Informática I

Claudio Paz

claudiojpaz@gmail.com

Agosto 2019

Unidad 7

Funciones en lenguaje C

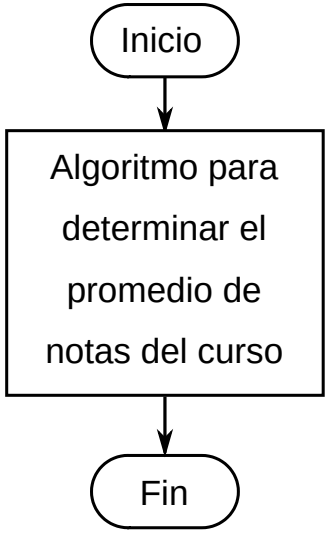
Programación modular

Programación modular

La programación modular consiste en resolver problemas dividiendolos en problemas más pequeños

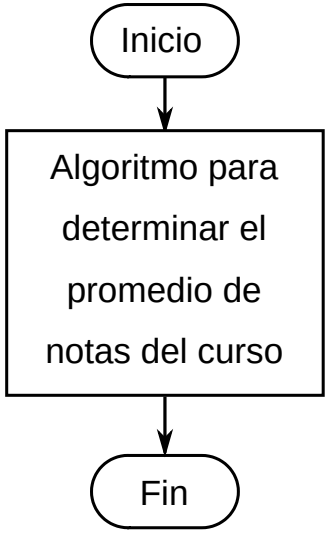
Programación modular

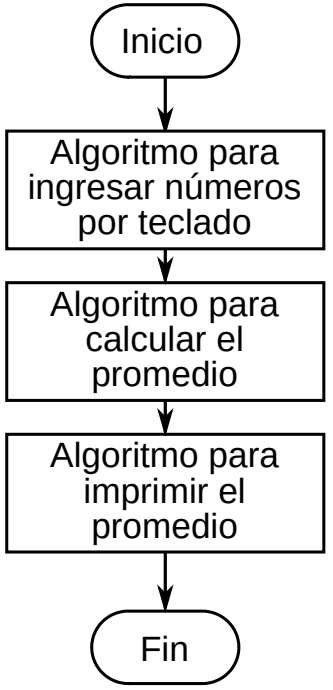
La programación modular consiste en resolver problemas dividiendolos en problemas más pequeños



Programación modular

La programación modular consiste en resolver problemas dividiendolos en problemas más pequeños





Funciones en el lenguage C

Funciones en el lenguage C

Funciones de la biblioteca estándar...

Funciones en el lenguage C

Funciones de la biblioteca estándar...

(de `stdio.h`: `printf`, `scanf`, etc.)

Funciones en el lenguage C

Funciones de la biblioteca estándar...

(de `stdio.h`: `printf`, `scanf`, etc.)

(de `math.h`: `sqrt`, `pow`, `log`, etc.)

Funciones en el lenguage C

Funciones de la biblioteca estándar...

(de `stdio.h`: `printf`, `scanf`, etc.)

(de `math.h`: `sqrt`, `pow`, `log`, etc.)

o nuevas funciones *definidas* por el usuario

Definición de una función

Definición de una función

```
tipo-valor-retorno nombre-funcion (lista-parametros)
{
  sentencias
}
```

Definición de una función

```
tipo-valor-retorno nombre-funcion (lista-parametros)
{
  sentencias
}
```

Definición de una función

```
tipo-valor-retorno nombre-funcion (lista-parametros)
{
  sentencias
}
```

Definición de una función

```
tipo-valor-retorno nombre-funcion (lista-parametros)
{
  sentencias
}
```

Definición de una función

```
tipo-valor-retorno nombre-funcion (lista-parametros)
{
  sentencias
}
```

Definición de una función

```
tipo-valor-retorno nombre-funcion (lista-parametros)
{
  sentencias
}
```

Definición de una función

```
tipo-valor-retorno nombre-funcion (lista-parametros)
{
  sentencias
}
```

Definición de una función

```
tipo-valor-retorno nombre-funcion (lista-parametros)
{
  sentencias
}
```

Definición de una función

Definición de una función

Ejemplo

Definición de una función

Ejemplo

```
float media (int n1, int n2)
{
  int suma;
  float resultado;

  suma = n1 + n2;

  resultado = (float) suma / 2;

  return resultado;
}
```

```c
#include <stdio.h>

float media (int n1, int n2)
{
  int suma;
  float resultado;

  suma = n1 + n2;

  resultado = (float) suma / 2;

  return resultado;
}

int main (void) {
  float m;

  m = media(7, 8);

  printf("La media entre %d y %d es %.1f\n", 7, 8, m);

  return 0;
}
```

```c
#include <stdio.h>

float media (int n1, int n2)
{
  int suma;
  float resultado;

  suma = n1 + n2;

  resultado = (float) suma / 2;

  return resultado;
}

int main (void) {
  float m;

  m = media(7, 8);

  printf("La media entre %d y %d es %.1f\n", 7, 8, m);

  return 0;
}
```

```
La media entre 7 y 8 es 7.5
```

