

Proyecto Final

de

Programación

**Nombre:** Nailé Puerto López

**Nombre de la Facultad :**Matemáticas,Física y Computación

**Nombre de la Universidad:**Universidad Central “Marta Abreu”de las Villas

Introducción:

El software propuesto busca evaluar las pruebas psicométricas que realizan varias personas de las cuales se conoce su nombre, edad y sexo para optar por un empleo. Todas las pruebas que se realizan son de tipo test. Una prueba de tipo test es un conjunto de preguntas a las cuales se debe contestar con una letra (A,B,C,D), de cada test se conoce su número, cuánto valen los aciertos y cuánto se descuenta por los desaciertos, además cuenta con una cadena donde aparecen todas las respuestas correctas del test y una cadena con la respuesta de la persona. Los test se dividen en dos categorías: los simples que se califican sumando les puntos de cada acierto y los puntos de cada desacierto: y los complejos en los cuales dos de las preguntas (cuyo números se conocen) se duplican su valor (tanto si fue un acierto como si fue un desacierto).Así por ejemplo los datos de un test pueden ser:

Test No. 1

Aciertos: 5 puntos

Desaciertos: -1 punto

Respuestas correctas:AABBCDBDDC

Respuesta dada por la persona:AABCEABDAC

A través de la implementación de este sistema en lenguaje C++, se espera agilizar y simplificar el proceso de selección de personal, brindando una solución confiable y eficaz para evaluar las habilidades y competencias de los candidatos mediante pruebas psicométricas.

<< Test>>

-puntos\_acierto:int

-puntos\_desacierto:int

-numero:int

-respuestas\_correctas:string

+operator==(Test &):bool

+ isValido(string): bool

+ int calcularCalificacion(string):int

Ñ

Persona

-nombre:string

-sexo:string

-edad:int

-ci:string

1\*

1\*

TestComplejo

-pregunta1:int

-pregunta2:int

+calcularCalificacion (string):int

TestSimple

+calcularCalificacion (string respuesta):int

1,1 1,1

GestorTest

- tests: vector <Test \*>

- personas: vector <Persona>

- exámenes: vector <pair<int, pair<string, string> > >

+ obtenerPersonas():vector<Persona>

+listarPersonas():void

+addPersona(Persona persona):void

+indexTest(int numero):int

+findTest(int numero):bool

+indexPersona(Persona persona):int

+findPersona(Persona persona):bool

+ findPersona(string ci):bool

+addExamen(int numero, Persona, string respuesta):void

+addTest(Test\* test):void

+indexExamen(int numero, string ci):int

+calificar(int numero, string ci):int

+ notasPersona(Persona):void

+getTests():vector<Test \*>

La **Clase "Persona"** representa a una persona con atributos como nombre, sexo, cédula de identidad (ci) y edad.Los métodos públicos incluyen un constructor con parámetros que asigna los valores de los parámetros a los correspondientes atributos de la clase utilizando el operador de asignación (this->). El operador de igualdad (==) sobrecargado para esta clase compara si la cédula de identidad (ci) de la instancia actual es igual a la cédula de identidad de la persona pasada como parámetro. Retorna true si las cédulas de identidad son iguales, y false en caso contrario.

Bool Persona::operator==(Persona &persona) const {

return this->ci == persona.getCi();

}

La **Clase "Test"** contiene atributos como (numero),un entero que representa el número del test; (puntos\_acierto), un entero que indica la cantidad de puntos asignados por cada respuesta correcta en el test;( puntos\_desacierto), un entero que indica la cantidad de puntos descontados por cada respuesta incorrecta en el test;(respuestas\_correctas), una cadena de caracteres que representa las respuestas correctas del test.En ella se implementa varios métodos como

isValido(string): método estático que verifica si una cadena de caracteres es válida según algún criterio no especificado.

calcularCalificacion(string): método virtual puro que calcula la calificación obtenida en el test a partir de las respuestas dadas.

