Estructuras de Datos y Algoritmos (INFO053).
2° Sem., 2018.
Universidad Austral de Chile.
Profesor: Héctor Ferrada

Arreglos en C++

Estructuras de Control:
Antes de entrar a los arreglos revisemos las estructuras básicas de control en C/C++

if( )
Esta instrucción hace que se ejecuten unas sentencias u otras dependiendo del valor que toma la expresión que se esta evaluando. si es verdadera, entrara en su cuerpo y ejecutara las instrucciones de este.

Alternativa simple	Alternativa doble o multiple.

`if (condicion)`
`  instrucción1;`
`if (condicion) {`
`  instrucción 1;`
`  instrucción 2;`
`  instrucción 3;`
`}`
	`if (condicion)`
	`  instrucción1;`
	`else`
	`  instrucción2;`
	`if (condicion) {`
	`  Instrucción 1;`
	`  instrucción 2;`
	`} else if (condicion) {`
	`  Instrucción 3;`
	`  instrucción 4;`
	`} else {`
	`  instrucción 5;`
	`  instrucción 6;`
	`}`

`if (x>=0) {`
`  cout << ”Número no negativo”;`
`  cout << endl;`
`}`
	`if (x > 0)`
	`  cout << "x es positivo";`
	`else if (x < 0)`
	`  cout << "x es negativo";`
	`else`
	`  cout << "x es 0";`

switch( )
La sentencia *switch* selecciona una de entre múltiples alternativas (parecido al *if* múltiple). La forma general de esta expresión es la siguiente:

Estructura	Note en el siguiente. ejemplo que mientras no encuentre una instrucción *break* continua imprimiendo números.
`switch (expresión){`	`int i = 2;` `switch (i) {`

`case constante1:`
`  instrucciones;`
`  break;`
`case constante 2:`
`  instrucciones;`
`  break;`
`. . .`
`default:`
`  instrucciones;`
`}`
	`case 1: std::cout << "1";`
	`case 2: std::cout << "2";  //en esta comienza`
	`case 3: std::cout << "3";`
	`case 4:`
	`case 5: std::cout << "45";`
	`  break;  // aquí recién sale`
	`case 6: std::cout << "6";`
	`}`
	`std::cout << endl;`

El código anterior imprime: 2345

while( )

La sentencia *while* ejecuta su cuerpo de instrucciones infinitamente mientras la condición sea verdadera.

Estructura	Ejemplo. Suma todos los números positivos que se lean y sale cuando el número leído no sea positivo.

`while (condicion) {`
`  instrucción 1;`
`  ..............`
`  instrucción N;`
`}`
	`int num = 0;`
	`while (num >= 0) {`
	`  suma = suma + num;`
	`  cout << "Introduzca un numero: ";`
	`  cin >> num;`
	`}`

for( )

Esta estructura ejecuta primeramente y por única vez las instrucciones de inicialización, luego evalúa la condición y entra tantas veces mientras sea verdadera. Las instrucciones del tercer miembro, llamadas instrucciones de incremento, se ejecutan siempre inmediatamente después de finalizar un ciclo de iteración.

Estructura	Ejemplo. Imprime: 0123456789

`for(inicializa; condicion; incr.){`
`  instrucción 1;`
`  ..........`
`  instrucción N;`
`}`
	`for (int i=0; i<10; i++)`
	`  cout << i;`
	`cout << endl;`

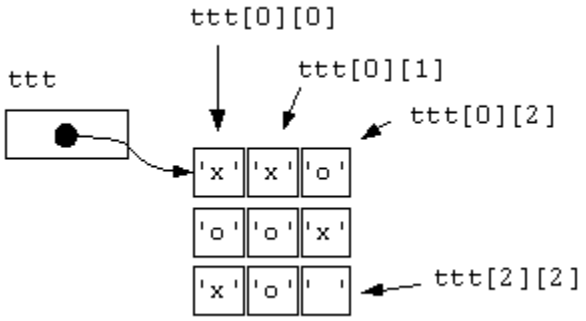
Algunos Recursos de programación:

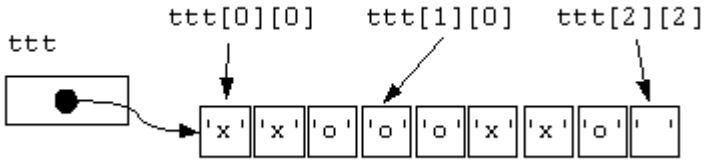
Librería estándar: http://www.cplusplus.com/reference/
Tutorial: http://www.cplusplus.com/doc/tutorial/

Arreglos (Arrays):

