Información General

Nombre del Módulo:	Lenguajes de Programación I
Autor(es) :	Paola Elvira Ortega Jurado
Año, versión:	2013
Créditos académicos:	4
Competencia Global:	Desarrollar programas de software aplicando los conocimientos generales de la programación, para solucionar situaciones problémicas emergentes en contextos reales

Justificación

Sin duda alguna los lenguajes de programación son uno de los principales pilares para el correcto desarrollo de aplicativos que son utilizados en la vida diaria, es así como encontramos sistemas de información para facturación, médicos, para búsqueda de documentos, bibliográficos, para construcción, etc. Como podemos ver en la cotidianidad encontramos a los lenguajes de programación inmersos en los recursos que a diario utilizamos siendo esto transparente al usuario de tal manera que al realizar por ejemplo una transferencia bancaria veamos los resultados tangibles sin conocer la manera como el sistema de información lo realizó.

Basados en lo anterior el módulo de Lenguajes de Programación I, nos brindará las bases principales para desarrollar aplicaciones elementales de situaciones problema planteadas en las cuales lograremos evidenciar la importancia de la programación en la cotidianidad.

En el siguiente enlace encontramos una descripción bastante clara de que son los lenguajes de programación http://www.youtube.com/watch?v=SQtvPaQGVQg y las diferencias con lenguajes de máquina y lenguaje ensamblador.

Metodología de aprendizaje del módulo

Para desarrollar la competencia propuesta en el módulo, se han planteado elementos de competencia, cada uno con criterios de desempeño específicos; para todos los criterios de desempeño se han preparado actividades de aprendizaje y recursos que facilitan al estudiante el desarrollo y la presentación de las evidencias que serán valoradas por el facilitador para determinar si se alcanza o no la competencia.

Es importante tener presente la interacción como elemento fundamental en la metodología de estudio virtual; mediante los comentarios, aportes y preguntas se fortalecen los conocimientos y competencias; todos los integrantes del grupo están en capacidad de aportar.

El estudiante es el centro del proceso y su disciplina y autoestudio se constituyen en factores fundamentales para esta modalidad educativa. Adicionalmente las herramientas de la plataforma ofrecen espacios para la presentación de los contenidos, la interacción, la entrega de evidencias y verificación de competencias alcanzadas.

Evaluación

La evaluación por competencias en la Católica del Norte se orienta por los siguientes principios:

1. Continua: no hay momentos específicos para la evaluación, se realiza durante todo el periodo y a través de las actividades programadas en los módulos.

2. Integral: se valoran el ser, el saber y el hacer; es decir, el ser humano en todas sus dimensiones.

3. Refuerzo permanente: está implícito en el desarrollo de las actividades, porque el mismo criterio de desempeño se puede valorar en diferentes momentos. Los refuerzos culminan a la par de la terminación del bloque.

   La evaluación de las competencias se realiza a partir de la entrega de las evidencias, éstas a su vez están planteadas como actividades y cada una cuenta con una serie de criterios que estandarizan la valoración de los resultados.

   Los criterios de desempeño definidos para cada elemento de la competencia, son la base para determinar los resultados de aprendizaje que se estructuran con base en EVIDENCIAS DE APRENDIZAJE que son las pruebas manifiestas de aprendizaje, recogidas directamente durante el proceso formativo. Son recolectadas con la orientación del docente o facilitador, utilizando técnicas, métodos e instrumentos de evaluación seleccionados, según sean evidencias de conocimiento, de producto o de desempeño, permitiendo reconocer los logros obtenidos por el estudiante en los tres tipos de saberes: conceptual, procedimental y actitudinal.

   EVIDENCIAS DE CONOCIMIENTO. Apuntan al dominio cognoscitivo para procesar e identificar información relevante, su clasificación, su interpretación de manera útil, y la búsqueda de las relaciones entre información nueva e información adquirida previamente. Incluye el conocimiento de hechos y procesos, la comprensión de los principios, y teorías y las maneras de utilizar el conocimiento en situaciones cotidianas y nuevas.

   EVIDENCIAS DE DESEMPEÑO. Evidencias del saber procedimental, relativas al cómo ejecuta el estudiante una actividad, en donde pone en juego sus habilidades, conocimientos y actitudes. Permiten recoger información directa, de mejor calidad y más confiable, sobre la forma como el estudiante desarrolla su proceso de aprendizaje y así poder identificar cuáles han sido sus logros y cuáles le faltan por alcanzar. Incluye las evidencias actitudinales.