La **Clase "TestSimple"** hereda de la clase **"Test"** tiene un constructor con parámetros y la implementación del método calcularCalificacion(string respuesta) override ,para esta clase en específico , que calcula la calificación comparando la respuesta dada con las respuestas correctas del test.

int TestSimple::calcularCalificacion(string respuesta) {

int i = 0, puntos = 0;

for (auto u: this->respuestas\_correctas) {

if (i >= respuesta.size()) {

break;

}

if (u == respuesta[i]) {

puntos += puntos\_acierto;

} else {

puntos -= puntos\_desacierto;

}

i++;

}

return puntos;

}

La **Clase "TestComplejo"** hereda de la clase **"Test"** ,tiene atributos como (pregunta1) un entero que representa la pregunta 1 específica del test complejo y (pregunta2) un entero que representa la pregunta 2 específica del test complejo.Se implementa en ella el método calcularCalificacion(string) override que calcula la calificación comparando las respuestas dadas con las respuestas correctas del test, teniendo en cuenta que las preguntas 1 y 2 tienen un valor duplicado en la calificación.

int TestComplejo::calcularCalificacion(string respuesta) {

int i = 0, puntos = 0;

for (auto u: this->respuestas\_correctas) {

if (i >= respuesta.size()) {

break;

}

if (u == respuesta[i]) {

if (i + 1 == pregunta1 || i + 1 == pregunta2) {

puntos += 2 \* puntos\_acierto;

} else {

puntos += puntos\_acierto;

}

} else {

if (i + 1 == pregunta1 || i + 1 == pregunta2) {

puntos -= 2 \* puntos\_acierto;

} else {

puntos -= 2 \* puntos\_desacierto;

}

}

i++;

}

return puntos;

}

La Clase **"GestorTest"**, se encarga de gestionar tests, personas y exámenes.Tiene atributos como (tests): es un vector que almacena punteros a objetos de tipo Test. Representa la colección de tests que gestiona el GestorTest;(personas): es un vector que almacena objetos de tipo Persona. Representa la colección de personas relacionadas con los tests y exámenes gestionados por el GestorTest;(exámenes): es un vector que almacena pares de datos. Cada par contiene un número de test, y otro par que contiene una cadena de caracteres (CI) y una respuesta en forma de cadena. Representa los exámenes realizados por las personas.Contiene métodos como:

int GestorTest::indexPersona(Persona persona): busca la posición en el vector personas donde se encuentra una persona específica.

void GestorTest::addPersona(Persona persona): agrega una nueva persona al vector personas

bool GestorTest::findTest(int numero): verifica si un test con un número específico está presente en el vector tests.

int GestorTest::indexExamen(int numero, string ci): busca la posición en el vector examenes donde se encuentra un examen correspondiente a un número de test y un número de identificación específicos (numero y ci).

void GestorTest::addExamen(int numero, Persona persona, string respuesta): agrega un nuevo examen al vector examenes.

vector<Persona> GestorTest::obtenerPersonas(): devuelve el vector de objetos Persona almacenados en el atributo personas.

bool GestorTest::findPersona(string ci): verifica si una persona con un número de identificación específico está presente en el vector personas.

El flujo general del programa se basa en mostrar un menú en la salida estándar con varias opciones para el usuario.

Conclusiones:

En resumen, el desarrollo de este software para evaluar pruebas psicométricas ha sido una herramienta fundamental en el proceso de selección de personal para un empleo. A partir de los datos proporcionados.El programa ha sido capaz de calificar las pruebas de manera eficiente, considerando los puntos asignados a los aciertos y a los desaciertos. Además, se ha tenido en cuenta la diferencia entre las pruebas simples y complejas. En el caso de las pruebas complejas, se ha duplicado el valor de dos preguntas específicas, tanto para los aciertos como para los desaciertos.Esto ha permitido ahorrar tiempo y recursos, al eliminar la necesidad de evaluar manualmente cada prueba y realizar cálculos complejos. Proporciona resultados precisos y consistentes, lo que brinda a los responsables de la selección una base sólida para tomar decisiones informadas sobre los candidatos y su idoneidad para el empleo en cuestión.