**UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS**

**(Universidad del Perú, Decana de América)**

****

**FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**

**ESCUELA PROFESIONAL DE INGENIERÍA DE SOFTWARE**

**GRUPO 2**

**CURSO:** ALGORITMICA I

**PROFESOR: SALINAS AZAÑA, GILBERTO ANIBAL**

**INTEGRANTES:**

Chavez Farro Luis Andre

Espinola Ravello Annie Katerine

Reyes De la Cruz, Andrés Eduardo

Silva Barra, Ernesto Franco

Tocto Mallqui, Alexis

**LIMA – PERÚ**

**2020**

1. INTRODUCCIÓN

El presente trabajo fue realizado por un equipo de cinco personas, quienes hemos trabajado en conjunto con el fin de poder desarrollar y fortalecer las habilidades de trabajo en equipo que posteriormente nos serán de mucha utilidad en el campo laboral.

1. Problema:

Crear una a aplicación para determinar el archivo de los promedios finales de N alumnos, teniendo como datos de entrada los archivos alumnos.dat, exparcial.dat y exfinal.dat. ¿será viable el desarrollo de esta aplicación?

1. Objetivos:

General.-

* Hacer uso de la programación estructurada para desarrollar un programa que calcule el promedio de alumnos de un determinado curso.

Específicos.-

* Afianzar conocimientos previos.
* Analizar y definir las entidades o tipos de datos.
* Hacer un uso efectivo de la programación modular de acuerdo con las buenas prácticas.
* Implementar cada una de las operaciones básicas a los registros ya inicializados correctamente, así como lograr un correcto funcionamiento de estas para no caer en errores lógicos o de compilación.
* La correcta captura de los datos almacenados en los vectores registros hacia archivos.
* Implementar correctamente los archivos de alumnos y sus respectivas notas finales.
* Mostrar la solución e integración de los programas.

1. Justificación:

El presente trabajo se hace en base a nuestros conocimientos adquiridos a lo largo de las clases semanales, es por este motivo que, con el fin de consolidar nuestros conocimientos y desarrollar una mayor habilidad de programación, ponemos en práctica todo lo aprendido buscando ofrecer un trabajo final correctamente implementado. Al buscar la funcionalidad de nuestra aplicación, esperamos también que pueda ser útil en cualquier ámbito administrativo y ayude así a solucionar problemas de recolección de datos, cálculo de notas, mejorar la rapidez y precisión de la entrega de notas en los fines de ciclo.

1. MARCO TEÓRICO

El presente trabajo fue realizado usando todo lo aprendido en el curso, por lo tanto, haremos una muy breve introducción a los conceptos aplicados.

1. Algoritmo:

* Algoritmo: Es un conjunto de pasos que nos lleva a la solución de un problema.
* Características:
  + Precisión: Los pasos o instrucciones son precisados claramente y sin ambigüedad.
  + Determinismo: Para los mismos datos el algoritmo debe dar los mismos resultados.
  + Finitud: Por más complejo que sea el problema el algoritmo debe ser finito.
  + Relación de entorno: Todo algoritmo requiere de información externa y “compartir ” los resultados y el método.
* Partes : Todo algoritmo consta de tres partes.
  + Datos de entrada: Requiere de información para resolver el problema.
  + Proceso: Procesa los datos para modificar sus estados hasta llegar a los resultados.
  + Salida de datos: La salida es solución del problema.

1. Estructuras de control selectivas:

* De acuerdo a una condición, ejecutar un grupo instrucciones u otra estructura de control (SI-SINO, if then else).
* De acuerdo con el valor de una variable, ejecutar un grupo de instrucciones u otra estructura de control (SIM, switch-case).

1. Estructuras de control repetitivas:

* Ejecutar un grupo de instrucciones un número determinado de veces (PARA, for).
* Ejecutar un grupo de instrucciones mientras se cumpla una condición(MIENTRAS, while).
* Ejecutar un grupo de instrucciones donde al menos se repite una si se cumpla una condición(HACER-MIENTRAS, do-while).

1. Apuntadores:

* Un apuntador es una variable que almacena la dirección de una variable.
* Cuando declaramos variables de algún tipo, lo que estamos haciendo es separar espacio de memoria para almacenar valores de esas variables.

Por ejemplo: int a=5, b=10, c;

Los valares pueden accesarse a través del identificador de la variable.

* Un apuntador es un “tipo especial” de variable que almacena la dirección de memoria de ese identificador de variable en lugar de valor de la variable.

