Algoritmos y Estructuras de Datos II

Apuntadores y memoria dinámica

Dr. Edgard I. Benítez-Guerrero

cursofei@gmail.com

Apuntadores

- Un apuntador es una variable cuyo valor es la dirección de memoria de otra variable
- Declaración de apuntadores en C:
  - <tipo de dato apuntado> *<identificador del apuntador>
- Ejemplos
  - Apuntador a un dato de tipo entero

```
int *p_edad;
```

  - Dos apuntadores a datos de tipo caracter

```
char *cad1, *cad2;
```

  - Apuntador a un dato de tipo punto flotante

```
float *ptr2;
```

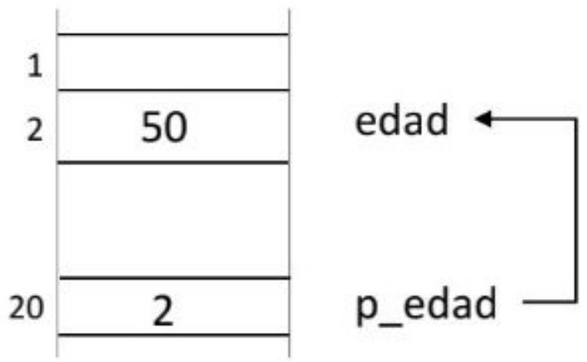
Apuntadores

- Para asignar una dirección de memoria a un apuntador se utiliza el operador de referencia &

```
int edad = 50;
int *p_edad = &edad;
```

- Para obtener el valor al que apunta un apuntador se utiliza el operador de indirección (*)

```
*p_edad
```



Asignación Dinámica de Memoria

- Asignación Dinámica es la creación de una variable mientras el programa está en ejecución. Para ello se usa el operador *new*.
- Las variables creadas con new son almacenados en una espacio de memoria libre llamado heap.
- Cuando son creados de esta manera, las variables permanecen en el heap hasta que son removidas de él con el operador *delete*.

Creando una variable

- Ejemplo: creación de una variable entero en el heap asignando su dirección a P.

- int *P = new int;

- El apuntador puede usarse como de costumbre

```
*P = 25;                    // assign a value

cout << *P << endl;
```

Operadores new y delete

- El operador new retorna la dirección a una variable recién creada
- El operador delete borra la variable y la deja no disponible.

```
Estudiante *pS = new Estudiante; // se crea un estudiante

// se usa pS …

delete pS;     // se elimina el estudiante y se devuelve
               // el espacio de memoria
```

Apuntadores inválidos

- Cuando una variable referenciada es borrada y luego se trata de usar el apuntador se genera un error de apuntador inválido

```
double *pD = new double;
*pD = 3.523;

delete pD;   // pD es inválido...

*pD = 4.2;   // error
```

Usando new en Funciones

- Atención con el alcance de las variables

```
void MySub()
{
  Estudiante * pS = new Estudiante;
  // se usa pS

  delete pS;        // borra el estudiante pS
}                   // pS desaparece
```

Fugas de memoria

- Un *memory leak* (o fuga de memoria) es una condición de error creada cuando una variable es dejada en el heap sin ningún apuntador conteniendo su dirección.
- Esto puede pasar si el apuntador al objeto sale fuera del alcance:

```
void MySub()
{
  Estudiante *pS = new Estudiante;
  //
} // pS sale del alcance
```

(el Estudiante al que hace referencia pS permanecerá en el heap)

Direcciones devueltas por funciones

- Una función puede retornar la dirección de un objeto que fue creado en el heap.

```
Estudiante *CreaEstudiante()
{
  Estudiante *pS = new Estudiante;
   return pS;
}
```

Direcciones devueltas por funciones (cont.)

- El que llama la función puede recibir una dirección y almacenarla en una variable apuntador.

- Estudiante *pS;
- pS = CreaEstudiante();
- // Ahora pS apunta a un Estudiante;

Creación de un Arreglo en el heap

- Es posible crear arreglos usando el operador new.
- Se debe eliminar cuando corresponda, incluyendo "[]" antes del nombre del arreglo en el delete correspondiente

```
void main() {
  double *samples = new double[10];

  // samples es un arreglo....
  samples[0] = 36.2;

  delete []samples;  //eliminación de un arreglo desde el heap
}                              // no se requiere poner en número de entradas.
```

Trabajo asignado

1. Apuntadores y memoria dinámica
    1. Escriba un programa que:
        a) Cree un arreglo dinámico de enteros cuyo tamaño *n* sea definido por el usuario al momento de ejecución
        b) Inicialice el arreglo con los números del 1 a n, uno por cada casilla del arreglo. Use apuntadores
        c) Imprima el arreglo. Use apuntadores
        d) Elimine el arreglo antes de terminar para regresar la memoria utilizada al heap.
    2. Reescriba el programa anterior para que la creación, la inicialización, la impresión y la eliminación del arreglo dinámico se haga en una función que se llame desde el programa principal (main)
2. Funciones recursivas
    1. Implemente la función recursiva Factorial
    2. Implemente la función recursiva Fibonacci