DESARROLLO DE SOFTWARE

INSTITUTO NELSON TORRES

INFORME

TODO SOBRE FUNDAMENTOS DE PROGRAMACION

- **ASIGNATURA** | METODODLOGIA DE LA INVESTIGACION
- **PROFESOR** | ING.SOLIS ACOSTA WALTER SANTIAGO
- **APELLIDOS Y NOMBRE** | QUISHPE NOQUEZ ELVIS RAHI

- **EMAIL** | er.quishpe@intsuperior.edu.ec **TELÉFONO** | 0959933798
- **TRABAJO GRUPAL Nº** | Subtítulo **FECHA DE ENTREGA** | Subtítulo

INSTITUTO SUPERIOR TEGNOLOGICO “NELSON TORRES”

DESARROLLO DE SOFTWARE

NOMBRE; QUISHPE NOQUEZ ELVIS RAHI

ASIGNATURA; METODOLOGIA DE LA INVESTIGACION

CURSO; PRIMERO “B”

TEMA

DESARROLLE UN INFORME DE FUNDAMENTOS DE PROGRAMACION

PERIODO ACADEMICO: OCTUBRE 2023-FEBRERO 2024

Tabla de contenido

Fundamentos de Programación

Los fundamentos de programación son los principios básicos que se utilizan en el desarrollo de los programas informáticos. Los programadores necesitan tener una comprensión sólida de estos fundamentos para poder escribir código para nuevos programas, así como para poder depurar y mantener los ya existentes.

Objetivo principal:

Comprender los fundamentos de programación para poder desarrollar soluciones a problemas computacionales de manera efectiva y eficiente.

Objetivos secundarios:

1. Dominar los conceptos de lógica de programación, incluyendo la descomposición de problemas en pasos más pequeños y la creación de algoritmos claros y concisos.

2. Familiarizarse con las estructuras de control, como los bucles y las estructuras condicionales, para controlar el flujo de ejecución del programa.

3. Entender cómo trabajar con variables y tipos de datos, incluyendo la declaración, asignación y manipulación de datos en la memoria de la computadora.

4. Aprender a utilizar funciones y procedimientos para modularizar el código, promoviendo la reutilización y la organización del mismo.

Introducción

La programación es el arte y la ciencia de escribir instrucciones que una computadora puede seguir para realizar una tarea específica. Los fundamentos de programación son los conceptos básicos que todo programador debe comprender para poder escribir código de manera efectiva y eficiente. (Areatecnologia.com, s.f.)

En esta introducción, exploraremos algunos de los conceptos fundamentales de la programación:
Lógica de programación: La lógica de programación implica la capacidad de resolver problemas de manera estructurada y lógica. Esto implica descomponer un problema en pasos más pequeños y comprensibles que puedan ser traducidos a instrucciones para la computadora.

Algoritmos: Un algoritmo es un conjunto finito de instrucciones precisas y no ambiguas que se deben seguir para resolver un problema o ejecutar una tarea. Los algoritmos son la base de la programación y pueden ser expresados en pseudocódigo o en un lenguaje de programación específico.

Estructuras de control: Las estructuras de control, como los bucles y las estructuras condicionales, permiten controlar el flujo de ejecución del programa. Esto incluye la ejecución repetida de un bloque de código (bucles) y la toma de decisiones basadas en condiciones (estructuras condicionales).

Variables y tipos de datos: Las variables son contenedores para almacenar datos en la memoria de la computadora. Los tipos de datos definen qué tipo de datos puede almacenar una variable, como números enteros, flotantes, cadenas de texto, booleanos, entre otros.

Funciones y procedimientos: Las funciones y procedimientos son bloques de código reutilizables que realizan una tarea específica. Las funciones pueden aceptar parámetros de entrada y devolver un resultado, mientras que los procedimientos pueden realizar una tarea sin necesidad de devolver un valor.