1. Programación modular:

* Simplificación del problema y de los subprogramas resultantes de cada descomposición.
* Las diferentes partes del problema pueden ser programadas de modo independiente e incluso por diferentes personas.
* El programa principal queda estructurado en forma de bloques o módulos lo que hace sencilla su mantenimiento y su lectura.
* El subprograma recibe valores del subprograma que lo referencia, hace los cálculos y los devuelve o retorna los resultados al subprograma que lo llamó.

1. Arreglos:

Es una estructura estática, contigua, homogénea, ordenada y finita de elementos.

* Estática: Tiene un número máximo de elementos y debe ser declarado antes de definirse el arreglo.
* Contigua: Elementos uno a continuación de otro elemento.
* Homogénea: Todos sus elementos son del mismo tipo.
* Ordenada: Ocupan un determinado lugar en la estructura y podemos citar como primer, segundo , …, enésimo.
* Finita: Pues tiene un numero finito de elementos.

Existe varios tipos de arreglos.

* Unidimensionales: Denominados también vectores, se accede a través de un nombre y un subíndice.
* Bidimensionales: Denominados también matrices, se accede a través de su nombre y dos subíndices, el primero denominado fila y el segundo columna.
* Multidimensionales: Se refiere a arreglos de más de dos dimensiones por ejemplo de subíndices i, j, k. Em realidad es un arreglo con múltiples vectores.

1. Registros:

Son tipos de datos estructurados, así como las Uniones y tipos Enumerados .

* Tipos abstractos: Tipos de datos abstracto (TAD) o tablas como los vectores, matrices y de múltiples dimensiones. Son conjuntos de datos u objetos a los cuales se asocian operaciones.
* TAD. Tipos abstractos de datos es un modelo que define valores y las operaciones que se pueden realizar sobre ellos.
  + Es una práctica que permite crear un tipo de dato abstracto (TAD) independiente de estructura y de lenguaje.
* También se denomina: tipo de datos creados por el usuario , pues permite crear tipos de datos de la realidad.
  + Alumno, Producto, TipoComplejo , TipoRacional , Persona, etc.

Siendo un nuevo tipo de dato tiene las mismas prerrogativas que un tipo de dato predefinido del lenguaje:

* Vector de registros: Se puede crear vectores de registros.
* Registros con vectores: Pue contener como campos del registro tipos datos estructurados como vectores.
* Registros anidados: Puede contener con campos del registro tipos de dato de tipo registro.

1. Archivos:

Un archivo es un conjunto de datos estructurados en una colección de entidades elementales o básicas denominadas registros que son de igual tipo y constan a su vez de diferentes entidades de nivel más bajos denominadas campos.

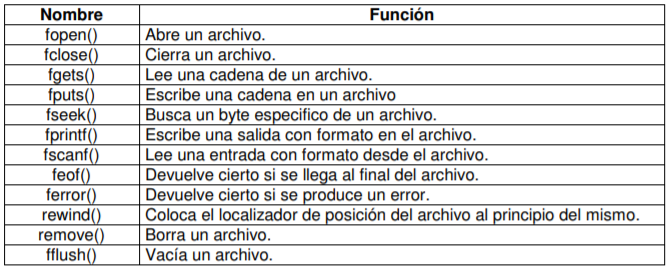
Hay dos tipos de archivos, archivos de texto y archivos binarios.

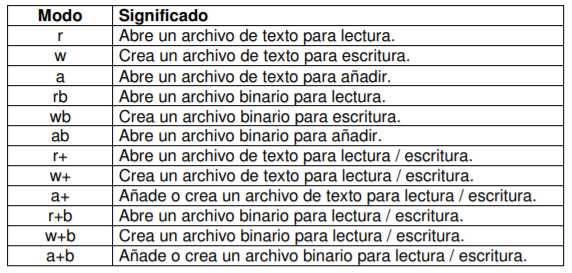
Un archivo de texto: es una secuencia de caracteres organizadas en líneas terminadas por un carácter de nueva línea. En estos archivos se pueden almacenar canciones, fuentes de programas, base de datos simples, etc. Los archivos de texto se caracterizan por ser planos, es decir, todas las letras tienen el mismo formato y no hay palabras subrayadas, en negrita, o letras de distinto tamaño o ancho.

Un archivo binario: es una secuencia de bytes que tienen una correspondencia uno a uno con un dispositivo externo. Así que no tendrá lugar ninguna traducción de caracteres. Además, el número de bytes escritos (leídos) será el mismo que los encontrados en el dispositivo externo. Ejemplos de estos archivos son Fotografías, imágenes, texto con formatos, archivos ejecutables (aplicaciones), etc.