Un arreglo (*array)* es un segmento contiguo de memoria que almacena una secuencia de objetos del mismo tipo de datos. Los tipos de datos pueden ser los primitivos del lenguaje (booleanos, enteros, cadenas y flotantes) o datos definidos por el usuario. Un arreglo posee además dimensionalidad ya que son estructuras multidimensionales. Así, los arreglos de dos dimensiones pueden ser representados como matrices que almacenan elementos de un mismo tipo y están juntos físicamente en una porción de memoria.
Ejemplo de una matriz de 3x3 cuya memoria es solicitada y reservada en su misma declaración (memoria estática):

```
char ttt[3][3] = {{'x', 'x', 'o'}, // declara, define e inicializa la matriz
                  {'o', 'o', 'x'},
                  {'x', 'o', ' '}
                 };
```





Note en la figura como en memoria se acomodan los elementos de la matriz (arreglo en 2D). Las filas se ubican una a continuación de otra. Sin embargo, es posible que para ciertos arreglos creados de forma dinámica (no inicializados en su declaración y la memoria se solicita durante la ejecución del código), las filas no estén juntas en memoria, pero siempre todas las celdas de una fila estarán en segmentos contiguos de memoria.

Otros ejemplos:

```
int numeros[]={1,3,34,54};                //su capacidad es para 4 int
char alfabeto[5]={‘A’,’B’,’C’,’D’,’E’};   //arreglo de caracteres de largo 5

Char nombres[][40]={“pedro”, “pablo”, “luis”, “karina”, “lisa”};
//5 filas y sus columnas son de 40 caracteres.
```

```
int Coordenadas[2][2]={ {0,0},{1,1}};     //arreglo bidimensional de enteros
```

Los ejemplos anteriores inicializan inmediatamente a los arreglos, esto fuerza a que todos los elementos se encuentren físicamente juntos en un segmento contiguo de memoria. Podemos también crear el arreglo sin asignar los elementos de inmediato (se crean físicamente pero tienen basura):

```
int numeros[4];
char alfabeto[5];
char nombres[][40];
int coordenadas[2][2];
```

Creando arreglos de forma dinámica

Estos se crean en dos pasos:

1.- Se declara. Esto no los crea, solo existe una variable, por cada arreglo, que guarda la dirección de memoria en donde se localiza físicamente el arreglo. Por tanto inicialmente no tiene memoria asignada (valor NULL).

```
int* numeros;
char* alfabeto;
Char** nombres;
int** coordenadas;
```

2.- Se pide la memoria para el array. Después de este paso existen aunque tendrán basura.

```
numeros = new int[4];
alfabeto = new char[5];

nombres = new char*[5]
for(int k=0; k<5; k++)        // 5 arrays de largo 40
   nombres[k] = new char[40]; // No necesariamente las filas estarán contiguas

*coordenadas = new int*[2];   // array de punteros a enteros, de largo 2
cordenadas[0] = new int[2];   // array de enteros, de largo 2
cordenadas[1] = new int[2];   // array de enteros, de largo 2
coordenadas[0][0] = coordenadas[0][1] = 0;
coordenadas[1][0] = coordenadas[1][1] = 1;
```

Trabajando con Arreglos

Vamos a crear un array de enteros, A, de 50 celdas, con números aleatorios <= 20. Se pide:

1.- Un procedimiento que en un solo recorrido encuentre: el menor, el mayor y la media.
utilice rand() para generar los valores aleatorios (necesita la cabecera `#include <stdlib.h>`).
Ejemplo, dado un entero k:

```
int val = rand() % k; // crear un valor aleatorio entre 0 y k-1
```

2.- Ahora, apoyándose en un arreglo `Cont[0 .. max]`, cree una función que en un solo recorrido de A cuente (y devuelva) cuantos números repetidos hay en A, también encuentre cual es el número con más repeticiones y cuantas repeticiones posee.

Para esto, cree un programa **IntroArrays.cpp** que implemente las siguientes funciones/procedimientos, las cuales reciban al arreglo A y el valor de n y escriban las variables con los valores pedidos:

```
void menorMayorMedia(int *A, int n, int *min, int *max, float *media);
int repetidos(int *A, int n, int max, int *masRep, int *cantMRep);
```

Dentro de `repetidos()` debe de crear de forma dinámica el arreglo de contadores `Cont[0..max]`.

