Universidad de Morón
Escuela Superior de Ingeniería, Informática y Cs. Agroalimentarias

Asignatura:

(3362) Programación II

CLASE 11

Funciones
Vectores como Parámetros
Vectores de Struct como Parámetros

Funciones *printf* y *scanf* de entrada/salida en C

Prof. Lic. Sonia Zugna de Jausoro

Ejercicio Extra – Práctica de pasaje de parámetros

- Escribir una FUNCIÓN que dado un monto *(probar con el valor 100)*:
  1) lo devuelva incrementado en un 10 porciento (sobre sí mismo)
  2) devuelva qué porcentaje es ese monto del Total de Ventas
  3) devuelva cuánto le falta a ese Total de Ventas para llegar a 1 millón

Se solicita que la función sea de **tipo void**.

Usos de los parámetros:

- **Entrada/Salida**
- **Entrada solamente**
- **Salida solamente**

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
using namespace std;

// Prototipo  de funcion
void f4(float &monto,float totVentas,float &porc,float &cuanto);

int main (void)
{
float monto, porcentaje, dif, totVentas, resu;
   totVentas=700;
  monto=100;

  //  Llamada a la funcion
  f4 (monto,totVentas,porcentaje,dif);

  cout << "el monto incrementado en un 10% es: " << monto << endl;
  cout << "el total de ventas es  : " << totVentas << endl;
  cout << "el porcentaje es : " << porcentaje << endl;
  cout << "la diferencia es : " << dif << endl << endl;
  cout << " ********* la direccion de la variable monto es : " << &monto << endl;   // pruebo a imprimir la dirección

  getch();
  return(0);
}

// Funcion f4
void   f4  (float & monto,  float totVentas,  float & porc,  float & cuanto)
{
 monto=monto+(10*monto/100);
 porc=(100*monto/totVentas);
 cuanto=1000000-monto;
}
```

Ejercicio Extra – Imprime pino navideño

```
#include <stdio.h>
#include <iostream>
#include <conio.h>
#include <iomanip>
using namespace std;

void asteriscos (int cant)
 {
    for ( int i=0; i<cant;  i++)
    cout << "*";
    cout<<endl;
 }

int main (void)
{
   int cant;
   int cuanto;
    int filas, i;
   cout<<"Ingrese cantidad de filas del pino: ";
   cin>>filas;

for (cant=filas; cant>0; cant-=2) {
   cuanto = ((filas +1) - cant);
   cout << setw (cant / 2);  //  con esta sentencia se dejan los blancos delante de la línea de asteriscos
   asteriscos(cuanto);
   }
   for(i=0;i<6;i++) {
     cout<<setw((filas / 2)-2);
     asteriscos(5);
   }
 getch();
 return 0;
}
```

Paso de Vectores como Parámetros

En C/C++:

Los arreglos en general son pasados como parámetros **por referencia**.

El **nombre** del arreglo es la *dirección del primer elemento del arreglo*.

Notar que:

Un **elemento** cualquiera de un arreglo, **puede ser pasado** a una función **por valor o por referencia**, tal y como se hace con una variable simple.

Sintaxis

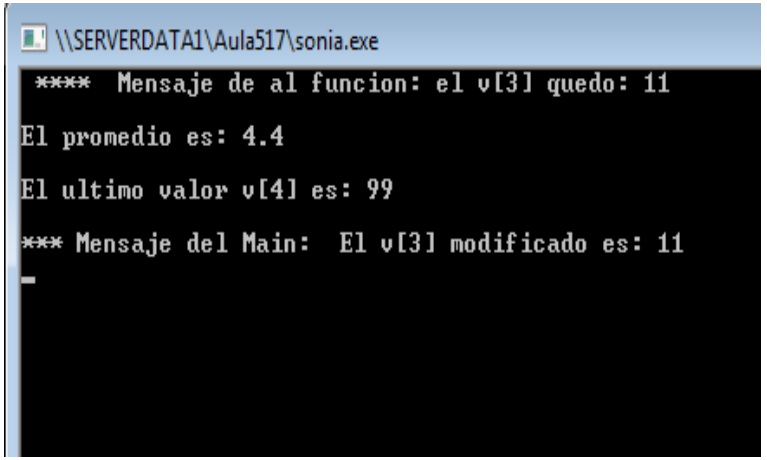
```
float promedio ( int long, float vec[] )     // Definición de la función Promedio
                                             // en vec se incluyen los corchetes
{
…    // calculo  y return del promedio
}
```