El lenguaje C ofrece un amplio conjunto de funciones de librerías para crear y utilizar archivos de datos. Lo primero que debemos hacer con los archivos de datos orientados a flujo es crear un área de buffer o de almacenamiento. Un área de buffer es un lugar de memoria que se usa para transferir datos desde la memoria a los dispositivos de almacenamiento secundario y viceversa.

Se puede conseguir la entrada y la salida de datos a un archivo a través del uso de la biblioteca de funciones; C no tiene palabras claves que realicen las operaciones de E/S. La siguiente tabla da un breve resumen de las funciones que se pueden utilizar. Se debe incluir la librería STDIO.H. Observe que la mayoría de las funciones comienzan con la letra ""F', esto es un vestigio del estándar C de Unix."





1. ESTADO DEL ARTE

La programación modular resuelve el problema de la programación “espaguetti”, además permite el trabajo en conjunto, para así lograr mejores soluciones a determinados problemas.

Entre los lenguajes de programación que admiten el concepto de programación modular se encuentran C, Ada, PL/I, Erlang, Pascal, Algol, COBOL, RPG, Haskell, Python, HyperTalk, IBM/360 Assembler, MATLAB, Ruby, IBM RPG, SmallTalk, Morpho, Java (los paquetes se consideran módulos), Perl, etc.

Vemos entonces que la programación modular es usada en casi todos(si no todos) los lenguajes de programación que se usan en la actualidad, sus ventajas saltan a relucir al permitir una programación independiente ya que varios programadores pueden trabajar en el mismo proyecto diseñando distintos módulos que luego formarán parte de un programa más grande y complejo.

En la actualidad se usa la programación modular para lograr productos finales de calidad y de gran complejidad, ya que este nos permite un desarrollo más eficiente y con mayor rapidez.

Este modo de trabajar en el código también permite una depuración y modificación más fáciles, ya que, si se encuentra un error o se desea mejorar algo, solo debes ir directo al módulo o módulos que intervienen en el problema.

1. APORTE TEÓRICO

Utilizando la metodología de la programación modular, logramos un mejor avance del proyecto, trabajando en equipo (remotamente) gracias a ello.

Algunos de los pasos a realizar son:

* Tener conocimientos previos de los temas llevados en clase.
* Dividir las estructuras a trabajar por cada integrante.
* Avanzar por separado la implementación de las operaciones básicas a cada registro.
* Comparar el avance final en un determinado día.
* Corregir errores, buscar en conjunto la optimización de la mejor solución.
* Implementar el mismo modelo a todos los registros.
* Trabajar todo el código usando la metodología de la programación modular.
* Realizar una correcta implementación de los archivos para calcular la nota final de los alumnos.

1. APORTE PRÁCTICO

Es este segmento detallaremos los pasos realizados en la solución y/o realización del proyecto final, el cual consiste en implementar una aplicación para calcular el promedio final de un alumno utilizando archivos y todos los conocimientos adquiridos a lo largo del curso.

1. Procesos

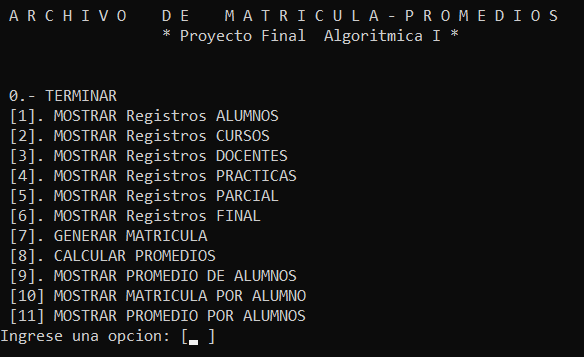
* En primer lugar, tuvo lugar la distribución de tareas equitativas, para que así cada integrante del proyecto aportara en igual medida a los demás.
* Se tuvo que afianzar los conocimientos previos, para ello se brindó ayuda a quienes aún no terminaban de comprender un determinado tema, de esta forma garantizamos un mejor avance en conjunto.
* En la implementación de operaciones básicas en los registros, se tuvo que trabajar por separado, luego de ello que procedió a comparar soluciones y elegimos a la que a nuestro criterio sería una mejor solución.
* Luego de la depuración vimos conveniente optimizar esta mejor solución e implementarla a las demás estructuras, para así presentar un primer entregable más uniforme.
* En la etapa de salvar y recuperar seguimos el mismo proceso hasta elegir la mejor solución (a nuestro criterio) y procedimos a implementarla a las demás estructuras.
* Luego de estos procesos vimos conveniente definir la cantidad de alumnos, docentes y cursos a trabajar, para que así los datos de prueba vayan acorde a una simulación de situación real con el manejo de notas de una institución.
* Procedimos a crear los archivos de notas finales, notas parciales y prácticas, todas estas necesarias para realizar el cálculo de la nota final.
* Se procederá a unir cada estructura y archivo para lograr un programa integral con todo lo pedido para realizar el cálculo.