Comencemos por el cuerpo de la rutina main():

IntroArrays.cpp

```
#include <iostream>   //Biblioteca donde se encuentra la funcion cout
#include <stdlib.h>
using namespace std;  //uso del espacio de nombre std

void menorMayorMedia(int *A, int n, int *min, int *max, float *media);
int repetidos(int *A, int n, int max, int *masRep, int *cantMRep);

#define MAX 21;   // no es ni una variable ni una constante, es solo una expresion

int main()
{
   const int n = 50;  // n es una constante
   int A[n];
   int i, min, max;
   float media;

   // llenamos el array de enteros aleatorios < MAX
   for (i=0; i<n; i++)
      A[i] = rand()%MAX;

   // listamos el arreglo
   cout << "A[0," << n-1 << "] = " << endl;// cout imprime, endl es salto de linea
   for (i=0; i<n; i++)
      cout << " " << A[i];
   cout << endl;

   // obtenemos: menor, mayor y media de A
   menorMayorMedia(A, n, &min, &max, &media);

   cout << "Menor: " << min << endl;
   cout << "Mayor: " << max << endl;
   cout << "Media: " << media << endl;

   int rep, masRep, cantMRep;

   // contamos cuantos tienen al menos una repetición y obtenemos el mas repetido
   rep = repetidos(A, n, max, &masRep, &cantMRep);

   cout << "Hay " << rep << " números con repeticiones" << endl;
   cout << "El mas repetido es " << masRep << " con " << cantMRep  << "
repeticiones " <<  endl;

   return 0;
}
```

```
void menorMayorMedia(int *A, int n, int *min, int *max, float *media){

}

int repetidos(int *A, int n, int max, int *masRep, int *cantMRep){

   return 0;
}
```

Ejercicios

prog3.cpp
Cree una matriz de n por m, donde n y m son enteros leídos desde el teclado (entrada estandar). Luego recorra la matriz almacenando elementos aleatorios entre 100 y 999.

prog4.cpp
Cree un arreglo de números enteros aleatorios de dos dígitos, `A[0..n-1],` donde n sea leído por consola. Luego:

a.- Recorra la matriz y por cada celda imprima d1, d2 y d3 separadamente, que son los dígitos de cada número.

b.- Lea una posición p<n desde el teclado, y para x=A[p] cree una función que devuelva cual sería la posición final k de x si el arreglo estuviese totalmente ordenado (ascendentemente).

c.- Dado los mismos p, k y x del punto anterior, haga un procedimiento que intercambie los elementos de A particionando el arreglo de tal manera que, una vez puesto x=A[p] en su posición final k (como si A estuviese ordenado), se cumpla:
`A[i] <= x, 0<=i<=k`
`A[j] >  x, k<j<n.`

Argumente si su forma de particionar A puede considerarse eficiente considerando la cantidad de veces que le fue necesario recorrer el arreglo completo en su algoritmo.

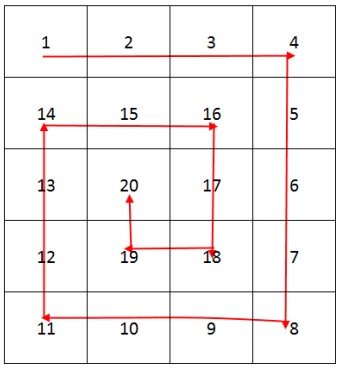
Más ejercicios en clases …

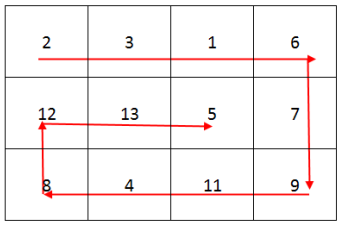
E1.- Codifique un programa que reciba n, c y m como argumentos y cree las matrices A de n x c y B de c x m con enteros aleatorios en el intervalo [-99..99]. Luego, en una función, realice la multiplicación y déjela en C de n x m.

```
void multiplicar(int *A, int *B, int*C, int m, int c, int n);
```

.- Codifique un programa que reciba n y m como argumentos y cree una matriz A de n x m con enteros aleatorios < 100. Realice:

E2.- Cree un procedimiento que liste la matriz según un recorrido de espiral, de afuera hacia adentro, en sentido horario (como indica la figura).





Output :	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Output:	2 3 1 6 7 9 11 4 8 12 13 5

E3.- Cree una función que calcula la matriz traspuesta de A y la deja en T:

```
void traspuesta(int *A, int m, int n, int *B);
```

E**4.-** Valide su resultado invocando dos veces a la función anterior, validando que la matriz devuelta es igual a la original (la traspuesta de la traspuesta es igual a la original).

```
bool esTraspuestaTdeA(int *A, int m_a, int n_a, int *T, int m_t, int n_t);
```

.- Codifique un programa que reciba n como argumento y cree un arreglo A de n enteros.

E5.- Genere una permutación aleatoria de n en A

E6.- Dada la permutación de n en A, cree una función que imprima todos los ciclos en A y retorne el número de ciclos.
Ejemplo:

```
                 0  1  2  3  4  5  6   7   8  9  10 11
Sea n=12 y A = {9, 6, 2, 4, 7, 0, 10, 11, 3, 5, 8, 1}

3 Ciclos: {0, 9, 5, 0} {1, 6, 10, 8, 3, 4, 7, 11, 1} {2}
```