CIMAT

Informática I para Bachillerato

Introducción a la programación en C/C++
Input, Variables, Estructura de Control IF

José Luis Alonzo Velázquez

CIMAT

Sesión 3

El primer ejemplo de programación:

```
1 #include <stdio.h>
2
3 int main(){
4     printf("HOLA MUNDO\n");
5     return 0;
6 }
```

La función **printf**

```
1 int printf( const char * format, ... );
```

INPUT

El hola mundo que hicimos es tan solo un programa que escribe a pantalla. Es decir, produce una salida(output). Sin embargo este no hace nada más. Además no recibe ninguna entrada(input) del usuario. Los típicos programas realmente muestran salidas que dependen de entradas dadas por el usuario del programa.

MAIN es la principal función de un código C

```
1 int main (); // no arguments
2 int main (int argc, char* argv[]); // arguments
```

arcg: cantidad de parámetros contando el nombre del ejecutable que es el primer parámetro.
argv[]: Un arreglo que contiene todos los parámetros recibidos.

Ejemplo

```
1 copy "C:\archivo.ext" "C:\User\archivo.ext"
```

el valor de argc= 3, ya que el nombre del programa es el primer parámetro y luego van los dos parámetros adicionales.
Como puedes apreciar argv

es un arreglo, de modo que, para este caso, *argv*[0]= "copy", *argv*[1]= "C:/User/archivo.ext" y *argv*[2] ==="C:/archivo.ext" (sin las comillas).

Variable

Para poder leer algo, necesitamos un lugar donde poner lo leído, i.e. necesitamos un lugar en la memoria de la maquina donde podamos guardar esta información. A este "lugar" lo llamaremos **objeto**. Un objeto es una región de memoria que tendrá un **tipo** que especifica que clase de información esta siendo colocada en el. Este objeto es llamado **variable**. Será en estas variables donde guardaremos información en nuestros programas.

bool x	x es a Booleano (valor true and false).
char x	x is a character (usually 8 bits).
short x	x is a short int (usually 16 bits).
int x	x is the default integer type.
float x	x is a floating-point number.
double x	x is a double-precision floating-point number.
void *p	p is a pointer to raw memory.
const T x	x is a constant (immutable) version of T.
long T x	x is a long T.
unsigned T x	x is an unsigned T.
signed T x	x is a signed T.

Especificador	Salida	Ejemplo
c	Carácter	a
d or i	Entero	392
e	Notación científica usando e	3.9265e+2
E	Notación científica usando E	3.9265E+2
f	Decimal de punto flotante	392.65
g	El más corto entre %e or %f	392.65
G	El más corto entre %E or %f	392.65
o	Octal con signo	610

Especificador	Salida	Ejemplo
s	Cadena de caracteres	ejemplo
u	Entero decimal sin signo	7235
x	Entero Hexadecimal sin signo	7fa
X	Entero Hexadecimal sin signo	7FA
p	Apuntador a dirección	B800:0000
n	No imprime nada.	

Como hacer comentarios

```
1 // esto comenta una linea.
2 /*
3     esto comenta
4     un bloque de instrucciones
5
6 */
```

variable de tipo **bool**

```
1 #include <stdio.h>
2
3 int main (){
4     bool x=true;
5     printf("El valor de x es: %d",x);
6     return 0;
7 }
```

variable de tipo **short**

```
1 #include <stdio.h>
2
3 int main (){
4     short n;
5     n=32;
6     printf("El valor de n es: %d",n);
7     return 0;
8 }
```

variable de tipo **int**

```
1 #include <stdio.h>
2
3 int main (){
4     int x=5;
5     printf("El valor de x es: %d",x);
6     return 0;
7 }
```

variable de tipo **float**

```
1 #include <stdio.h>
2
3 int main (){
4     float x=3.1416;
5     printf("El valor de x es: %f",x);
6     return 0;
7 }
```

variable de tipo **double**

```
1 #include <stdio.h>
2
3 int main (){
4     double x=3.1416;
5     printf("El valor de x es: %lf",x);
6     return 0;
7 }
```

variable de tipo **char**

```
1 #include <stdio.h>
2
3 int main (){
4     char a;
5     a='r';
6     printf("El valor de x es: %c",a);
7     return 0;
8 }
```

variable de tipo **char***

```
1 #include <stdio.h>
2
3 int main (){
4     char* a;
5     a="Una cadena de caracteres";
6     printf("El valor de x es: %s",a);
7     return 0;
8 }
```

variable de tipo **string**

```
1  #include <stdio.h>
2  #include <string>
3
4
5
6  int main (){
7      string a;
8      a="Una cadena de caracteres";
9      printf("El valor de x es: %s",a);
10     return 0;
11 }
```