```
void main ()
{
int n;
float prom;
float vector[100];
…
prom = promedio(n, vector);

// Esta llamada pasa como parámetros actuales la longitud del vector y el vector.
// En la llamada → No se incluyen los corchetes
…
}
```

```
#include<stdio.h>
#include<iostream.h>
#include<conio.h>

float promMod(int, float [], float &);     // prototipo

int main(void)
{
   float resu;

   float v[]={2.2, 4.4, 6.6, 8.8, 0.0};

    resu = promMod(5, v, v[4]);    // llamada

   cout<<"El promedio es: "<<resu<<endl;
   cout<<"El ultimo valor v[4] es: "<< v[4] <<endl;
   cout<<«*** Mensaje del Main:  El v[3]  modificado es: "<< v[3] <<endl;

   getch();
   return 0;
}

float promMod (int lon, float v[], float &x)    // función
{
    float sum = 0, p;
    int i;
    for(i=0;i<lon;i++)
       sum = sum + v[i];
    p = sum/lon;
    v[3]=v[2]+v[1];
    cout<<"Mensaje de  la funcion: el v[3] quedo: "<<v[3]<<endl;
    x=99;
    return p;
}
```

Ejercicio:

FUNCIÓN: Calcula el promedio del vector, modifica el v[3], y por referencia recibe el elemento de la última posición y lo pone en 99.



Recordatorio para usar *printf* en el ejercicio siguiente.

Funciones *printf* y *scanf* de entrada/salida en **C**

Incluyen una **cadena de texto** para indicar diferentes tipos y opciones de formato y justificación.

El formato ***completo*** de los **modificadores** es el siguiente:

% [signo] [longitud] [.precisión] modificadorTipoDato

***Signo*:** indicamos si el valor se **ajustará** a la izquierda, en cuyo caso utilizaremos el signo menos, o **a la derecha (por defecto)**.

***Longitud*:** especifica la longitud máxima del valor que aparece por pantalla. Si la longitud es menor que el número de dígitos del valor, éste aparecerá ajustado a la izquierda.

***Precisión*:** indicamos el número máximo de decimales que tendrá el valor.

Los **modificadores** de Tipo de dato:

c Un único caracter
d Un entero con signo, en base decimal
u Un entero sin signo, en base decimal
o Un entero en base octal
x Un entero en base hexadecimal
e Un número real en coma flotante, con exponente
f Un número real en coma flotante, sin exponente
s Una cadena de caracteres
p Un puntero o dirección de memoria

Ejercicio

```
//Producto escalar de dos vectores

#include<conio.h>
#include<stdio.h>

using namespace std;

double productoEscalar(double v1[], double v2[], int d);

int main()
{
    int largo=3;
    double vector1[]={1,2,3};
    double vector2[]={4,5,6};
    double resultado=productoEscalar(vector1,vector2,largo);
    printf("(%5.2f,%5.2f,%5.2f).(%5.2f,%5.2f,%5.2f)=%5.2f\n",
                                        vector1[0],vector1[1],vector1[2],
                                        vector2[0],vector2[1],vector2[2],
                                        resultado);

    getch();
    return 0;
}

double productoEscalar(double v1[], double v2[], int largo)
{
       double resultado=0;
       int i;
       for (i=0;i<largo;i++)
       {
           resultado+=v1[i]*v2[i];
       }
       return resultado;
}
```

- En cabecera de función, en el parámetro **formal** de un arreglo→ **no se especifica su longitud**