EVIDENCIAS DE PRODUCTO. Son los resultados que obtiene el estudiante en una actividad que refleja el aprendizaje alcanzado y permite hacer inferencias sobre el proceso desarrollado, o método utilizado.

El Glosario

Arreglos (Array): Es una secuencia de objetos del mismo tipo que se numeran de manera consecutiva. Los elementos del Array son:

- Valores de índice o subíndice: que es la manera como se numeran.
- Elemento: Así se denomina a cada ítem del Array.

Bucles: Son estructuras de control iterativas o repetitivas, que realizan la iteración de acciones o sea controlan el número de veces que una sentencia o lista de sentencias se deben ejecutar. C++ tiene tres tipos de estrcutruas, for, while y do-hile.

Búsquedas: Es una operación que se lleva a cabo dentro de los arreglos. Se trata de encontrar un elemento dentro de un array teniendo en cuenta un valor definido.

Comentarios: Es una información añadida al código, pero que no se tendrá en cuenta en la compilación del mismo. Cuando se desea realizar un comentario de mas de una línea se lo hace por medio de los caracteres especiales /* y */. Si se trata de una sola línea con los caracteres //.

Constantes: Es exactamente el opuesto a variable las cuales corresponden exactamente a aquellos objetos-datos que pueden recibir nuevas asignaciones de valor a lo largo del programa, en otras palabras son entidades cuyo valor solo es conocido en tiempo de ejecución.

De esta manera una cosntante es una entidad cuyo valor no se puede cambiar y utiliza la

palabra reservada const, de esta manera se declaran las constantes. En C++ una constante se debe inicializar cuando se declara.

Do-while: Se utiliza cuando se desea que las instrucciones dentro del bucle se ejecuten por lo menos una sola vez, esto se logra al tener la condición después de las instrucciones. Esta es la diferencia con el bucle while
Sintaxis:

```
do
    sentencia
while (expresión)
```

Estructura: Tipo de dato definido por el usuario y que se declara antes de que se pueda utilizar. Es la colección de uno o más tipos de elementos a los cuales se los denomina miembros, cada uno tiene un nombre único definido como nombre del miembro.

Estructuras de control repetitivas: Son aquellas estrucutras qeu realizan varias veces una acción determinada, C++ tiene en cuenta tres tipos de estrucutras de este tipo, estos son los bucles: while, for y do-while. estas controlan el número de veces que se repetinen las instrucciones.

Eliminación: Es una operación que se lleva a cabo dentro de los arreglos. Se trata de Borrar un elemento dentro de un array teniendo en cuenta un valor determinado.

For: Se utiliza por lo general cuando deseamos estructuras de control por medio de contadores, este tipo de bucle ejecuta un conjunto de sentencia una vez por cada valor d eun rango especificado, en otras palabras pro cada valor de una variable contador de un rango específico: ejecutar sentencias.

Sintaxis:

for (inicialización; condición_iteracion; incremento)
sentencias,

Funciones: Uno de los elementos más importentes dentro de todo lenguaje de programación. En terminos de autores destacados dentro de la programación tenemos: " todo programa no era más que la suma de código (rutinas, procedimientos o funciones, como se les quiera llamar) y datos (variables, matrices, etc.). sea como sea, las funciones tienen un papel estelar en el desarrollo de aplicaciones en la actualidad ". Teniendo en cuenta lo anterior se puede decir que las funciones son trozos de código que tienen un nombre y que se utilizan en cualquier parte de nuestro código con una llamada directa.

Identificadores: Son conjuntos de letras y/o números que se utilizan para simbolizar todos los elementos que en un programa son definibles por el usuario del mismo llamese programador ó ingeniero de software. Se definen también como elementos también llamados símbolos que nombran entidades del lenguaje. Como ejemplo se pueden citar: las variables, constantes, tipos de datos, etiquetas, subrutinas. Se debe tener en cuenta que si un identificador corresponde a una palabra clave o reservada, ya no puede utilizarse para referirse a otro tipo de entidades como varialbes o constantes.

Inserción: Es una operación que se lleva a cabo dentro de los arreglos. Se trata de Insertar un elemento dentro de un array teniendo en cuenta que no exista duplicidad de datos.

