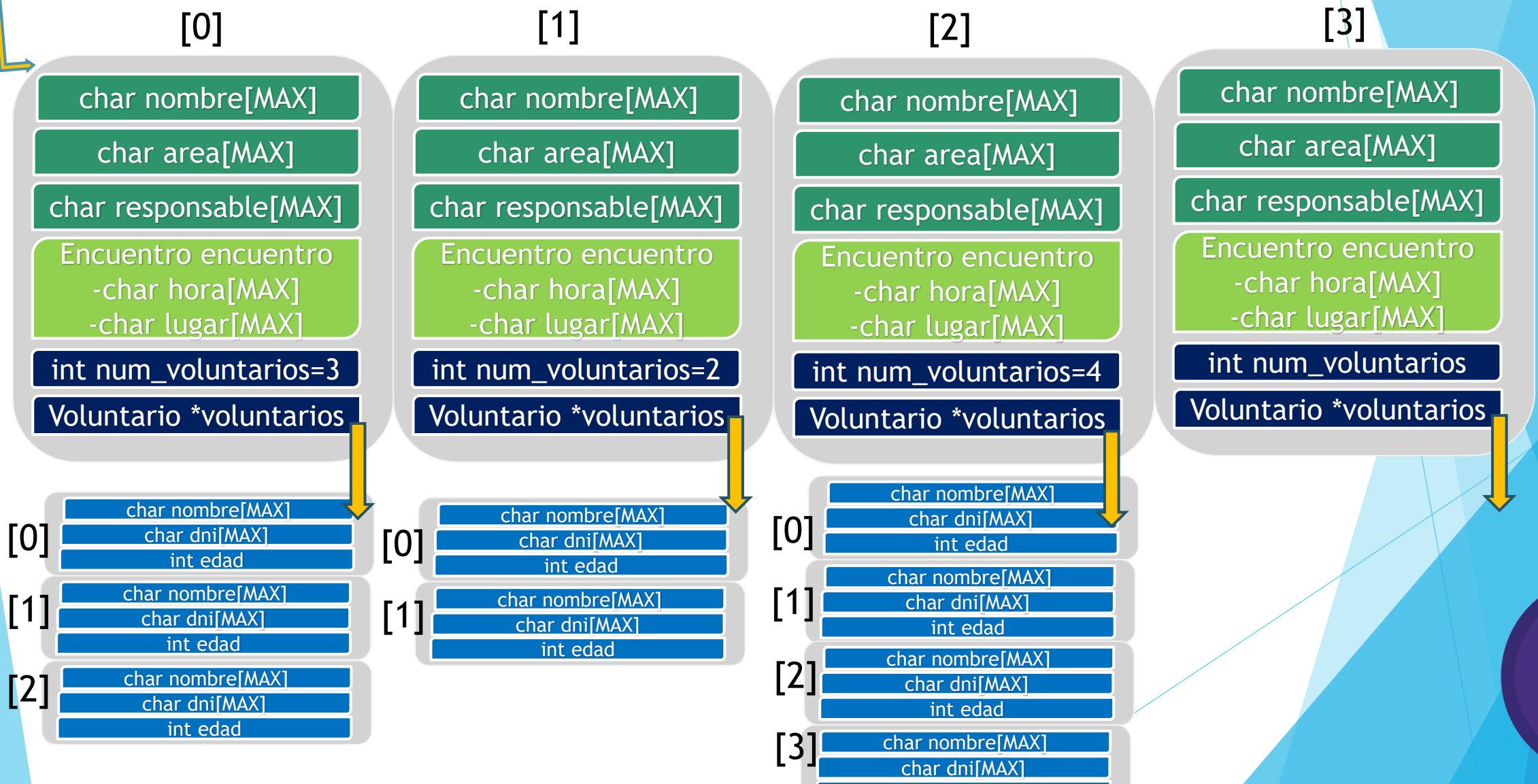
```c
typedef struct {
    char horario[50];
    char lugar[50];
} Encuentro;


typedef struct {
    char nombre[50];
    char dni[20];
    int edad;
} Voluntario;


typedef struct {
    char
nombre_actividad[100];
    char area[50];
    char responsable[50];
    Encuentro encuentro;
    int num_voluntarios;
    Voluntario *voluntarios;
} Actividad;
```

```
actividades[0] = (Actividad){"Restauración de documentos históricos","Conservación","Ana Pérez",{"10:00",
"Sala de restauración"},3,(Voluntario *)malloc(3 * sizeof(Voluntario))};
        actividades[0].voluntarios[0] = (Voluntario){"Lucas Martínez", "12345678", 21};
        actividades[0].voluntarios[1] = (Voluntario){"María López", "87654321", 25};
        actividades[0].voluntarios[2] = (Voluntario){"Julián Gómez", "11223344", 23};


actividades[1] = (Actividad){"Limpieza de artefactos arqueológicos","Arqueología","Roberto Sánchez",{"14:00",
"Depósito"}, 2,(Voluntario *)malloc(2 * sizeof(Voluntario))};
        actividades[1].voluntarios[0] = (Voluntario){"Sofía Ramírez", "55667788", 19};
        actividades[1].voluntarios[1] = (Voluntario){"Pedro Fernández", "66778899", 22};


actividades[2] = (Actividad){"Catalogación de piezas de arte","Arte","Gabriela Torres",{"09:00", "Sala de
exposición"}, 4,(Voluntario *)malloc(4 * sizeof(Voluntario))};
        actividades[2].voluntarios[0] = (Voluntario){"Laura García", "99887766", 24};
        actividades[2].voluntarios[1] = (Voluntario){"Tomás Díaz", "44556677", 20};
        actividades[2].voluntarios[2] = (Voluntario){"Valeria Cruz", "33445566", 26};
        actividades[2].voluntarios[3] = (Voluntario){"Diego Castro", "22334455", 23};

actividades[3] = (Actividad){"Visitas guiadas educativas","Historia","Hist. Elena Morales",{"10:00", "Museo
principal"},5,(Voluntario *)malloc(5 * sizeof(Voluntario))};
        actividades[3].voluntarios[0] = (Voluntario){"Camila Rodríguez", "77889900", 22};
        actividades[3].voluntarios[1] = (Voluntario){"Jorge Hernández", "55667788", 25};
        actividades[3].voluntarios[2] = (Voluntario){"Carla Mendoza", "11223344", 20};
        actividades[3].voluntarios[3] = (Voluntario){"Luis Navarro", "33445566", 23};
        actividades[3].voluntarios[4] = (Voluntario){"Fernanda Ortiz", "77889911", 24};
```

