**Checklist de Revisión del Código**

|  |  |  |  |
| --- | --- | --- | --- |
| Estudiante | José Guillermo Saldaña Cárdenas | Fecha | 8 mar. 20 |
| Programa | ITC11 | # Programa | 3 |
| Instructor | Adriana González | Lenguaje | C++ |

|  |  |
| --- | --- |
| Propósito | Guiarte en la conducción de una revisión efectiva de tu diseño |
| General | * Escribe en el primer renglón el nombre de las partes que vas a revisar (aquellas que son nuevas o que van a ser modificadas) * Revisa tu diseño parte por parte. No continúes con la siguiente parte hasta que no hayas revisado por completo la parte anterior. * Cuando estés revisando una parte, revisa una categoría a la vez * Cuando termines de revisar una categoría para una parte, márcala como revisada |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre de las partes: | | Main | Main | Main |  |
| Congruencia | Validar que las clases escritas tengan relación directa al diseño, y que todos los nombres de métodos y variables estén de acuerdo con el diseño estipulado. | ✓ | ✓ | ✓ |  |
| Sintaxis | Verificar que las variables no tengan errores de tecleo o que estén mal escritas y confundidas por otras variables dentro del código. | ✓ | ✓ | ✓ |  |
| Outputs | Revisar que los mensajes de aceptación y de excepción estén de acuerdo con los planteados en el Diseño de Pruebas. | ✓ | ✓ | ✓ |  |
| Dependencias | Verificar que cada archivo de código fuente tenga solamente las dependencias que necesita para funcionar por su cuenta, ni más ni menos. | ✓ | ✓ | ✓ |  |
| Condicionales | Validar que haya casos de entrada y de salto para cada condicional. También validar la complejidad de la condicional y evaluar si hay condicionales equivalentes más sencillas de comprender. |  | ✓ | ✓ |  |
| Uso eficiente de variables y memoria | Verificar que cada variable creada es realmente necesaria para el funcionamiento del programa. Reutilizar variables donde es posible para no alocar memoria innecesaria. También revisar los inicializadores y valores iniciales de cada variable para verificar que sean los adecuados. |  |  | ✓ |  |
| Estándar de codificación | Asegurarme de cumplir con todas las reglas y obligaciones que se estipulan en el estándar de código |  |  | ✓ |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre de las partes: | | Lector |  |  |  |
| Congruencia | Validar que las clases escritas tengan relación directa al diseño, y que todos los nombres de métodos y variables estén de acuerdo con el diseño estipulado. | ✓ |  |  |  |
| Sintaxis | Verificar que las variables no tengan errores de tecleo o que estén mal escritas y confundidas por otras variables dentro del código. | ✓ |  |  |  |
| Outputs | Revisar que los mensajes de aceptación y de excepción estén de acuerdo con los planteados en el Diseño de Pruebas. | ✓ |  |  |  |
| Dependencias | Verificar que cada archivo de código fuente tenga solamente las dependencias que necesita para funcionar por su cuenta, ni más ni menos. | ✓ |  |  |  |
| Condicionales | Validar que haya casos de entrada y de salto para cada condicional. También validar la complejidad de la condicional y evaluar si hay condicionales equivalentes más sencillas de comprender. | ✓ |  |  |  |
| Uso eficiente de variables y memoria | Verificar que cada variable creada es realmente necesaria para el funcionamiento del programa. Reutilizar variables donde es posible para no alocar memoria innecesaria. También revisar los inicializadores y valores iniciales de cada variable para verificar que sean los adecuados. | ✓ |  |  |  |
| Estándar de codificación | Asegurarme de cumplir con todas las reglas y obligaciones que se estipulan en el estándar de código | ✓ |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre de las partes: | | Analizador |  |  |  |
| Congruencia | Validar que las clases escritas tengan relación directa al diseño, y que todos los nombres de métodos y variables estén de acuerdo con el diseño estipulado. | ✓ |  |  |  |
| Sintaxis | Verificar que las variables no tengan errores de tecleo o que estén mal escritas y confundidas por otras variables dentro del código. | ✓ |  |  |  |
| Outputs | Revisar que los mensajes de aceptación y de excepción estén de acuerdo con los planteados en el Diseño de Pruebas. | ✓ |  |  |  |
| Dependencias | Verificar que cada archivo de código fuente tenga solamente las dependencias que necesita para funcionar por su cuenta, ni más ni menos. | ✓ |  |  |  |
| Condicionales | Validar que haya casos de entrada y de salto para cada condicional. También validar la complejidad de la condicional y evaluar si hay condicionales equivalentes más sencillas de comprender. | ✓ |  |  |  |
| Uso eficiente de variables y memoria | Verificar que cada variable creada es realmente necesaria para el funcionamiento del programa. Reutilizar variables donde es posible para no alocar memoria innecesaria. También revisar los inicializadores y valores iniciales de cada variable para verificar que sean los adecuados. | ✓ |  |  |  |
| Estándar de codificación | Asegurarme de cumplir con todas las reglas y obligaciones que se estipulan en el estándar de código | ✓ |  |  |  |