Operadores: Se utilizan junto con las expresiones para resolver operaciones. Se tiene operadores aritméticos, lógicos y relacionales, de mnipulación de bits, condicionales y especiales.

Operador Aritmético: Se utilizan para realizar cálculos de aritmética de números reales y de punteros. Se tienen los siguientes:

- + Con dos posibilidades: Suma binaria y más unitario
- ++ Incremento unitario
- - Dos posibilidades: Resta binaria y menos unitario
- -- Decremento unitario
- Multiplicación. Se debe tener cuidado con este símbolo que tiene también otros usos
- / División
- % Resto ó módulo

Operador Lógico: Producen un resultado booleano y sus operadores son de igual manera valores lógicos . Se tienen tres tipos, dos de los cuales son binarios y el de negación que es unario. Son los siguientes:

- Y lógico (AND) &&
- lógico (OR) ||
- Negación Lógica (NOT) !

Operador Relacional: Son denominados también operadores lógicos y de comparación, se usan para comprobar la falsedad ó veracidad de unas relaciones propuestas. En otras palabras es como las respuestas de F ó V ante determnadas preguntas, en otras palasbras el resultado de estos operadores siempre será booleano. Se tienen los siguientes:

- > Mayor que
- < Menor que
- >= Mayor igual que

- <= Menor igual que
- == Igual que
- != Diferente que

Operador de manipulación de bits: Esta es una gran ventaja que presenta C++ sobre otros lenguajes de programación y son métodos para acceder a los bits individuales dentro de un byte, lo cual es de gran utilidad teniendo en cuenta que:

Se pueden almacenar hasta ocho variables lógicas en un solo byte, de esta manera ahorramos espacio

Algunos dispositivos codifican la información a nivel de bits, de tal manera que los programas que se comuniquen con dichos dispositivos deben ser capaces de manejar bits

Algunas rutinas de cifrado y descifrado de información utilizan operaciones a nivel de bits.

Las operaciones a nivel de bits, directamente heredadas del lenguaje ensamblador, permiten a C comprobar, asignar o desplazar los bits de un byte con toda libertad. Entre los operadores tenemos:

- & : Operación AND a nivel de bits
- I : Operación OR a nivel de bits
- ^ : Operación XOR a nivel de bits (OR exclusivo)
- ~ : Complemento a uno
- >> : Desplazamiento a la derecha
- << : Desplazamiento a la izquierda

Programa (Informático): Es un conjunto de instrucciones que una vez ejecutadas realizarán una o varias tareas en una computadora. Por los sistemas Unix se los conoce como binarios, teniendo en cuenta que los ficheros no necesitan el uso de extensiones. En los sistemas windows son conocidos por su extensión .exe que los convierte en ficheros ejecutables. Los programas pasan por diferentes fases, primero son escritos en un lenguaje de programación que es reconocido por los seres humanos de tal manera que es más fácil escribirlos, luego se traduce a un único lenguaje de máquina que está conformado por ceros y unos a esto se denomina código de máquina. Así tenemos que en un programa ejecutable el código fuente se transforma en un binario cuando es compilado.

Recorridos: Es la acción que se lleva a cabo en array para encontrar un determinado valor dentro del mismo.

Reglas de asociatividad: Son la manera como se evalúa una expresión con la misma prioridad. La interpretación de cualquier expresión en C++ se da por la prioridad y asociatividad de los operadores en la expresión. Cada operador tienen una prioridad y los operadores en la expresión se evalúan en orden de mayor a menor prioridad, como lo vimos en el anterior concepto. La evaluación de operadores con la misma prioridad viene determinado por su asociatividad la cual puede ser de derecha a izquierda o viceversa.

Reglas de prioridad: Es importante aclarar que las reglas de prioridad cambian si se tienen paréntesis dentro de las expresiones. Las reglas de prioridad son utilizadas para evaluar la manera como el codificador tendrá en cuenta los operadores en una expresión. Si en una expresión no existen paréntesis y se tiene primero un signo mas + y luego un signo por *, lo primero que se calcula es la multiplicación y luego la suma, teniendo en cuenta que la multiplicación tiene mayor prioridad que el operando +.

Sentencias de control: Controlan la secuencia o flujo de ejecución de las sentencias. Se dividen en tres grandes categorías: secuencia, selección y repetición.