Estructucturas

NOTA: Utilice asignación de memoria dinámica para el arreglo principal de actividades y para los arreglos de voluntarios.

Actividad *puntero



- Estructuras ✓
- Comenzamos a programar ✓
- Inicializaremos con valores de prueba ✓
- Informes
- Liberar memoria antes de finalizar

Informes

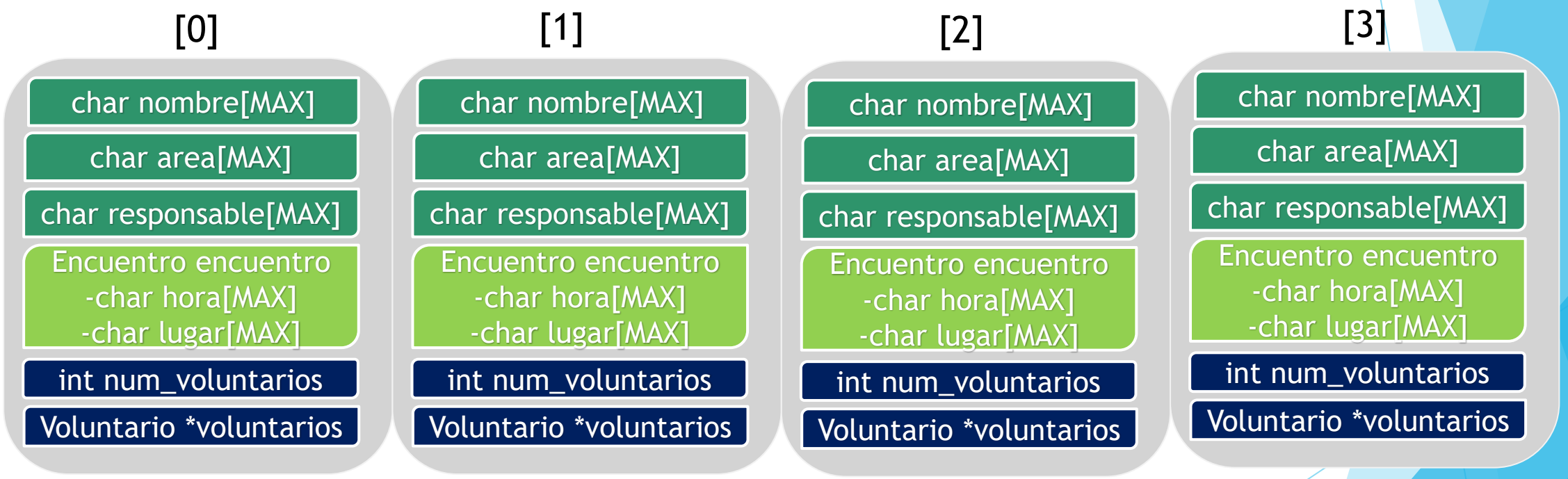
Informe/consulta requerida:

