Computación y Algoritmos

MAT-151

Dr. Alonso Ramirez Manzanares
CIMAT A.C.

e-mail: alram@cimat.mx
web: www.cimat.mx/~alram/comp_algo/

Objetivos del curso

- Entender los conceptos de algoritmo y estructura de datos.
- Entender la importancia del buen diseño de algoritmos.
- Al final del curso que sean capaces de resolver problemas e implementarlos en una computadora utilizando un lenguaje de alto nivel. (C++)
- Que sepan entender y analizar el desempeño de un algoritmo y determinar las estructuras de datos adecuadas para su implementación.
- Motivarlos que el area de análisis de algoritmos es muy interesante tanto para matemáticos como para computólogos (hay una intersección enorme)

Pre-requisitos

- Conocimientos de programación en C, C++. (A lo largo del curso, veremos lo que necesitamos de C++, ¡*pregunten todas las dudas!*)

Clases

- Las clases son los martes y jueves de 11:00 a 12:20

- Laboratorio los viernes de 11:00 a 12:20 (con ayudante@cimat.mx y YY yy@cimat.mx)

- Las notas de las clases al igual que las tareas se podrán bajar en el sitio web: http://www.cimat.mx/~alram/comp_algo/

Políticas de evaluación

- Calificaciones
    - Exámenes (2 o 3) - 35%
    - Proyecto (en 2 entregas) - 30%
    - Tareas - 30%, (se entregan “a la semana siguiente”, es decir 23:59 hrs. del día antes de la clase, ¡ tarea no entregada no vale !, pero **SI HAY** calificaciones parciales por trabajo parcial)
    - Exámenes rápidos – 5%

Temario

- Introducción al análisis y diseño de algoritmos.
- Conceptos básicos para analizar algoritmos: notación asintótica.
- Estructuras de datos básicas: tipos de datos, arreglos, cadenas, listas ligadas, pilas, colas.
- Recursión, recursión con memoria.
- Algoritmos fundamentales: búsqueda y ordenamiento.
- Algoritmos de gráfos y árboles.
- Estrategias de implementación y diseño de algoritmos: fuerza bruta, algoritmos glotones, dividir para vencer, programación dinámica.

Estructuras de Datos y Algoritmos, Temario detallado

1. Introducción al Análisis de Algoritmos
    1. Complejidad de los problemas y de los algoritmos.
        1. Introducción a la Notación Asintótica.
        2. Clases de crecimiento asintótico y propiedades.
        3. Clases de complejidad de los algoritmos.
    2. Ejemplos (Quick-Find, Quick-union, Búsqueda secuencial y binaria, …)
2. Estructuras de datos básicas y estructuras de datos abstractos
3. C++ Constructores y Listas
4. Sobrecarga de operadores, listas
5. Introducción a recursión, FIFO, LIFO y ecuaciones de recurrencia
6. Recursividad con memoria en el lenguaje C++
7. La bolsa del ladrón. Evaluación de expresiones infijas mediante evaluación de expresiones postfijas.
8. Templates
9. Árboles
10. Algoritmos de Ordenamiento
11. Colas de prioridad, heaps y heap sort
12. BST y rotaciones
13. Árboles AVL
14. Tablas de Hash
15. La Standard Template Library (STL)
16. Programación Dinámica
17. Algoritmos Greedy
18. Introducción a la Teoría de Grafos
    1. Definiciones.
    2. Recorridos: DFS, BFS.
    3. Ejemplo: Topological Sort

Referencias

- R. Sedgewick. *Algorithms in C++*. Addison Wesley. (puesto en reserva)

- B.Preiss. *Data Structures and Algorithms with Object Oriented Design Patterns in (C++, Java).* http://www.brpreiss.com/

- C.Cormen, C.Leiserson, R.Rivest y C.Stein. *Introduction to Algorithms.* MIT Press. (puesto en reserva)

- J.Kleinberg y E.Tardos. *Algorithm Design*. Addison Wesley.

