Estructura de Datos y Algoritmos 1

Práctico #2:

Punteros, vectores, strings, matrices.

De la clase anterior: Vectores

Declaración y creación

```
<tipo>* <nombre> = new <tipo>[<longitud>];
```

Ejemplo: `int* puntuaciones = new int[30];`

Destrucción (liberación de memoria)

```
delete [] <nombre>;
```

Ejemplo: `delete [] puntuaciones;`

Acceso a un elemento

```
<nombre>[<indice>]
```

Ejemplo: `puntuaciones[5];`

Sumario

1. Punteros
2. Traspaso de parámetros a funciones
3. Cadenas de caracteres (strings)
4. Vectores de strings

Punteros

Un ***puntero*** es una variable cuyo valor es una dirección de memoria.

Para cualquier tipo de datos **`T`**, **`T*`** es el tipo "puntero a **`T`**". Una variable de tipo **`T*`** puede tomar la dirección de una variable de tipo **`T`**.

Ejemplo:

```
int *pcount, count;
```

`pcount` almacenará la dirección en memoria donde se almacena un valor entero.

`count` almacenará directamente un valor entero.

Operadores de referencia y desreferencia

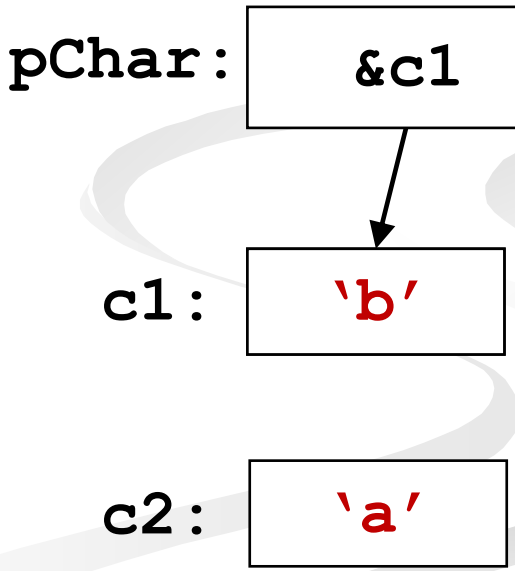
Operador de dirección o referencia (**&**): Unario, prefijo, retorna la dirección de una variable.

Operador de indirección o desreferencia (*****): Unario, prefijo, retorna el valor del objeto hacia el cual apunta su operando.

Ejemplo:

```
char c1 = 'a';
char *pChar = &c1;
char c2 = *pChar;

*pChar = 'b';
```



Inicialización de punteros

Es importante inicializar los punteros antes de utilizarlos, así evitamos que tengan valores arbitrarios.

Se pueden inicializar usando la constante **NULL**

Ejemplo:

```
int *ptr = NULL;
```

Punteros, referencias y traspaso de parámetros en las funciones

Por defecto, en C/C++, el traspaso de parámetros es por valor. La función recibe una copia del parámetro.

```
int main()
{
   int a = 4;
   int b = 2;
   swap(a,b);
   cout << a << endl;
   cout << b << endl;
   return 0;
}
```

Aquí se intercambian los parámetros, pero no las variables originales **a** y **b**.

```
void swap(int x, int y)
{
   int aux = x;
   x = y;
   y = aux;
}
```

Punteros, referencias y traspaso de parámetros en las funciones

En C/C++ existe el traspaso por referencia. Los parámetros por referencia se especifican mediante el operador &

```
int main()
{
   int a = 4;
   int b = 2;
   swap(a,b);
   cout << a << endl;
   cout << b << endl;
   return 0;
}
```

Una variable referencia actúa como un ***alias*** de la variable referenciada

```
void swap(int& x, int& y)
{
   int aux = x;
   x = y;
   y = aux;
}
```

Punteros, referencias y traspaso de parámetros en las funciones

Otra variante es que la función reciba direcciones de memoria de los parámetros.

```
int main()
{
   int a = 4;
   int b = 2;
   swap(&a,&b);
   cout << a << endl;
   cout << b << endl;
   return 0;
}
```

```
void swap(int* x, int* y)
{
   int aux = *x;
   *x = *y;
   *y = aux;
}
```

¿Qué imprimen y cuándo imprimen?

```
void mist1(int* ptr)
{
  if (ptr == NULL)
      return;
  cout << *ptr;
}
```

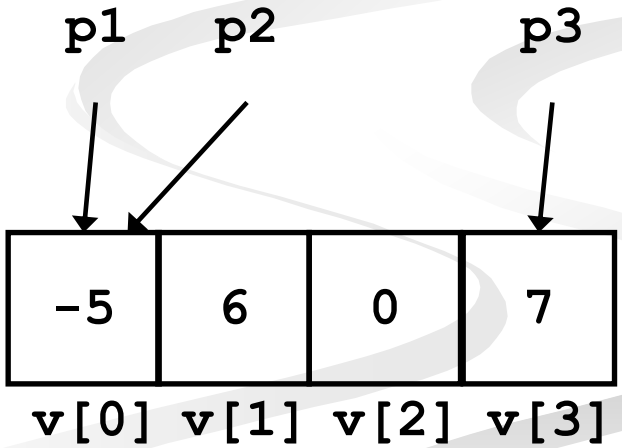
```
void mist2(int* ptr1,
           int* ptr2)
{
  if (ptr1 == ptr2)
      cout << *ptr1;
}
```

```
void mist3(int* ptr1,
           int* ptr2)
{
  if (ptr1 != NULL)
      && (ptr2 != NULL){
    if (*ptr1 == *ptr2)
      cout << *ptr1;
  }
}
```

```
void mist4(int* ptr)
{
  if (ptr != NULL)
      cout << (*ptr) + 1;
}
```