- Módulo que muestre la información de todas las actividades del voluntariado.
- Reporte que liste los voluntarios asignados a cada actividad.
- Informe que indique la edad promedio de los voluntarios por actividad.
- Reporte que muestre el total de actividades organizadas por un área que ingrese el usuario.

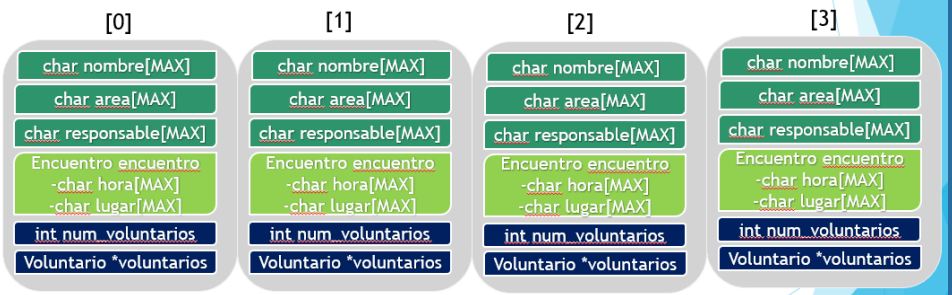
Informes

Informe/consulta requerida:

- Módulo que muestre la información de todas las actividades del voluntariado.



Informes



Informe/consulta requerida:

- Módulo que muestre la información de todas las actividades del voluntariado.