```
// Búsqueda del Máximo único y cálculo del Promedio

#include<iostream>  #include<conio.h>  #include<stdio.h>  using namespace std;

int buscarMax (double valores[], int cantElem)
{
   int maxPos=0;
   for (int i=1   ; i<cantElem ; i++)
   {
      if (valores[i]>valores[maxPos])
      {
        maxPos=i;
      }
   }
   return maxPos;   //  devuelve la posición del máximo
}
double prom (double valores[], int largo)
{
      double suma=0;
      for (int i=0;i<largo;i++)
      {
         suma+=valores[i];
      }
      return suma/largo;    //  devuelve el promedio del vector
}

int main()
{
   int i, posMax;
   int cantElem=5;
   double vecValores[]= { 6.1, 2.2, 8.0, 4.3, 5.4 };

    posMax=buscarMax(vecValores,cantElem);     //  llama a la función

   cout<<"El maximo es: "<<vecValores[posMax]<<endl;
    cout<<endl<<"El promedio de los valores es: "<< prom(vecValores,cantElem);

   getch();
}
```

```
// Búsqueda del Máximo único y cálculo del Promedio

//  llamar a las dos funciones con dos vectores distintos


int main()
{
   int i, posMax;
   int cantElem1= 5;
   double vecValores1[]= { 6.1, 2.2, 8.0, 4.3, 5.4 };

    posMax= buscarMax(vecValores1,cantElem1);          //  llama a la función para el  primer vector

   cout<<"El maximo del primer vector es: "<<vecValores1[posMax]<<endl;

   cout<<endl<<"El promedio de los valores es: "<< prom(vecValores1,cantElem1);


   int cantElem2= 3;
   double vecValores2[] = { 2.2, 5.5,  3.3  };

    posMax= buscarMax (vecValores2,cantElem2);        //  llama a la función para el segundo vector

   cout<<"El maximo del segundo vector es: "<< vecValores2[posMax]<<endl;

   cout<<endl<<"El promedio de los valores es: "<<  prom(vecValores2,cantElem2);

   getch();
}
```

Función “**cuadrados**” recibe un vector
Función “**cuadrado**” recibe uno a uno los elementos del vector

```
void cuadrados(double vector[], int largo)
{
        for (int i=0;i<largo;i++) {
                vector[i]=cuadrado(vector[i]);
        }
}
...
double cuadrado(double valor) {
        return valor*valor;
}
...
        cuadrados(elementos,num_elem);
...
```

```
// Función Cuadrados que llama a Función Cuadrado

#include<iostream.h>
#include<conio.h>

float cuadrado(float valor){
      float auxResultado;
      auxResultado = valor*valor;
      return(auxResultado);
}

void cuadrados(float vec[], int largo){
     int i;

     for(i=0;i<largo;i++){
        vec[i]=cuadrado(vec[i]);
        }
}

int main(){
    float vec[]={5.2, 3.3, 2.2};
    int i,cant;
    float resultado;
    cant=3;
    cout<<"El primer vector es:"<<endl;
    for(i=0;i<cant;i++){
      cout<<vec[i]<<endl;
     }
    cuadrados(vec,cant);
    cout<<"El vector del doble de cada elemento: "<<endl;
    for(i=0;i<cant;i++){
       cout<<vec[i]<<endl;
       }

    getch();
    return(0);