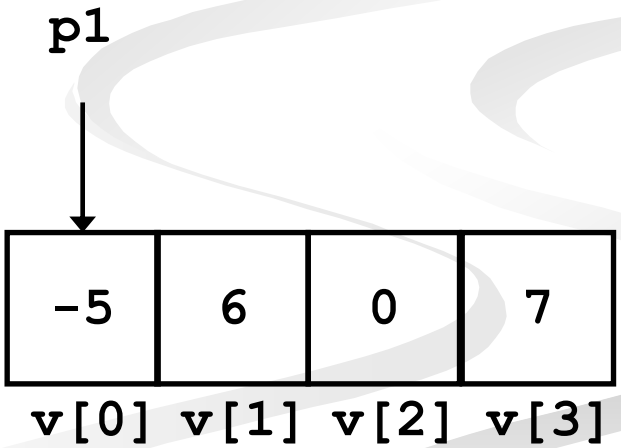
Punteros a elementos de un vector

```
int main()
{
   int* v = new int[4]{-5, 6, 0, 7};
   int *p1 = v;
   int *p2 = &v[0];
   int *p3 = &v[3];
   return 0;
}
```



Operadores de desplazamiento

```
int main()
{
   int* v = new int[4]{-5, 6, 0, 7};
   int *p1 = v;
   p1++;
   p1--;
   p1 += 3;
   p2 -= 2;
   return 0;
}
```

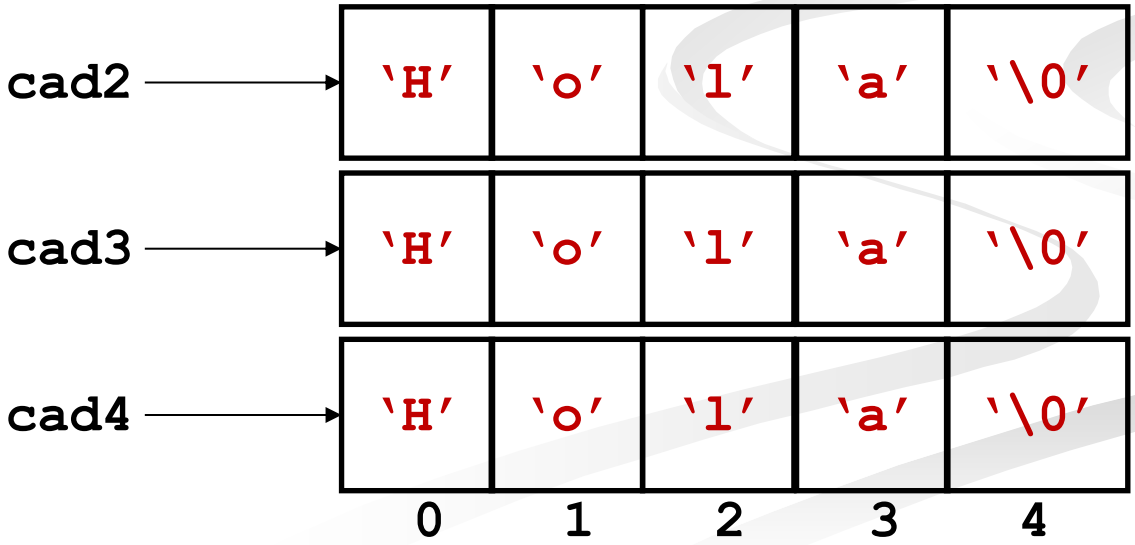


Cadenas de caracteres (strings)

Una cadena en C es un array de caracteres que debe terminar con el caracter `'\0'`

Declaración y creación

```
char* cad1 = new char[n];
char* cad2 = new char[5]{'H','o','l','a','\0'};
char* cad3 = new char[]{'H','o','l','a','\0'};
char* cad4 = "Hola";
```



Ejemplo: Largo de un string

```
int main()
{
  char* cad = new char[3]{'O','K','\0'};
  cout << largoCadena(cad) << endl;
  return 0;
}
```

```
int largoCadena(char* str)
{
  int largo = 0;
  while(*str != '\0'){
      largo++;
      str++;
  }
  return largo;
}
```

Vectores de strings

Un vector de strings es un array de elementos de tipo **char***

```
char** vectorStrings = new char*[3];
char* cad1 = "Hola";
char* cad2 = "Buenos";
char* cad3 = "días";
vectorStrings[0] = cad1;
vectorStrings[1] = cad2;
vectorStrings[2] = cad3;
```