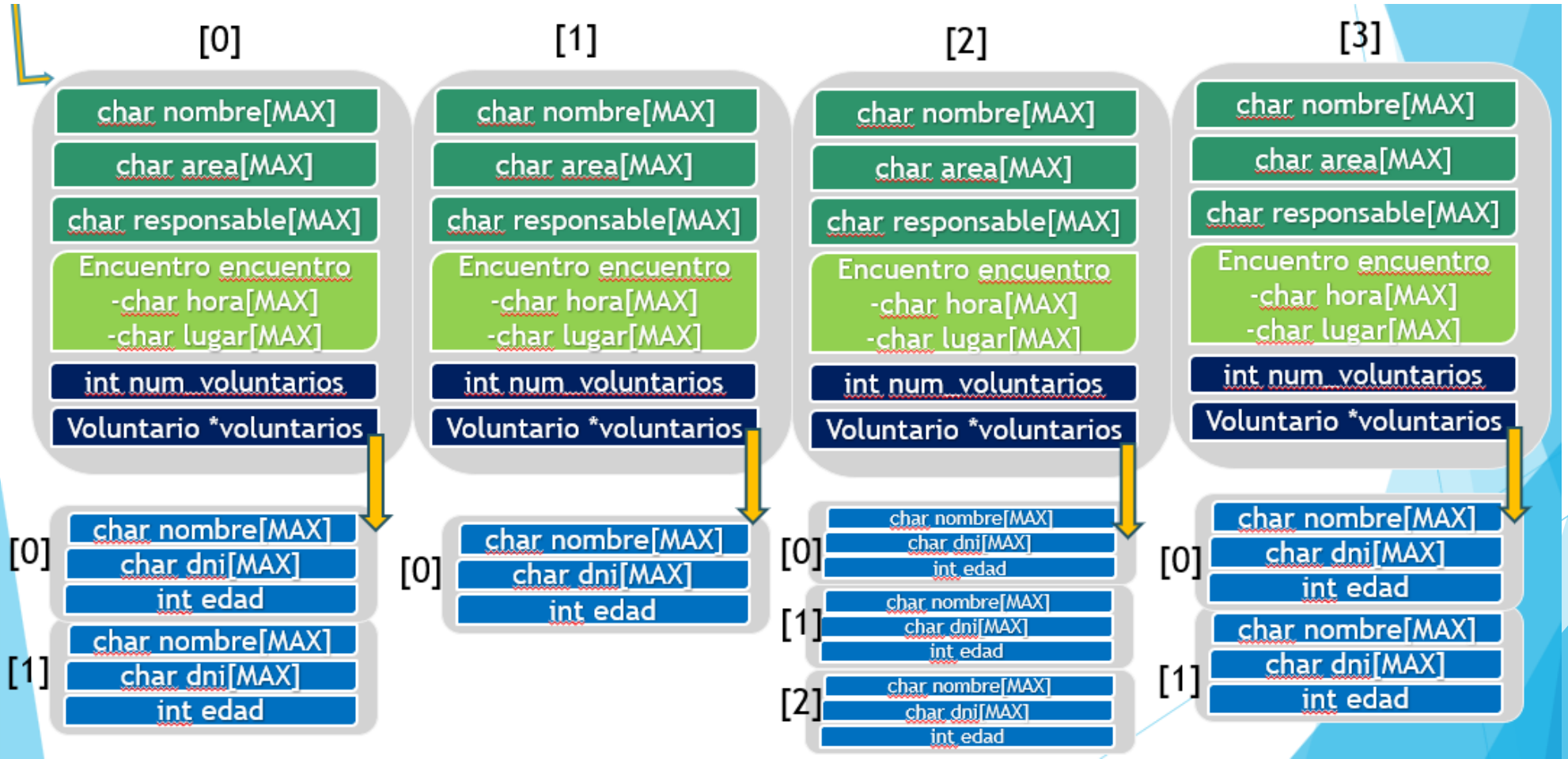
```
// Función para mostrar todas las actividades
void mostrarActividades(Actividad *actividades, int num_actividades) {
    printf("\n--- Información de todas las actividades ---\n");
    for (int i = 0; i < num_actividades; i++) {
        printf("\nActividad: %s\n", actividades[i].nombre_actividad);
        printf("Área: %s\n", actividades[i].area);
        printf("Responsable: %s\n", actividades[i].responsable);
        printf("Horario: %s\n", actividades[i].encuentro.horario);
        printf("Lugar: %s\n", actividades[i].encuentro.lugar);
        printf("Número de voluntarios: %d\n", actividades[i].num_voluntarios);
    }
}
```

```
MAIN invocamos a la función:
    // Mostrar información de las actividades
    mostrarActividades(actividades, num_actividades);
```

Informes

Informe/consulta requerida:

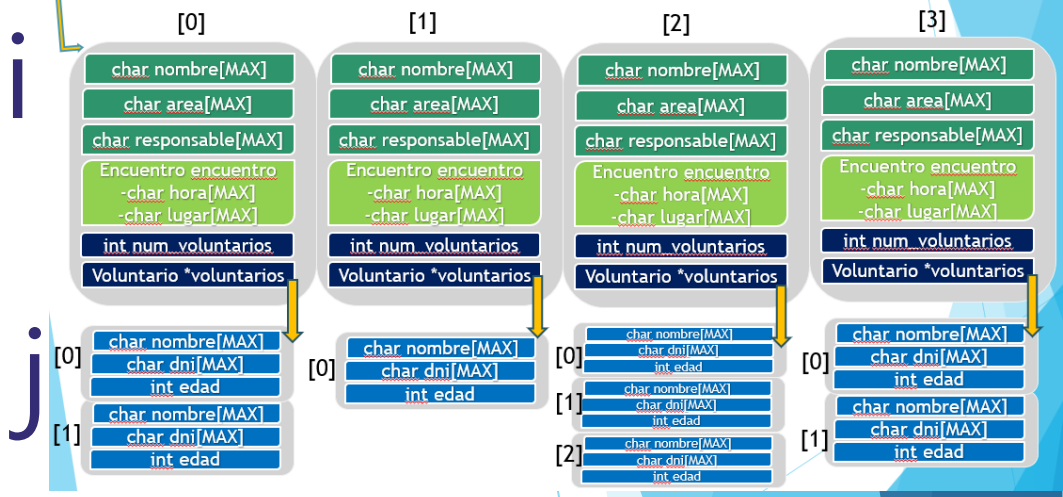
- Reporte que liste los voluntarios asignados a cada actividad.