}
```

Función: **gets**

Efecto de invocarla: Lee una cadena de texto desde teclado.

Biblioteca: stdio.h

Declaración: char ***gets*** (char *cadena);

Parámetro: La variable en la que se quiere guardar la cadena.

Valor devuelto: Devuelve un puntero a la dirección de la cadena leída (en caso de error, devuelve NULL).

Cuidado: Esta función no comprueba si los caracteres leídos, son más que el espacio reservado y puede dar algo erróneo. Por ejemplo si se reservan 20 lugares, solo se pueden guardar 19 caracteres y el '**\0**'.

Cin y Scanf, capturan solamente una cadena de texto **hasta** que aparece el primer espacio o fin de línea (desechando todo lo que venga a continuación).
Por ejemplo si el texto tipeado en pantalla es: "Bienvenida primavera"
Con **Cin**, solo capturará: "Bienvenida", perdiéndose el resto del texto.

En cambio con **gets** se captura **todo el texto**.

Tanto con cin, scanf o gets, se introduce automáticamente un fin de cadena (**\0**) al final del texto capturado.
Por eso es que un array de caracteres siempre contendrá un carácter adicional además del texto visible, el correspondiente al final de cadena o tira de caracteres.

Función: **puts**

Biblioteca: stdio.h

Declaración: int ***puts*** (char *cadena);

Parámetro: De Entrada, la cadena de caracteres a mostrar por pantalla.

// Función que pasa una cadena de caracteres a MAYÚSCULAS - Parámetro vector de caracteres

```
#include <string.h>
#include <conio.h>        // para getch
#include <stdio.h>        //para puts y gets
#include <iostream.h>   // para cint y cout

void  fPasaMayusculas (char cadena[], int largo)
{
int i;
for (i=0;  i<largo;  i++)
cadena[i] = toupper(cadena[i]);
}

int main () {
char palabra[80] = "Programar";
int longitud = strlen(palabra);
cout<<"La palabra original es:          ";
puts(palabra);
cout<<"La longitud de la palabra es: "<<longitud<<endl;
cout<<"La palabra pasada a mayuscula es: ";

fPasaMayusculas (palabra, longitud);

puts(palabra);
getch();
return 0;
}
```

```
#include<iostream.h>
#include<conio.h>

//definición del  struct
struct Datos
{
     int cod;
     float precioCompra;
     };

void  ordenarStruct (Datos orden[], int cant)
{
    int i,j;
    Datos aux;
    for(i=1; i<cant; i++)
    {
      aux=orden[i];       // copia todo el struct en una variable aux (de tipo struct)
      j=i-1;
      while ( (j>=0) && (aux.cod<orden[j].cod) )
      {
         orden[j+1] = orden[j];      // correr el elemento hacia la derecha
         j--;
      }
      orden[j+1]= aux;
    }
}
```

```
int main(){
   int cant,i;
   //  inicialización del  struct  por programa
   Datos productos[5];
   productos[0].cod=100;
   productos[0].precioCompra=1.2;
   productos[1].cod=60;
   productos[1].precioCompra=6.2;
   productos[2].cod=10;
   productos[2].precioCompra=1.6;
   productos[3].cod=70;
   productos[3].precioCompra=4.2;
   productos[4].cod=20;
   productos[4].precioCompra=7.2;

    cant=5;
   //  se invoca a la función ordenarStruct
   ordenarStruct ( productos, cant );
   cout<<  " Impresión de los datos, ordenados por código: "<<endl;
   for(i=0; i<cant; i++ )   {
        cout<< "Codigo: "<<productos[i].cod << "  ";
        cout<<"Precio de compra:  $ "<< productos[i].precioCompra <<endl;
   }
   getch();
   return(0);
}
```

Ejercicio extra: con la estructura del ejercicio anterior.
Realizar la Búsqueda del Máximo y del Mínimo precio de compra.
Versión 1

```
10  //definición del  struct
11  struct Datos
12  {
13      int cod;
14      float precioCompra;
15      };
16
17
18  // función que busca el MÁXIMO y el MÍNIMO precio de compra
19
20  // versión 1: devuelde:
21  // el MÁXIMO a través del nombre de la función y el MÍNIMO como parámetro de uso de salida
22
23  float  busMaxMin1(Datos vec[], int cant, float &min) {
24          float max;
25          int i;
26          min= vec[0].precioCompra;
27          max= vec[0].precioCompra;
28
29          for (i=0; i<cant; i++) {
30            if ( vec[i].precioCompra > max ) {
31                max= vec[i].precioCompra; }
32                else
33                    {
34
35                if ( vec[i].precioCompra< min){
36                  min= vec[i].precioCompra;
37                    }
38              }
39          }
40          return(max);
41  }
42
```

Ejercicio extra: con la estructura del ejercicio anterior.
Realizar la Búsqueda del Máximo y del Mínimo precio de compra.
Versión 2