- D.Knuth. *The Art of Computer Programming. Vol.1 Fundamental Algorithms, Vol.3 Sorting and Searching.* Addison-Wesley.

Tutoría

- Tienen horas de practica con ayudantes (xx@cimat.mx y yyy@cimat.mx) viernes 11 - 12:30 hrs. Laboratorio de cómputo.

- Previa cita, ya sea por e-mail (alram@cimat.mx) o teléfono (4494), mi cubículo es el K217 (antes H1) de CIMAT.

Cosas que hay que saber de programación en C

- Ciclos-iteradores al 100%
- Apuntadores
- Memoria dinámica
- Leer escribir archivos (texto y binarios)
- Pasar argumentos a los programas
- Estructuras
- etc...

Modo de programación

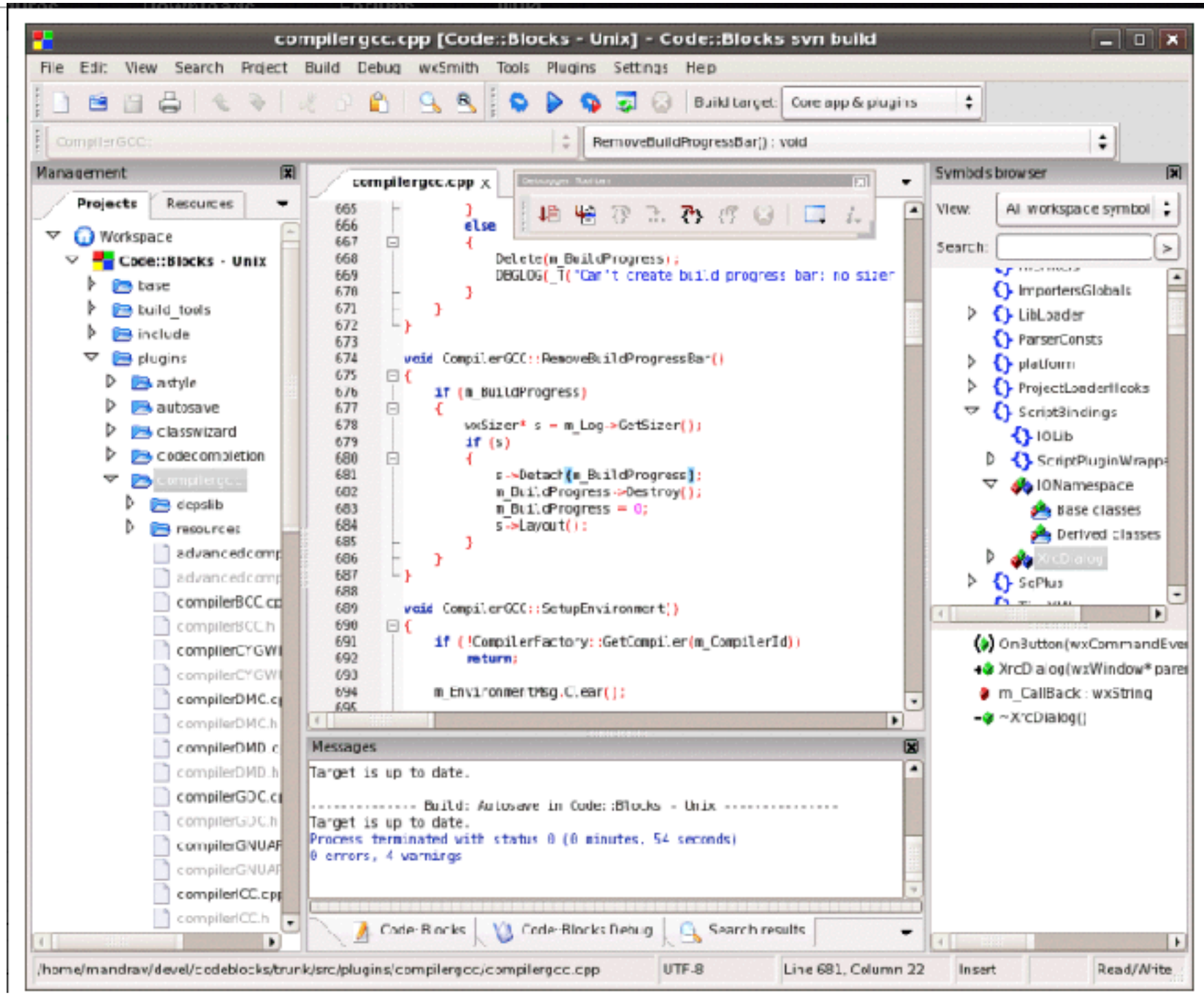
- Usaremos C++ con el compilador linux gcc o Windows MinGW (http://www.mingw.org/).