Informes

Informe/consulta requerida:

- Reporte que liste los voluntarios asignados a cada actividad.



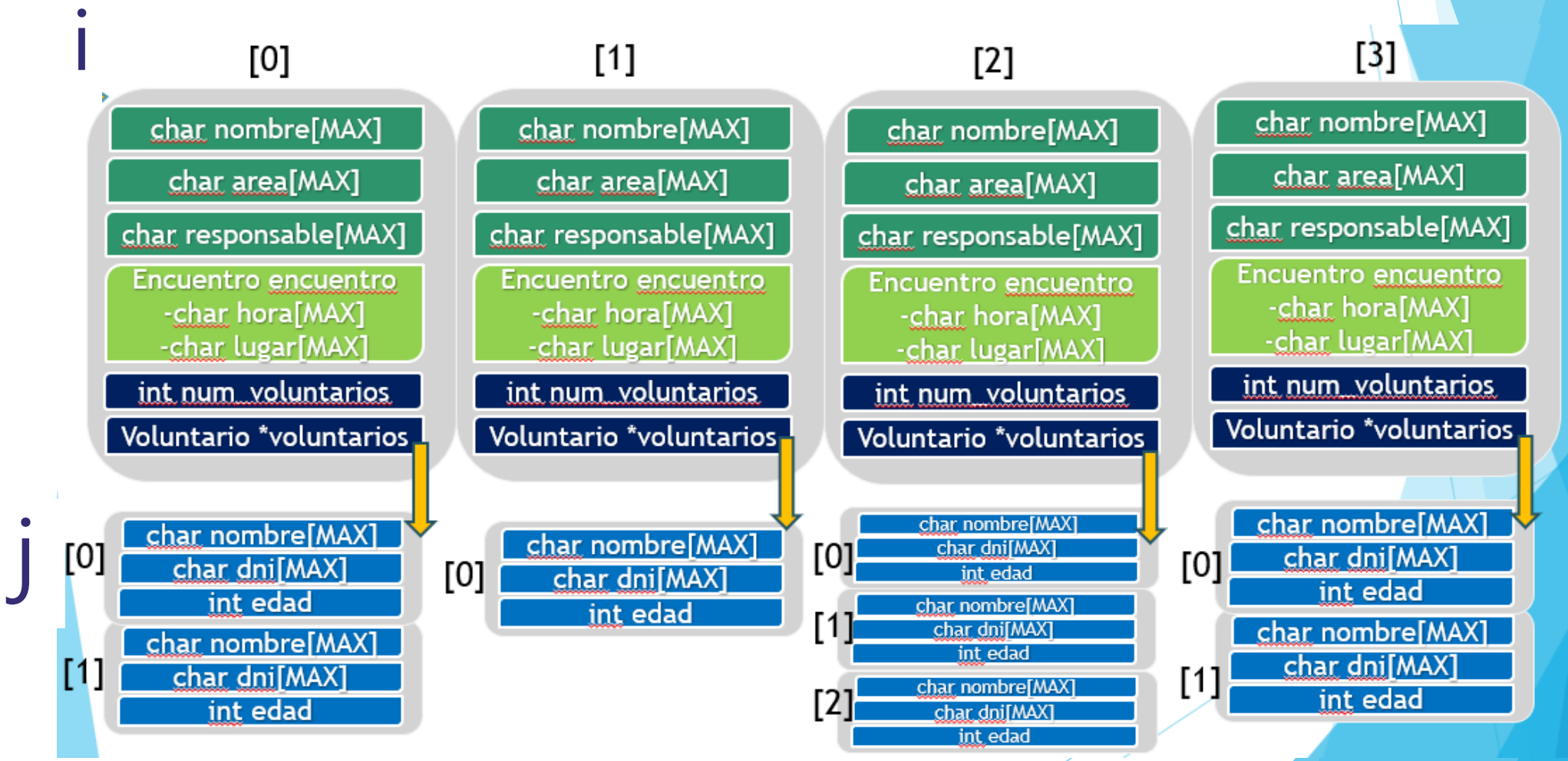

```
// Función para listar los voluntarios por actividad
void listarVoluntarios(Actividad *actividades, int num_actividades) {
    printf("\n--- Voluntarios asignados a cada actividad ---\n");
    for (int i = 0; i < num_actividades; i++) {
        printf("\nActividad: %s\n", actividades[i].nombre_actividad);
        for (int j = 0; j < actividades[i].num_voluntarios; j++) {
            printf("  Nombre: %s, DNI: %s, Edad: %d\n",
                   actividades[i].voluntarios[j].nombre,
                   actividades[i].voluntarios[j].dni,
                   actividades[i].voluntarios[j].edad);
        }
    }
}
```

```
MAIN invocamos a la función:
    // Mostrar información de los voluntarios
    listarVoluntarios(actividades, num_actividades);
```