Lista de Parámetros Vs Lista de Argumentos

Lista de Parámetros Vs Lista de Argumentos

Cuando se llama a una función, y se le pasan los valores con los que se quiere que trabaje, como el 7 y 8 del siguiente ejemplo...

```
...
m = media(7, 8);
...
```

se dice que 7 y 8 son la *Lista de Argumentos*

Lista de Parámetros Vs Lista de Argumentos

Cuando se llama a una función, y se le pasan los valores con los que se quiere que trabaje, como el 7 y 8 del siguiente ejemplo...

```
...
m = media(7, 8);
...
```

se dice que 7 y 8 son la *Lista de Argumentos* ...podrían ser variables o cualquier otra expresión

```
...
m = media(a, b);
...
```

Lista de Parámetros Vs Lista de Argumentos

Lista de Parámetros Vs Lista de Argumentos

Cuando se define la función, las variables en donde se *copian* los argumentos se llama *Lista de Parámetros*

```
float media (int n1, int n2)
{
  int suma;
  float resultado;

  suma = n1 + n2;

  resultado = (float) suma / 2;

  return resultado;
}
```

Lista de Parámetros Vs Lista de Argumentos

Lista de Parámetros Vs Lista de Argumentos

La *Lista de Argumentos* debe coincidir con la *Lista de Parámetros* en tipo y número

Lista de Parámetros Vs Lista de Argumentos

La *Lista de Argumentos* debe coincidir con la *Lista de Parámetros* en tipo y número

Las variables de la *Lista de Argumentos* puede tener nombres diferentes de la *Lista de Parámetros*

Uso del valor devuelto por una función

Uso del valor devuelto por una función

```
m = media(7, 8);
printf("La media entre %d y %d es %.1f\n", 7, 8, m);
```

Uso del valor devuelto por una función

```
printf("La media entre %d y %d es %.1f\n", 7, 8, media(7, 8));
```

Uso del valor devuelto por una función

Uso del valor devuelto por una función

```
int valor_absoluto (int numero)
{
  int absoluto;

  if (numero < 0)
    absoluto = -1*numero;
  else
    absoluto = numero;

  return absoluto;
}
```

Uso del valor devuelto por una función

```c
int valor_absoluto (int numero)
{
  int absoluto;

  if (numero < 0)
    absoluto = -1*numero;
  else
    absoluto = numero;

  return absoluto;
}
```

```c
int n;
printf("Ingrese un número para aplicar raiz cuadrada: ");
scanf("%d", &n);

radicando = valor_absoluto(n);
printf("La raiz cuadrada es %.2f\n", sqrt(radicando));
```

(reescribiendo un poco para repasar)

(reescribiendo un poco para repasar)

```
int valor_absoluto (int numero)
{
  int absoluto;

  if (numero < 0)
    absoluto = -1*numero;
  else
    absoluto = numero;

  return absoluto;
}
```

(reescribiendo un poco para repasar)

```c
int valor_absoluto (int numero)
{
  int absoluto;

  if (numero < 0)
    absoluto = -1*numero;
  else
    absoluto = numero;

  return absoluto;
}
```

(reescribiendo un poco para repasar)

```c
int valor_absoluto (int numero)
{
  int absoluto;

  if (numero < 0)
    absoluto = -numero;
  else
    absoluto = numero;

  return absoluto;
}
```

(reescribiendo un poco para repasar)

```
int valor_absoluto (int numero)
{
  int absoluto;

  if (numero < 0)
    absoluto = -numero;
  else
    absoluto = numero;

  return absoluto;
}
```

(reescribiendo un poco para repasar)

```
int valor_absoluto (int numero)
{
  int absoluto;

  if (numero < 0)
    absoluto = -numero;
  else
    absoluto = numero;

  return absoluto;
}
```

(reescribiendo un poco para repasar)

```c
int valor_absoluto (int numero)
{
  int absoluto;

  absoluto = numero;
  if (numero < 0)
    absoluto = -numero;

  return absoluto;
}
```

(reescribiendo un poco para repasar)

```
int valor_absoluto (int numero)
{
  int absoluto;

  absoluto = numero;
  if (numero < 0)
    absoluto = -numero;

  return absoluto;
}
```

(reescribiendo un poco para repasar)

```
int valor_absoluto (int numero)
{
  int absoluto;

  absoluto = numero;
  if (numero < 0)
    absoluto = -numero;

  return absoluto;
}
```

(reescribiendo un poco para repasar)

```
int valor_absoluto (int numero)
{



  if (numero < 0)
    numero = -numero;

  return numero;
}
```

(reescribiendo un poco para repasar)

```
int valor_absoluto (int numero)
{



  if (numero < 0)
    numero = -numero;

  return numero;
}
```

(reescribiendo un poco para repasar)

```
int valor_absoluto (int numero)
{
  int r;

  r = numero < 0 ? -numero : numero;

  return r;
}
```

Operador condicional ternario

Operador condicional ternario

expresion1 ? expresion2 : expresion3

Operador condicional ternario

expresion1 ? expresion2 : expresion3

Si se cumple *expresion1* el operador devuelve *expresion2*, de lo contrario devuelve *expresion3*

Precedencia de Operadores (Actualizada)

Precedencia de Operadores (Actualizada)

Operador	Asociatividad
`()` `[]`	Izq. a Der.
`+` `-` `(tipo)` `++` `--` `!`	Der. a Izq.
`*` `/` `%`	Izq. a Der.
`+` `-`	Izq. a Der.
`<` `<=` `>` `>=`	Izq. a Der.
`==` `!=`	Izq. a Der.
`&&`	Izq. a Der.
`||`	Izq. a Der.
`? :`	Der. a Izq.
`=` `+=` `-=` `/=` `*=` `%=`	Der. a Izq.