Primer Examen Lenguajes de Programacion - Manuel Morales - 03/05/2023

Primera Pregunta

Un Lenguaje de programacion es un conjunto de reglas sintacticas y semanticas que definen su estructura y el significado de sus elementos y expresiones. Un lenguaje de programacion permite a un programador especificar de manera precisa que datos se manipulan, como se manipulan, y que resultados se esperan, existen muchos de estos por la especificidad de cada uno.

Segunda Pregunta

- Primera generacion: Lenguaje de maquina, caracterizado por la utilizacion de codigos binarios.
- Segunda generacion: Lenguaje ensamblador, caracterizado por la utilizacion de mnemonicos. (Ejemplo: `MOV`, `ADD`, `SUB`, etc)
- Tercera generacion: Lenguajes de alto nivel, caracterizado por la utilizacion de palabras y simbolos. (Ejemplo: `C`, `C++`, `Java`, `Python`, etc)
- Cuarta generacion: Lenguajes de programacion de dominio especifico, caracterizado por la utilizacion de lenguajes de alto nivel para resolver problemas especificos. (Ejemplo: `SQL`, `HTML`, `CSS`, etc)
- Quinta generacion: Lenguajes de programacion logicos, caracterizado por la utilizacion de reglas logicas para la resolucion de problemas. (Ejemplo: `Prolog`, `Lisp`, `Haskell`, etc)

Tercera Pregunta

- Abstraccion de datos basicos: Representan los tipos de datos primitivos de cada lenguaje de programacion. (Ejemplo: `int`, `float`, `char`, `bool`, etc)
- Abstraccion de datos estructurales: Representan las estructuras de datos de un lenguaje de programacion. (Ejemplo: `array`, `list`, `stack`, `queue`, etc)
- Abstraccion de datos unitarios: Representan una unidad de logica independiente. (Ejemplo: `clases`, `modulos`, `bibliotecas`, etc)

Cuarta Pregunta

Rango de valores para un tipo integral sin signo.

```
[0, 2^n - 1]
```

Rango de valores para un tipo integral con signo.

```
[-2^(n-1), 2^(n-1) - 1]
```

Quinta Pregunta

Valor minimo, maximo dato integral 8 bits sin signo.

```
[0, 255]
```

Resultado de restar 1 al valor minimos de un dato integral 8 bits sin signo.

```
0 - 1 = 255
```

Representacion en binario:

```
  00000000   <- 0
- 00000001   <- 1
----------
  11111111   <- 255
```

Resultado de sumar 1 al valor maximo de un dato integral 8 bits sin signo.

```
255 + 1 = 0
```

Representacion en binario:

```
  11111111   <- 255
+ 00000001   <- 1
----------
 100000000   <- 256
```

Sexta Pregunta

Valor minimo, maximo dato integral 8 bits con signo

```
[-128, 127]
```

Resultado de restar 1 al valor minimo de un dato integral 8 bits con signo

```
-128 - 1 = 127
```

Representacion en binario:

```
  10000000   <- -128
- 00000001   <- 1
----------
  01111111   <- 127
```

Resultado de sumar 1 al valor maximo de un dato integral 8 bits con signo

```
127 + 1 = -128
```

Representacion en binario:

```
  01111111   <- 127
+ 00000001   <- 1
----------
 10000000   <- -128
```

Septima Pregunta

Value Types en C# que son?

Los value types en C# son tipos de datos que almacenan directamente sus valores en la memoria donde la variable es declarada. Estos son diferentes de los reference types, que almacenan una referencia a la ubicación de memoria del valor real. Los value types son típicamente más eficientes en términos de memoria y rendimiento para tipos de datos simples, debido a que acceden directamente a sus valores sin la indirección de una referencia. Estos son alamacenados en el stack. Ejemplo: `int`, `float`, `char`, `bool`, etc.

- Tipos Conocidos y sus respectivos valores en bits:

Data Type	Bytes	Range
byte	1	0 to 255
sbyte	1	-128 to 127
short	2	-32,768 to 32,767
ushort	2	0 to 65,535
int	4	-2,147,483,648 to 2,147,483,647
uint	4	0 to 4,294,967,295
nint	N/A	Signed 32-or 64-bit integer
nuint	N/A	Unsigned 32-bit or 64-bit integer
long	8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
ulong	8	0 to 18,446,744,073,709,551,615
float	4	Approximately ±3.40282347E+38F
double	8	Approximately ±1.79769313486231570E+308
decimal	16	Approximately ±7.9 x 10^28
char	16	Unicode
bool	8	true or false

Reference Types en C# que son?

Los reference types en C# son tipos de datos que almacenan referencias a la ubicación de memoria del valor real, en lugar del valor en sí. A diferencia de los value types, que se almacenan directamente en la pila, los reference types se almacenan en el heap, y la variable contiene una referencia al dato en el heap. Esto

permite que los reference types puedan ser null, representando la ausencia de un objeto. Los reference types incluyen clases, interfaces, delegados y arrays. Al asignar un reference type de una variable a otra, ambas variables apuntan al mismo objeto en memoria, lo que significa que los cambios realizados a través de una variable se reflejan en la otra. Esta característica facilita la construcción de estructuras de datos complejas y la implementación de comportamientos polimórficos.

Octava Pregunta

a. ¿Qué resultados se muestran en pantalla?

```
byte number = byte.MaxValue;
number++;
sbyte sNumber = sbyte.MaxValue;
sNumber++;
Console.WriteLine(number); // 0
Console.WriteLine(sNumber); // -128
```

- Desarrolle la representación del valor de las variables (number1, number2) en su forma binaria:

```
byte number1 = 207; // 11001111
sbyte number2 = -95; // 10011111
```

- ¿Cuales son los resultados de todas las variables despues de la siguiente ejecución?

```
int number = -1;
bool boolean1 = false & (number++ == 0); // false
bool boolean2 = false && (number++ == 0); // false
bool boolean3 = false || (number++ == 0); // true
bool boolean4 = true | (number++ == 1); // true
bool boolean5 = number == 1; // false
```

- ¿Cuales son los resultados en pantalla de?

```
Console. WriteLine(~13); // -14
Console. WriteLine( 16 | 20); // 20
Console. WriteLine( 5 & 13); // 5
Console. WriteLine( true ^ false); // true
Console. WriteLine( 2 ^ 3); // 1
Console. WriteLine( (0.1 + 0.2) == 0.3); // false
Console.WriteLine((double)(0.1m + 0.2m) == 0.3); //true
```

Novena Pregunta

El estándar IEEE 754 representa números de punto flotante utilizando un bit de signo, un exponente y una mantisa. El bit de signo indica el signo del número (positivo o negativo), el exponente representa la escala del número, y la mantisa contiene los dígitos significativos del número.

Para `float`, el estándar IEEE 754 utiliza 32 bits para representar el número:

Fórmula: `±(1 + M) * 2^(E - 127)`

Donde `M` es la mantisa (23 bits), `E` es el exponente (8 bits), y (1 bit) es el signo.

Para `double`, el estándar IEEE 754 utiliza 64 bits para representar el número:

Fórmula: `±(1 + M) * 2^(E - 1023)` Donde `M` es la mantisa (52 bits), `E` es el exponente (11 bits), y (1 bit) es el signo.