Informes

Informe/consulta requerida:

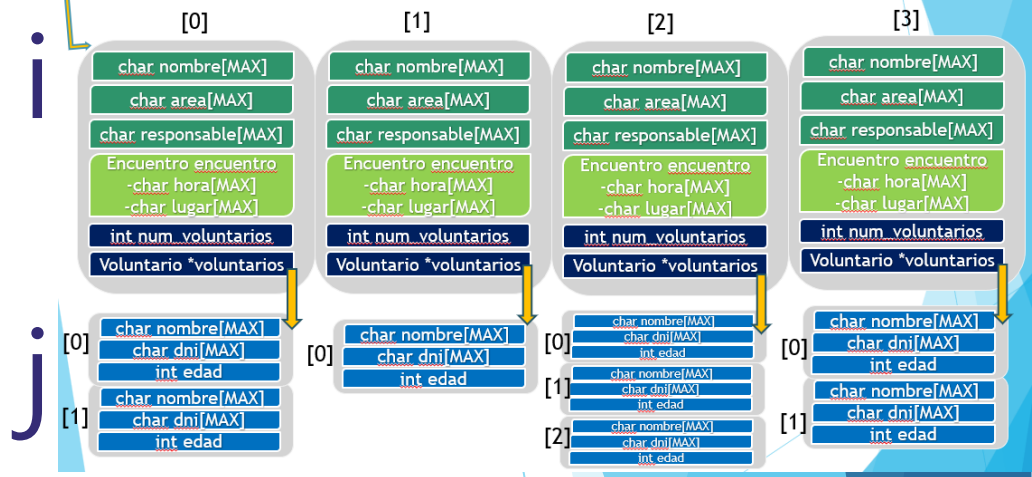
- Informe que indique la edad promedio de los voluntarios por actividad.



Informes

Informe/consulta requerida:

- Informe que indique la edad promedio de los voluntarios por actividad.