Estructuras de datos: Las estructuras de datos son formas de organizar y almacenar datos en la memoria de la computadora para su posterior manipulación. Algunas estructuras de datos comunes incluyen listas, arrays, conjuntos, diccionarios, pilas y colas.

Desarrollo:

Lógica de Programación

Resolución de problemas:

La resolución de problemas es el proceso de identificar, analizar y resolver problemas de manera estructurada y lógica. En programación, este proceso implica descomponer un problema en pasos más pequeños y comprensibles que puedan ser traducidos a instrucciones para la computadora. (Fundamentos de programación, s.f.) EJEMPLO:

1. Leer num1
2. Leer num2
3. Suma = num1 + num2
4. Imprimir Suma

Algoritmos:

Un algoritmo es un conjunto finito de instrucciones precisas y no ambiguas que se deben seguir para resolver un problema o ejecutar una tarea. Los algoritmos son la base de la programación y pueden ser expresados en pseudocódigo o en un lenguaje de programación específico.

EJEMPLO:

1. Leer número
2. Si el número módulo 2 es igual a 0:
3. Imprimir "El número es par"
4. De lo contrario:
5. Imprimir "El número es impar"

Pseudocódigo:

El pseudocódigo es una forma de expresar algoritmos utilizando un lenguaje informal que se asemeja al lenguaje de programación real. Se utiliza para planificar y diseñar algoritmos antes de implementarlos en un lenguaje de programación específico.

3. Estructuras de Control Estructuras condicionales:

Las estructuras condicionales permiten tomar decisiones basadas en condiciones específicas. Esto incluye el uso de declaraciones if-else para ejecutar diferentes bloques de código dependiendo de si se cumple una condición o no. EJEMPLO

```
x = 10 if x > 5:     print ("x es
mayor que 5") else:
   print ("x no es mayor que 5")
```

Bucles (loops):

Los bucles permiten ejecutar repetidamente un bloque de código mientras se cumpla una condición específica. Esto incluye bucles while, bucles for y en algunos lenguajes, bucles do-while, que ejecutan el bloque de código al menos una vez antes de evaluar la condición.

EJEMPLO

```
for i in range(1, 6):
    print(i)
```

Control de flujo:

El control de flujo se refiere a la dirección en la que se ejecutan las instrucciones en un programa. Las estructuras de control, como las condicionales y los bucles, son fundamentales para controlar el flujo de ejecución del programa.

Variables y Tipos de Datos

Variables:

Las variables son contenedores para almacenar datos en la memoria de la computadora. Se utilizan para almacenar valores que pueden cambiar durante la ejecución del programa. EJEMPLO nombre = "Juan"
edad = 25

Declaración y asignación:

Las variables se declaran utilizando un nombre único y opcionalmente se les asigna un valor inicial. La asignación de valores a variables se realiza utilizando el operador de asignación (=).
EJEMPLO
x = 10

Tipos de datos básicos:

Los tipos de datos básicos incluyen números enteros, números de punto flotante, cadenas de texto y booleanos. Cada tipo de dato tiene características y operaciones específicas asociadas.
EJEMPLO numero
= 10 decimal = 3.14
texto = "Hola"
verdadero = True

Funciones y Procedimientos

Definición y concepto:

Las funciones y procedimientos son bloques de código reutilizables que realizan una tarea específica. Las funciones pueden aceptar parámetros de entrada y devolver un resultado, mientras que los procedimientos pueden realizar una tarea sin necesidad de devolver un valor.
EJEMPLO

```
def suma(a, b):
return a + b

resultado = suma(3, 5)
print(resultado) # Salida: 8
```

Parámetros y argumentos:

Los parámetros son variables que se utilizan para pasar valores a una función o procedimiento. Los argumentos son los valores reales que se pasan a los parámetros cuando se llama a la función o procedimiento.
EJEMPLO def

```
saludo(nombre):
    print("Hola,", nombre)

saludo("Juan") # Salida: Hola, Juan
```

Retorno de valores:

Las funciones pueden devolver un valor como resultado de su ejecución utilizando la palabra clave return. Este valor puede ser utilizado por el código que llama a la función.
EJEMPLO

```
def suma(a, b):
return a + b

resultado = suma(3, 5) print(resultado)  #
Salida: 8
```

Ámbito de las variables:
El ámbito de una variable se refiere a la parte del programa donde esa variable es accesible. Las variables pueden tener ámbitos locales o globales, lo que determina dónde pueden ser utilizadas. EJEMPLO

```
def ejemplo():
x = 10
   print(x)

ejemplo()  # Salida: 10
print(x)   # Esto producirá un error ya que x no está definido fuera de la función
```

Estructuras de Datos
Listas y arrays:
Las listas y arrays son estructuras de datos que permiten almacenar colecciones de elementos en una secuencia ordenada. Los elementos de una lista o array pueden ser accedidos y manipulados individualmente. EJEMPLO

```
lista = [1, 2, 3, 4, 5]
```

Conjuntos (sets):
Los conjuntos son estructuras de datos que almacenan colecciones de elementos únicos. Los conjuntos no permiten elementos duplicados y proporcionan operaciones para realizar operaciones de conjunto como unión, intersección y diferencia. EJEMPLO

```
conjunto = set([1, 2, 3, 4, 5])
```

Pilas (stacks) y colas (queues):
Las pilas y colas son estructuras de datos que siguen el principio de "último en entrar, primero en salir" (LIFO) para pilas y "primero en entrar, primero en salir" (FIFO) para colas. Estas estructuras se utilizan para almacenar y manipular datos de manera ordenada. EJEMPLO

```
# Pila (stack) pila
= []
pila.append(1)
pila.append(2)
pila.pop()  # Devuelve el último elemento agregado, es decir, 2

# Cola (queue) cola =
[] cola.append(1)
cola.append(2)
cola.pop(0)  # Devuelve el primer elemento agregado, es decir, 1
```

Árboles y grafos (concepto básico):
Los árboles y grafos son estructuras de datos no lineales que se utilizan para representar relaciones jerárquicas entre elementos. Los árboles tienen una estructura jerárquica, mientras que los grafos pueden tener estructuras más complejas y no jerárquicas.

7. Modularidad y Reutilización de Código Modularización:
La modularización es el proceso de dividir un programa en módulos más pequeños y manejables que puedan ser desarrollados, probados y mantenidos de manera independiente. EJEMPLO

```
# modulo.py def
funcion():
   print ("Hola desde el módulo")

# main.py
import modulo

modulo.funcion()  # Salida: Hola desde el módulo
```

Importación de módulos (en algunos lenguajes):

En algunos lenguajes de programación, es posible importar módulos o bibliotecas externas para agregar EJEMPLO

```
import math

print(math.pi) # Salida: 3.141592653589793
```

Conclusión

En conclusión, los fundamentos de programación proporcionan los cimientos necesarios para comprender y aplicar eficazmente los conceptos básicos de la escritura de código. Estos fundamentos abarcan desde la lógica de programación hasta las estructuras de datos y el modularidad del código. Al dominar estos fundamentos, los programadores adquieren las habilidades necesarias para resolver problemas de manera estructurada, escribir código claro y legible, y desarrollar programas eficientes y escalables.

La lógica de programación enseña a los programadores a descomponer problemas complejos en pasos más pequeños y manejables, lo que facilita la creación de algoritmos claros y concisos. Las estructuras de control, como las condicionales y los bucles, permiten controlar el flujo de ejecución del programa, mientras que las variables y los tipos de datos proporcionan los medios para almacenar y manipular información.

Referencias

Areatecnologia.com. (s.f.). Obtenido de Areatecnologia.com: https://www.areatecnologia.com/TUTORIALES/FUNDAMENTOS%20DE%20PROGRAMACION.htm

Fundamentos de programación. (s.f.). Obtenido de mheducation.es: https://www.mheducation.es/bcv/guide/capitulo/844814645X.pdf