UCAB UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Algoritmos y Estructuras de Datos

Oscar Meza

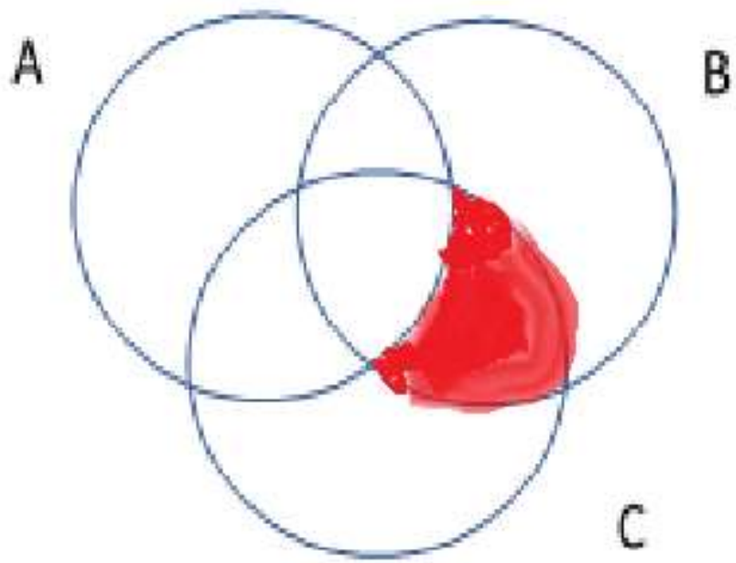
omezahou@ucab.edu.ve

- Crear una carpeta vacía llamada Conjunto (**NO PUEDE HABER OTRO .CPP DE OTROS PROGRAMAS, SOLO LOS DE CONJUNTO)**.
- Bajarse a esta carpeta, los archivos Conjunto_ref.h, Conjunto_ref.cpp de Modulo 7 (laboratorio 4)
- Ejecutar VSCODE y abrir esa carpeta (open folder)
- Ejecutar con DEBUG prueba_conjunto.cpp (DARA ERROR) pero crea carpeta vscode y dentro el archivo tasks.json....
- **cambiar el tasks.json : "${file}", por "${workspaceFolder}*.cpp",** si quiere ejecutar el programa con VScode. **Si no utilice el comando en terminal:**
  - **g++ -o prueba_conjunto prueba_conjunto.cpp Conjunto.cpp**

Agregue a la implementación dada del TDA conjunto de enteros (modificando los archivos .h y .cpp de Conjunto) :

- la operación de union de dos conjuntos y devolver un conjunto nuevo con la unión,
- la operación de diferencia de dos conjuntos y devolver un conjunto nuevo con la diferencia.
- Una función que devuelva los elementos de un conjunto en un vector, para poder tener acceso a los elementos del conjunto
- Y luego un programa de prueba para probar el tipo Conjunto.

Agregar a su programa de prueba que lea tres conjuntos A, B y C de enteros y determine e imprima los elementos que están en la zona roja de la figura:



```
Conjunto unir(const Conjunto& A, const Conjunto& B){
    Conjunto result = A;
    for (auto i : B)
        if (!pertenece(A , i)) result.push_back(i);
    return result;
}

Conjunto diferencia(const Conjunto& A, const Conjunto& B){
    Conjunto result;
    for (auto i : A)
        if (! pertenece(B , i)) result.push_back(i);
    return result;
}
vector<int> elementos(Conjunto A){
    return A;
}
```

Ejercicio 2:

Bajarse de modulo 7 (laboratorio 4) lista_simple.h y lista_simple.cpp.

Examinar el código. Un nodo contiene una sección de clases y mas adelante haremos la asignación horaria

Recuerden:

- Crear una carpeta vacía llamada Lista
- En VSCODE abrir esa carpeta
- Colocar los archivos (Lista_simple.h, Lista_simple.cpp) que estan en modulo 7 laboratorio 4
- -Crear un archivp donde haran la prueba de su funcion
- cambiar el tasks.json : "${file}", por "${workspaceFolder}*.cpp",

Agregar una función que devuelva el dato que está en el nodo en la posición i de la lista, deberán validar que i es una posición válida. Hacer programa de prueba que imprima el elemento en posición i.

Ejercicio 3:

Implementemos asignacion de horarios (una version simplificada) utilizando los tipos Conjunto y Lista_simple anteriores. Las funciones de asignación de horarios formarán parte de Lista_simple

Crear una carpeta llamada asignacion_horarios y copiar conjunto_ref.h, conjunto_ref.cpp, lista_simple.h y lista_simple.cpp

(Explicar...)

```
void hacer_asignacion(Lista_simple& lista){
    Lista_simple recorre_secciones=lista, anterior_seccion;
    vector<int> horas(10,0);
    while (recorre_secciones!=nullptr){
        anterior_seccion=lista;
        while (anterior_seccion!=recorre_secciones) {
            // Si hay conflicto marcar hora de anterior_seccion en horas[]
            if (anterior_seccion->datos.profesor==recorre_secciones->
                                                        datos.profesor
               || !esvacio(intersectar(anterior_seccion->datos.estudiantes,
                                       recorre_secciones->datos.estudiantes)))
                if (anterior_seccion->hora!=0) // si = 0,  posiblemente no fue
                                               // asignada una hora
                    // recordar que las horas van desde 1 por eso -1
                    horas[anterior_seccion->hora-1]=anterior_seccion->hora;
            anterior_seccion=anterior_seccion->proximo;
        }
```

```
        int i=0;
        while(i<horas.size()){
            if (horas[i]==0) {
                    // escoger el primero distinto de cero
                    // la hora a asignar a recorre_secciones
                    // sera (i+1), (i+1) es la menor hora no asignada

                    recorre_secciones->hora = i+1;
                    break;
                }
            i++;
        }
        // aqui se deberia guardar la seccion asignada en el archivo
        // note que si i=horas.size() entonces no se pudo asignar hora a la seccion
        // igual se deberia guardar la seccion indicando que no hay horas asignadas
        recorre_secciones=recorre_secciones->proximo;
        horas.assign(horas.size(),0); // se inicializa horas todos en 0
    }
}
```

Ejercicio 4:

Hacer un programa que cree en el main un arreglo dinámico (con NEW) de elementos tipo string de tamaño dado por el usuario N, luego llame a una función llamada Introducir_Datos, que pida al usuario N palabras por teclado y los guarde en el arreglo. Luego se llama a una función para imprimir el arreglo.

El prototipo de la funcion sería:

void Introducir_Datos(string s[ ], int n) // n es el tamaño del arreglo

Como declarar arreglo dinamico: string *p = new string[tam];