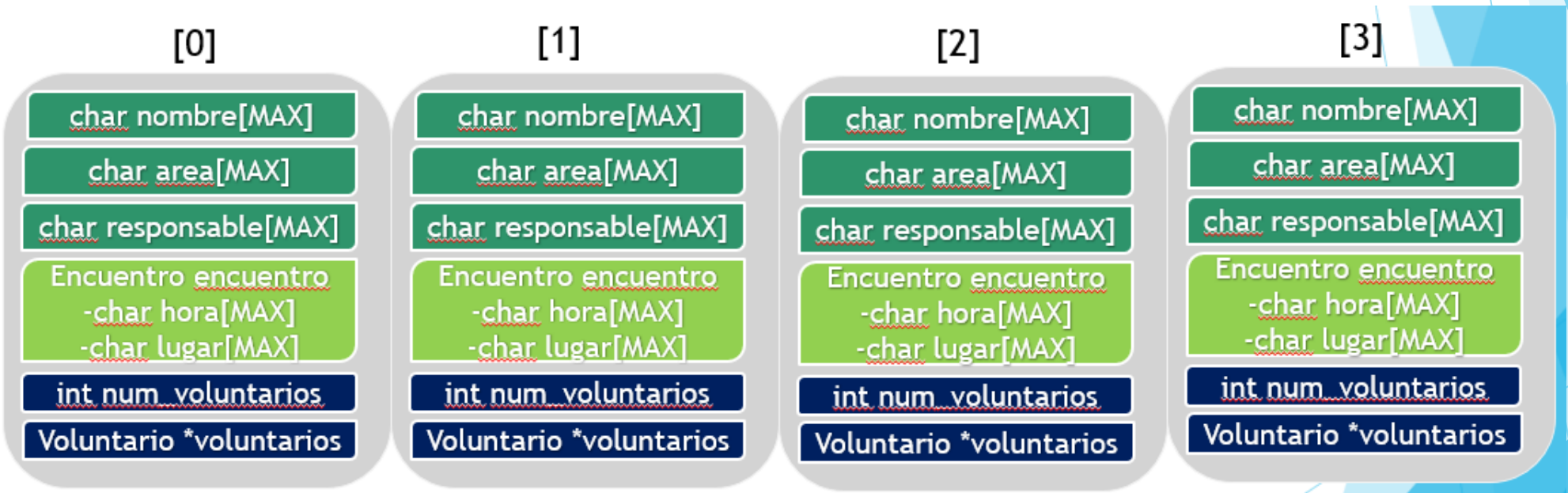



```
// Función para calcular la edad promedio por actividad
void calcularEdadPromedio(Actividad *actividades, int num_actividades) {
    printf("\n--- Edad promedio de voluntarios por actividad ---\n");
    for (int i = 0; i < num_actividades; i++) {
        int suma_edad = 0;
        for (int j = 0; j < actividades[i].num_voluntarios; j++) {
            suma_edad += actividades[i].voluntarios[j].edad;
        }
        float promedio;
         if(actividades[i].num_voluntarios > 0)
              {promedio = (float)suma_edad / actividades[i].num_voluntarios;}
         else{promedio= 0.0;}
        printf("Actividad: %s - Edad promedio: %.2f\n", actividades[i].nombre_actividad, promedio);
     }
}
```

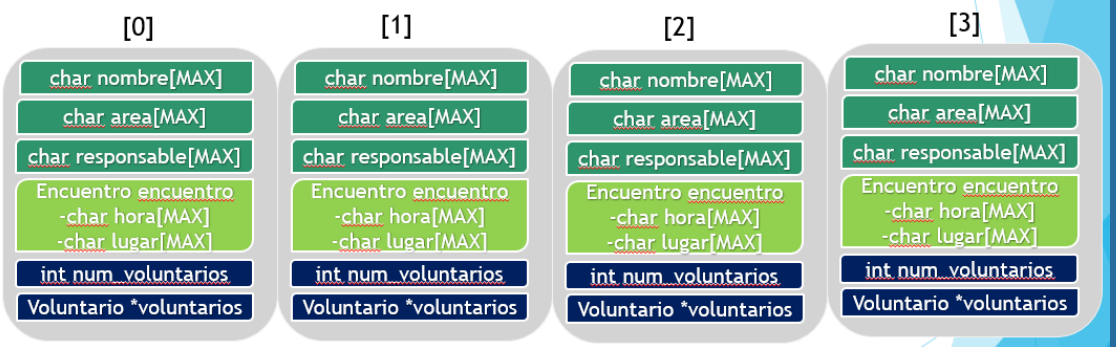
Informes

Informe/consulta requerida:

- Reporte que muestre el total de actividades organizadas por un área que ingrese el usuario.



Informes



Informe/consulta requerida:

- Reporte que muestre el total de actividades organizadas por un área que ingrese el usuario.