```
43
44  // función que busca el MÁXIMO y el MÍNIMO precio de compra
45  // versión 2: devuelde el MÁXIMO y el MÍNIMO como parámetros de uso de salida
46
47  void  busMaxMin2 (Datos vec[], int cant, float &min, float &max){
48
49          int i;
50          min= vec[0].precioCompra;
51          max= vec[0].precioCompra;
52
53          for (i=0; i<cant; i++) {
54            if ( vec[i].precioCompra > max ) {
55                max= vec[i].precioCompra; }
56                else
57                    {
58
59                if ( vec[i].precioCompra< min){
60                  min= vec[i].precioCompra;
61                    }
62              }
63          }
64
65  }
```

Función que devuelve un struct

```
#include <iostream>
#include <conio.h>
#include <string.h>
using namespace std;

struct Dato
{
     int a;
     int b;
     int c;
     };

Dato calculo( Dato struct1)
 {
    struct1.a=2;
    struct1.b=2;
    struct1.c=2;
    return  ( struct1 );   // devuelve un struct
    }

int main()
{
  Dato valores, prueba;

   valores.a=3;
   valores.b=3;
   valores.c=3;

   prueba = calculo(valores);

   cout<<"struct copiado: "<<prueba.a<<prueba.b<<prueba.c;
   getch();
   return 0;
}
```

Una varible de tipo struct, es pasada a la función Cálculo, **X COPIA**

La función devuelve una variable tipo struct, a través del nombre de la función.

Igual ejercicio: variable struct pasada por Referencia, a la Función fCalculo

Clase 11 - 8 - Función pasa a Mayúsculas .cpp | Clase 11 - 9 - Función. Vector parámetro - Invertir el vector .cpp | Clase 11 - 10 - Función. Vector de Struct .cpp | Clase 11 - 12 - Función. Variable Struct por REFERENCIA .cpp | Clase 11 - 11 - Función. Struct .cpp

```cpp
// Función que recibe un struct y devuelve un struct
// Por REFERENCIA

#include <iostream>
#include <conio.h>
#include <stdio.h>
#include <iomanip>
using namespace std;


struct Dato
{
      int a;
      int b;
      int c;
      };

Dato fCalculo( Dato &struct1)   // Función que recibe una variable struct  X REFERENCIA
 {
      struct1.a=2;
      struct1.b=2;
      struct1.c=2;
      return  ( struct1 );    // devuelve un struct
      }

int main()
{
  Dato valores, prueba;

    valores.a=3;
    valores.b=3;
    valores.c=3;

    cout<< endl << endl << "Variable valores, struct ORIGINAL: "<< endl << "Campo 1: " << valores.a << endl <<"Campo 2: " << valores.b << endl << "Campo 3: " << valores.c;

    prueba = fCalculo(valores);   //  llama a la función y recibe en variable prueba un struct

    cout<<endl <<endl<< "Variable prueba, struct que devuelve la funcion: "<< endl << "Campo 1: " << prueba.a << endl <<"Campo 2: " << prueba.b << endl << "Campo 3: " << prueba.c;

    // vamos a verificar cómo quedó la variable valores.???
    cout<< endl << endl << "Variable valores, struct Original MODIFICADO por la funcion: "<< endl << "Campo 1: " << valores.a << endl <<"Campo 2: " << valores.b << endl << "Campo 3: " << v

    getch();
    return 0;
}
```

Ejercicio extra: vector como parámetro

Escribir la ***función*** flnvierteVec

- Que reciba un vector de enteros y su longitud
- Que devuelva el mismo vector, ***con sus contenidos invertidos***.

- Si por ejemplo el vector es:

```
int v[] = { 10, 20, 30, 40, 50, 60, 70 };
```

el **vector INVERTIDO** será: 70 60 50 40 30 20 10