1. Funcionalidad

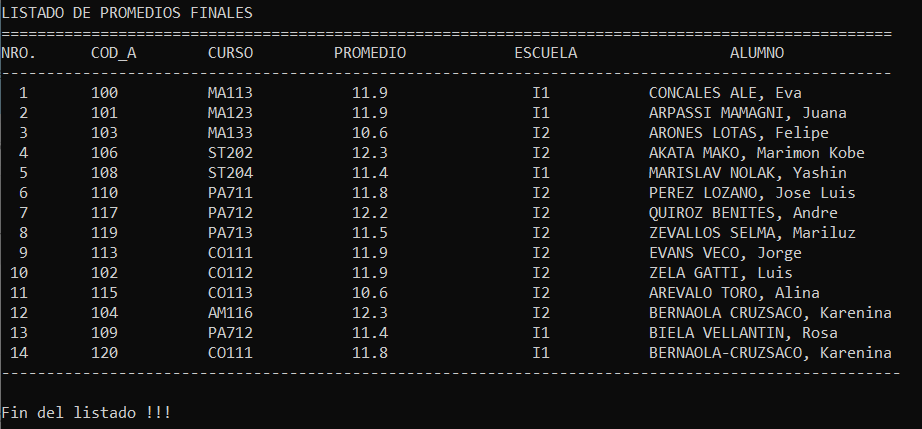
* Nuestro programa resuelve el problema de las notas de alumnos dispersas buscando evitar la pérdida de información como también optimizando la recolección y manejo de estas.
* El programa comienza inicializando las estructuras para cada archivo, ya sea docentes, alumnos, exámenes parciales, etc.
* Seguidamente se pasa a implementar las operaciones básicas aplicando el concepto de programación modular en cada una de ellas.
* Se procede a hacer uso del subprograma Adivinar para aumentar la cantidad de elementos de la estructura a 14.
* Se podrá elegir guardar el archivo, para que así se pueda acceder al archivo en cualquier momento que sea necesario, todo esto haciendo uso del subprograma Archivar.
* Se generarán archivos para cada estructura.
* Luego se realizar la segunda versión de la estructura en las que las operaciones básicas se realizaran solamente con archivos.
* Finalmente, se implementa el cálculo, para poder determinar los promedios de cada alumno individualmente.

1. Vistas de la solución

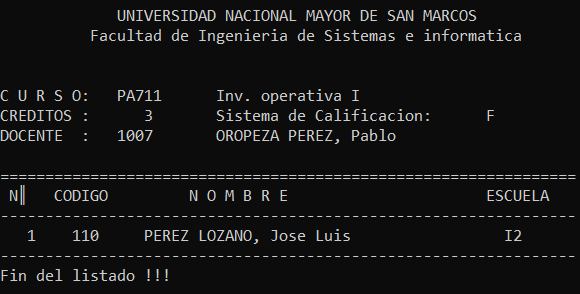
Menú de operaciones



Promedio de Alumnos

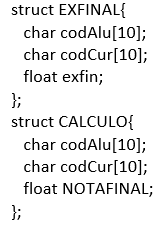
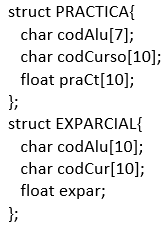


Matrícula por alumno



1. Código solución más importante

Estructuras utilizadas:



Pseudocódigo del cálculo:

**ACCION CALCULO** (FILE I, FILE J, FILE K,FILE L, PRACTICA RegistroP, EXPARCIAL RegistroPAR, EXFINAL RegistroF, CALCULO RegistroCAL)

REAL cont🡨0, promedio🡨0

ABRIR(I);

ABRIR(J);

ABRIR(K);

ABRIR(L);