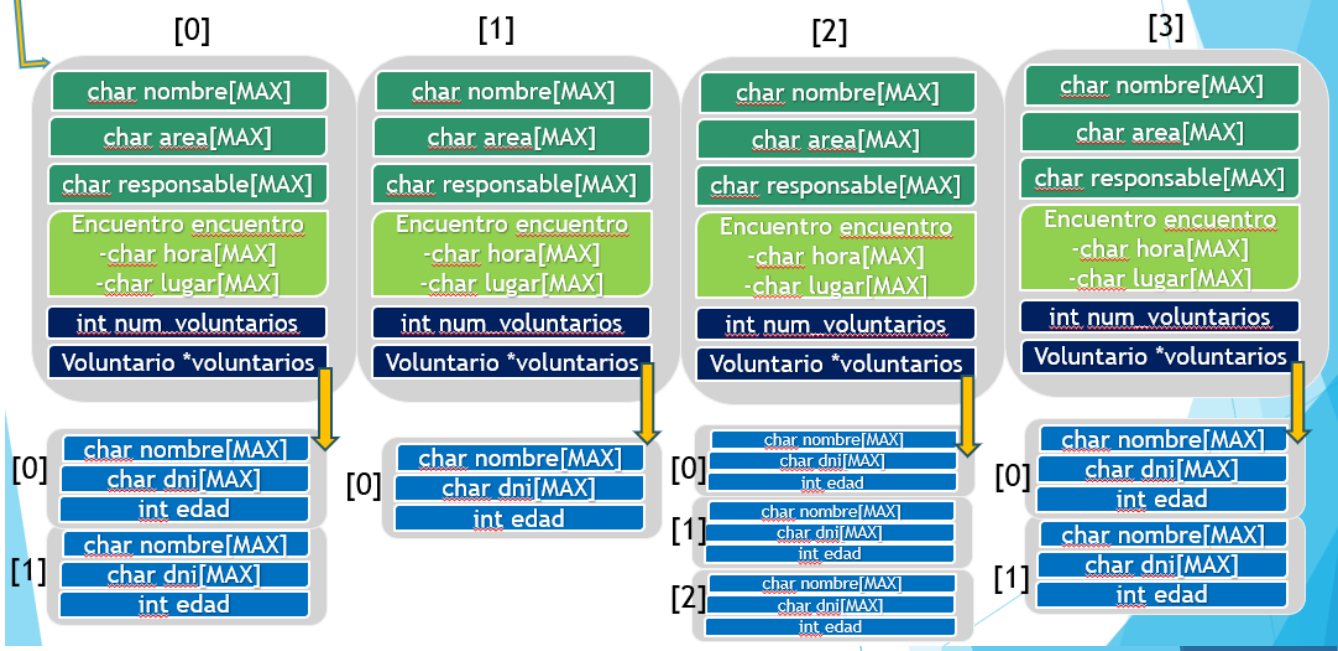
```c
// Función para contar actividades por área
void contarActividadesPorArea(Actividad *actividades, int num_actividades, char *area) {
    int contador = 0;
    for (int i = 0; i < num_actividades; i++) {
        if (strcmp(actividades[i].area, area) == 0) {
            contador++;
        }
    }
    printf("\nTotal de actividades en el área '%s': %d\n", area, contador);
}
```

```c
#include <string.h>
```

```c
MAIN :

  // Contar actividades por área
   char area_buscada[50];
   printf("\nIngrese un área para buscar actividades: ");
    fflush(stdin);
   scanf("%s", area_buscada);
   contarActividadesPorArea(actividades, num_actividades, area_buscada);
```

- Estructuras ✓
- Comenzamos a programar ✓
- Inicializaremos con valores de prueba ✓
- Informes ✓
- Liberar memoria antes de finalizar

Liberar memoria




```
// Liberar memoria dinámica
    for (int i = 0; i < num_actividades; i++) {
        // Liberar memoria dinámica arreglo voluntarios
        free(actividades[i].voluntarios);
    }
    // Liberar memoria dinámica, arreglo principal de actividades
    free(actividades);

    return 0;        //Fin del programa - Main
}
```

Realice nuevos reportes sugeridos:

- Muestre el promedio de edad de todos los voluntarios.
- Muestre todos los voluntarios que asistieron a eventos que se realizaron en un determinado lugar ingresado por el usuario.

¡Fin!