Funciones - Parametros x Direccion

Tipo de dato: Puntero

- Un puntero es una variable que contiene la dirección de memoria de otra variable.
- Es considerado un tipo dinámico

Se usa para:

1. Pasar información entre una función y sus puntos de llamada.
2. Armar en memoria estructuras dinámicas de datos (pilas, colas, etc.)

Ejemplo

```
int *punte = NULL;
```

Esto hará que el compilador reserve un lugar en memoria para la variable “punte”, la cual contendrá el valor NULL.

```
int *punte = &aux;
```

El operador de dirección &, permite obtener la dirección de una variable:

Pasaje de parametros

Los parámetros pueden ser pasados a la función de las formas:

1. Por valor
2. Por direccion o referecnia

Por Valor: se pasa a la función una copia del dato. **Por referecia**: sería equivalente a pasar la variable propiamente dicha donde se encuentra el dato

Ejemplo

```
// Función que intercambia 2 variables
// Pasaje de parámetros por REFERENCIA
#include<iostream.h>
#include<conio.h>
#include<stdio.h>

void intercambia(int &a, int &b)
{
    int aux;
    aux= a;
```

```
    a = b;
    b = aux;
}

int main() {
    int x = 10;
    int y = 20;

    cout << "Valores antes de Intercambiar\nx="<<x<<" y="<<y<<endl;
    cout << "x= " << x << " y= " << y << endl ;
    //Llamo a la funcion
    intercambia (x , y) ;
    cout << "x= " << x << " y= " << y ;

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

```
// Función que intercambia 2 variables
// Pasaje de parámetros por DIRECCIÓN
#include<iostream.h>
#include<conio.h>
#include<stdio.h>

void intercambia(int *a, int *b)
{
    int aux;
    aux= *a;
    *a = *b;
    *b = aux;
}

int main() {
    int x = 10;
    int y = 20;

    cout << "x= " << x << " y= " << y << endl ;
    //Llamo a la función
    intercambia (&x , &y) ;
    cout << "x= " << x << " y= " << y ;

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

Uso de parametros

- Entrada
- Salida
- Entada/Salida

Entrada:

Función que dado tres valores , devuelva el mayor de ambos.

```
int mayor ( int n1, int n2, int n3)
```

Salida

- Opc.1: A través del nombre de la función:

```
  int mayor ( int n1, int n2, int n3)
```

- Opc.2: A través de un parámetro de salida:

```
void mayor ( int n1, int n2, int n3, int &max)
//Un 4to. Parámetro de uso de salida estrictamente y función de tipo void
```

Entada/Salida

Función que dado un valor, lo devuelva incrementado en un 50 %.

```
void increm50porc ( float &nro)
```

Funciones que devuelven más de un resultado.

Función que dado 3 valores enteros, devuelve 3 valores el mayor, el menor y la suma.

1.

```
int fMayMenSum ( int n1, int n2, int n3, int &min, int &max )
```

Devuelve:

- 1 valor a través del nombre de la función (por ejemplo la suma)
- Y los otros 2 resultados los devuelve a través de 2 parámetros estrictamente de salida.

2.

```
void fMayMenSum ( int n1, int n2, int n3, int &min, int &max, int &sum )
```

Devuelve

- Nada a través del nombre de la función (void).
- Y los 3 resultados los devuelve a través de 3 parámetros estrictamente de salida.

Parametros actuales

El parámetro ACTUAL de un parámetro FORMAL por valor o por copia, puede ser:

- Una constante
- Una expresión
- Una variable
- Una llamada a otra función del mismo tipo de dato que el parámetro.

Ejemplo

Ejemplo: Para la cabecera de función: Int suma (int a, int b)

En las llamadas, son parámetros actuales VÁLIDOS:

- resu = suma (7, 8); // dos constantes
- resu = suma (nro, 6); // una variable y una constante
- resu = suma (nro+2, nro-3); // expresiones aritméticas del mismo tipo de dato que el parámetro
- resu = suma (suma(n1, n2), suma(n3, n4)); // llamadas a funciones del mismo tipo de dato que el parámetro

El parámetro ACTUAL de un parámetro FORMAL por referencia o por dirección, SIEMPRE DEBE ser:

- Una variable

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

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

Los modificadores más utilizados son:

- %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

El formato completo de los modificadores es el siguiente:

```
% [signo] [longitud] [.precisión] modificador
```

- Signo: indicamos si el valor se ajustará a laizquierda, 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.

```
// FUNCIONES - Pasaje de Parámetros
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

// Función f_uno --> 3 parámetros (x valor, x dirección y x referencia)
void f_uno(char e, char *f, char & d)
{
    *f = e;
    d = '9';
    *f = d;

    cout <<endl<<"*** Funcion 1"<<endl<<endl;
    printf("posicion de e (parametro por valor o copia): %p\n", &e);

    cout<<endl<<"contenido de e: "<< e <<endl<<endl;
    printf("posicion de d (parametro por referencia): %p\n", &d);

    cout<<endl<<"contenido de d: "<< d <<endl<<endl;
    printf("posicion de f (parametro por direccion): %p\n", &f);

    printf("posicion de f (parametro por direccion - a donde apunta)es:
%p\n", f);
    cout<<"contenido de a donde apunta f: "<< *f;

    getch();
}
```

```
// Función f_dos --> 2 parámetros (x valor y x referencia)
void f_dos (char x, char &y)
{
    char w;

    x = 'g';
    w = y;
    y = x;

    cout <<endl<<"*** Funcion 2"<<endl<<endl;
    printf("posicion de y (parametro por referencia): %p\n", &y);

    cout<<endl<<"contenido de y: "<< y<<endl<<endl;
    printf("posicion de x (parametro por valor o copia): %p\n", &x);

    cout<<endl<<"contenido de x: "<< x <<endl<<endl;
```

```
    printf("posicion de w (variable local): %p\n", &w);
    cout<<endl<<"contenido de w: "<< w <<endl<<endl;

    getch();
```

```
int main (void)
{
    char a, b,c;

    a= '3';
    b = 'z';
    c = b;

    cout <<endl<<"main"<<endl;
    printf ("posicion de a: %p\n", &a);
    printf ("posicion de b: %p\n", &b);
    printf ("posicion de c: %p\n", &c);
    cout<<endl<<endl;

    f_dos ( a, b);
    f_uno( '8', &a, c);

    cout<<endl<<c<<" " << b <<" " << a <<" a";

    getch();
}
```