**MIENTRAS** (NOULTIMO(I))

**PARA** i 🡨 inicio **HASTA** 10

cont=cont+RegistroP.praCt[j];

i 🡨 i + 1

**FIN\_PARA**

cont 🡨cont/10;

promedio = (cont+RegistroPAR.expar+RegistroF.exfin)/3

RegistroCAL.codAlu 🡨 RegistroP.codAlu

RegistroCAL.codCur 🡨 RegistroP.codCurso

RegistroCAL.NOTAFINAL=promedio

cont=0

PONER(L,RegistroCAL )

TOMAR(I, RegistroP)

TOMAR(J, RegistroPAR)

TOMAR(K, RegistroF)

**FIN\_MIENTRAS**

MARCAR(I)

MARCAR(J)

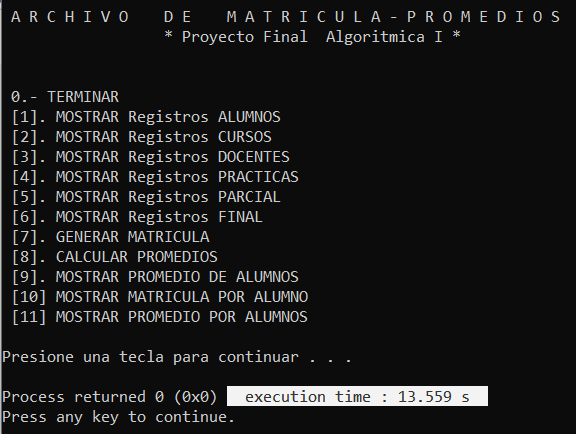
MARCAR(K)

MARCAR(L)

**FIN\_ACCIÓN**

1. EVALUACIÓN

La evaluación de un algoritmo (Serie de pasos bien definidos que ayudan a llegar a la solución de algún problema) tiene como propósito medir su desempeño, considerando el tiempo de ejecución y los recursos empleados (memoria de la computadora), para obtener una solución satisfactoria.  
  
En muchas ocasiones se le da mayor peso al tiempo que tarda un algoritmo en resolver un problema.  
  
Para medir el tiempo de ejecución, el algoritmo se puede transformar a un programa de computadora.



Como se puede visualizar en cada uno de los procesos que se realiza, se muestra el tiempo de ejecución de nuestro algoritmo, pudiendo así evaluar si cumple o no con los estándares requeridos para nuestro equipo de trabajo.

1. CONCLUSIÓN

* Primero Denotamos que mediante la elaboración de nuestro proyecto logramos afianzar todos los conocimientos previos expuestos en clase, algunos en mayor o menor medida, pero siempre tratando de apoyarnos como grupo.
* Otro punto importante Fuimos aplicando la metodología de la programación modular y las buenas prácticas hasta lograr un programa que consideramos cumple con los estándares del curso.
* \*Logramos también la implementación de las operaciones básicas para cada registro, las cuales no solo son aplicables a este proyecto, si no que sirven de base para elaborar una infinidad de programas.
* Y claro que también: Fortalecimos nuestro pensamiento lógico, haciendo así más sencillo la elaboración del proyecto, lo cual nos servirá para el desarrollo de diversos programas a futuro.
* Pudimos también capturar de manera eficaz los datos almacenados en los vectores de registros hacia archivos con la implementación de las funciones salvar y recuperar, para que los datos sean perdurables incluso después de cerrar y volver a abrir el programa. Todo esto visto en el segundo entregable.
* El mayor logro de nuestro equipo fue el de integrar todos los archivos generados previamente por los programas en su versión 1, dentro del programa calculo en el cual generamos los promedios entre los exámenes y practicas e incluso un archivo de matrícula donde podremos visualizar las notas, el curso y profesor asignado a cada alumno.
* Por último, podemos concluir que el equipo aprendió mucho en el proceso de elaboración del proyecto, logramos la correcta integración de los programas y mostramos una solución que esperamos sea del agrado tanto del profesor como de nuestros compañeros.

1. BIBLIOGRAFÍA

JOYANES, Luis: *Fundamentos de programación. Algoritmos y Estructura de Datos*.

4ª edición, Madrid, McGraw-Hill, 2008.

JOYANES, Luis, RODRÍGUEZ, Luis, y FERNÁNDEZ, Matilde: *Fundamentos de*

*programación. Libro de problemas*. McGraw-Hill, 1996.

JOYANES, Luis, y ZAHONERO, Ignacio: *Programación en C. Metodología, algoritmos y estructura de datos*, McGraw-Hill, 1998.

TED JENSEN. Apuntadores y arreglos en C, 1ª Edición. México, 2000.