Sentencia de control if: Permite una acción previamente determinada por el programador, la cual se cumplirá si la condición tiene valor lógico verdadero ó desarrollará otra acción si el valor lógico falso.

Su sintaxis es la siguiente:

```
If (expresión)
sentencia_1;
else /* opcional */

sentencia_2;
```

Sentencia if-else anidada: La sentencia if se anida cuando la condición verdadera o la condición falsa, es a su vez una sentencia if. De esta manera este tipo de sentencias se utilizan para implementar múltiples alternativas.

Su sintaxis es:

```
if (condicion-1)
   sentencia_1;
else if (condicion_2)
    sentencia_2;
     .....;
```

```
else if (condición_n)
    sentencia_n;
else
     sentencia;
```

Switch: Se utiliza para realizar una selección entre varias posibles.

Sintaxis:

```
switch(selector)
{
        case etiqueta1 : sentecias_1;
        case etiqueta2 : sentecias_2;
        ......;
        case etiquetan : sentencias_n;
        default : sentencias; // esta última instrucción es opcional
}
```

Sentencia while: Es una estructura de control repetitiva que tiene la posición de la condición delante del cuerpo del blucle o sea antes de las instrucciones, lo cual significa que es un bucle pretest, de esta manera se evalúa la condición antes de que se ejecuten las instrucciones

Sintaxis:

```
while (condición_bucle)
     sentencia;
```

Ó cuando se tienen varias sentencias tenemos:

```
while (condicion_bucle))
{
    sentencia_1;
    sentencia_2;
    sentencia_3;
    ......
    sentencia_n;
}
```

Tipos de Datos: Toda Variable definida dentro de un programa debe ser de un tipo específico de dato. Estos tipos de datos definen el posible rango de valores que una variable puede tomar al momento de ejecución del programa y a lo largo de la vida útil del mismo. En C++ se tienen varios tipos, de los cuales los más comunes son.

unsigned char de 8 bits de 0 a 255

char de 8 bits de -128 a 127

short int 16 bits -32.768 a 32,767

unsigned int 32 bits de 0 a 4.294.967.295

int 32 bits de -2.147.483.648 a 2--147.483.647

unsigned long 32 bits de 0 a 4.294.967.295

enum 16 bits -2,147,483,648 a 2,147,483,647

long 32 bits -2,147,483,648 a 2,147,483,647

float 32 bits 3.4 x 10-38 a 3.4 x 10+38(6 dec)

double 64 bits 1.7 x 10-308 a 1.7*10+308(15 dec)

long double 80 bits 3.4 x 10-4932 a 1.1 x 10+4932

void sin valor

Variables: Se define como un identificador que se utiliza para almacenar todos los datos generados durante la ejecución de un programa. Existen unas normas para definirlas:

* No deben tener espacios en blanco, ni símbolos extraños en ellas

* Se utilizan abreviaturas de carácter general

* No deben ser palabras reservadas del lenguaje

Estructuración general del módulo

Elemento de competencia 1: Generalidades			
SEMANAS	TEMAS	HORAS SEMANALES AD	HORAS SEMANALES TI
1.	Conceptos fundamentales de programación y Manejo del Entorno de desarrollo integrado de Code::Blocks	4	12
2	Forma General de un Programa en C++	5	15
3	Comentarios, variables, constantes y operadores	7	21
Elemento de competencia 2: Estructuras de Control			

SEMANA	TEMAS	HORAS SEMANALES AD	HORAS SEMANALES TI
4	Sentencias de control selectivas if, if-else, switch	6	18
5	Bucles for, while, do-while	6	18
6	Estructuras de control para el manejo de errores y excepciones	4	12
Elemento de competencia 3: Arreglos y Matrices			
SEMANA	TEMAS	HORAS SEMANALES AD	HORAS SEMANALES TI
7	Arreglos: Vectores y Matrices	8	24
8	Operaciones en arreglos: búsquedas, recorridos inserción y eliminación de elementos	8	24
TOTAL		48	144

PLANTEAMIENTO DEL PROBLEMA

Caso: Juego del Ahorcado.

El juego del ahorcado se juega con dos personas (o una persona y el computador). Un jugador selecciona una palabra y el otro trata de adivinarla seleccionando letras individuales. Diseñar un programa para jugar al ahorcado.

Sugerencia: Almacenar una lista de palabras un un array y seleccionar palabras aleatoriamente.