Ejemplo

```
1  #include <stdio.h>
2
3  int main(){
4      printf("Caracteres: %c %c \n", 'a', 65);
5      printf("Numeros enteros: %d %ld\n", 1977, 650000L)
       ;
6      printf("Precedidos con espacios blancos: %10d \n",
       1977);
7      printf("Precedidos con ceros: %010d \n", 1977);
8      printf("Flotantes: %4.2f %+.0e %E\n"
       ,3.1416,3.1416,3.1416);
9      printf("%s \n", "Una cadena de caracteres");
10     return 0;
11 }
```

¿Como le hacemos para que el usuario entre datos al programa?

```
int scanf ( const char * format, ... );
```

Ejemplo

```
1  #include <stdio.h>
2
3  int main (){
4    char str[80];
5    int i;
6    printf ("Dame tu nombre: ");
7    scanf ("%s",str);
8    printf ("Dame tu edad: ");
9    scanf ("%d",&i);
10   printf ("Sr. %s su edad es %d.\n",str,i);
11   return 0;
12 }
```

Estructuras de selección

C++ tiene dos estructuras de control para la selección, **if** (selección simple y binaria) y **switch** (selección múltiple).

Sintaxis de la estructura de control **if**

if(⟨Condición⟩){
⟨Instrucción⟩
⟨Instrucción⟩
⋮
⟨Instrucción⟩
}else{
⟨Instrucción⟩
⟨Instrucción⟩
⋮
⟨Instrucción⟩
}

Ejemplo

```
1  #include <stdio.h>
2
3  int main(){
4      int numero;
5      printf("Escribe un numero: ");
6      scanf("%d",&numero);
7      if(numero >= 4){
8          printf("El numero  %d >= 4",numero);
9      }else{
10         printf("El numero  %d < 4",numero);
11     }
12     return 0;
13 }
```

Checar paridad de un número

```
1  #include <stdio.h>
2
3  int main(){
4      int numero;
5      printf("Escribe un numero: ");
6      scanf("%d",&numero);
7      if((numero%2)==0){
8          printf("El numero  %d es par",numero);
9      }else{
10         printf("El numero  %d es impar",numero);
11     }
12     return 0;
13 }
```

Error clásico

if(123 == a) ...
if(a == 123) ...

Error clásico

if(123 == a) ...
if(a == 123) ...

Si nos equivocamos al escribir estas expresiones, y ponemos sólo un signo "=", en el primer caso obtendremos un error del compilador, ya que estaremos intentando cambiar el valor de una constante, lo cual no es posible. En el segundo caso, el valor de la variable cambia, y además el resultado de evaluar la expresión no dependerá de una comparación, sino de una asignación, y siempre será "true", salvo que el valor asignado sea 0.

Funcionamiento de if, ¿qué sucede?

```
1  #include <stdio.h>
2
3  int main(){
4      int numero;
5      printf("Escribe un numero: ");
6      scanf("%d",&numero);
7      if(numero=0){// siempre será "false"
8          printf("El numero  %d es cero",numero);
9      }else{
10         printf("El numero  %d no es cero",numero);
11     }
12     return 0;
13 }
```

Funcionamiento de if, ¿qué sucede?

```
1  #include <stdio.h>
2
3  int main(){
4      int numero;
5      printf("Escribe un numero: ");
6      scanf("%d",&numero);
7  // siempre será "true", ya que 13 es distinto de 0
8      if(numero=13){
9          printf("El numero  %d es cero",numero);
10     }else{
11         printf("El numero  %d no es cero",numero);
12     }
13     return 0;
14 }
```

Un uso más

```
1  if (operador == +)
2      resultado = A + B;
3  else if (operador == -)
4      resultado = A - B;
5  else if (operador == *)
6      resultado = A * B;
7  else if (operador == /)
8      resultado = A / B;
9  else
10     cout << "Operador invalido";
```

Tipos de Errores

Los compiladores clasifican los errores en dos tipos, dependiendo de lo serios que sean:

Tipos de Errores

Los compiladores clasifican los errores en dos tipos, dependiendo de lo serios que sean:

- "**Errores**": son errores que **impiden que el programa pueda ejecutarse**, los programas con "errores" no pueden pasar de la fase de compilación a la de enlazado, que es la fase en que se obtiene el programa ejecutable.

Tipos de Errores

Los compiladores clasifican los errores en dos tipos, dependiendo de lo serios que sean:

- "**Errores**": son errores que **impiden que el programa pueda ejecutarse**, los programas con "errores" no pueden pasar de la fase de compilación a la de enlazado, que es la fase en que se obtiene el programa ejecutable.
- "**Warnings**": son errores de poca entidad, (según el compilador que, por supuesto, no tiene ni idea de lo que intentamos hacer). **Estos errores no impiden** pasar a la fase de enlazado, y por lo tanto es posible ejecutarlos. Debes tener cuidado si tu compilador de da una lista de "warnings", eso significa que has cometido algún error, en cualquier caso repasa esta lista e intenta corregir los "warnings".

 Como Programar en C/C++, Deitel (Prentice Hall), 2da Edición.

 Programming Principles and Practice Using C++, Bjarne Stroustrup.

 http://www.codeblocks.org

 http://www.wxwidgets.org