- El ambiente de desarrollo será Code::Blocks. (http://www.codeblocks.org/ Instalación muy fácil en linux-ubuntu y windows). En Windows hay que bajar el que tiene MinGW integrado llamado IDE (Integrated development environment). Usaremos el tipo de proyecto “Console Application” con lenguaje C++.

- ¿Por qué usar gcc y MinGW?
  - Compatibilidad multiplataforma bajo normas ANSI.

- ¿Cuales son las bondades de un ambiente integrado de programación?
  - Tener un editor y un debugger integrado (Es MUY importante aprender a usar el DEBUGGER)

Screenshot de code::blocks



Tarea:

- mandar por correo una descripción breve del uso de las siguientes opciones de debugger en code::blocks:

  - ***continue, next line, next instruction, step into, step out, toggle break point, remove all break points, run to cursor, debugging windows*** y ***edit watches***.

Revisión de conceptos de C

- 1.- ¿Qué imprime en pantalla el siguiente programa?

```
int x,i;
for(i=10;i>5;i-=3)
 printf(“%d”,i);
```

Revisión de conceptos de C

- 2.- ¿Cuál es la diferencia entre las variables?

```
unsigned char a;
signed char b;
```

Revisión de conceptos de C

- 3.- Dado que el tamaño en bytes de un tipo `int` es 4, ¿cuál es la salida a pantalla?

```
int x=0;
char *y;
y = (char *)(&x);
y[1] = 255;
printf("%d\n",x);
```

- 4.- Indica la diferencia entre pasar un argumento por valor y uno por referencia. ¿Es uno más eficiente que otro?, si es así, ¿en qué sentido?

Revisión de conceptos de C

- 5.- Explica cual es el uso de la variable tipo **static**, por ejemplo en la función;

```
int valor(int param){
  static int b = 5;
  b = b + param;
  return b;
}
```

- 6.- Escribe un programa que use ciclos para generar las permutaciones de las letras 'A' 'B' y 'C' y 'D', por ejemplo, para solo 3 letras la salida debe mostrar ABC ACB BAC BCA CAB CBA

Revisión de conceptos de C

- 7.- **Evaluación de un polinomio en un punto.** Supón que los coeficientes de un polinomio de grado n (**`int n;`**) están guardados en un vector de tamaño *n+1* (**`double coeficientes[n+1]`**), y programa el código que escriba en pantalla la evaluación del polinomio en ***M*** puntos *x's* usando una función, los puntos están guardados en un vector de tamaño ***M*** (**`double xs[M]`**).

- Por ejemplo, para un polinomio de grado 3: $p(x)=3x^3+x-1$ (con coeficientes -1, 1, 0, 3) evaluado en los ***M***=2 puntos: x=0 y x=2 sería:

```
p(0) = -1
p(2) = 25
```

- La función que evalúa el polinomio debe de servir para cualquier *n*.

- Ayuda: La función `pow(x,y)`, definida en math.h, realiza la operación x^y y su definición es `double pow (double x, double y)`