Métodos de Programação – Turma: A
Andre Garrido Damaceno – 15/0117531

TRABALHO 3:
REVISÃO DO CÓDIGO DO JOGO DE 20 PERGUNTAS.

COMO FAZER O *DESIGN* DO SOFTWARE

1. Passou.

   Item: Faça um modelo conceitual do software, com suas necessidades, capacidades e limitações.

2. Passou.

   Item; Faça uma interface que comunique o modelo conceitual ao usuário.

3. Não passou. Não foi feito esse item antes do início do desenvolvimento, porém esse diagrama está disponível na documentação do programa.

   Item: Faça um diagrama de atividades para cada função criada.

4. Não passou. Não foi feito esse item antes do início do desenvolvimento, porém esse diagrama está disponível na documentação do programa.

   Item: Faça um diagrama de estados com as funções e interligações dos módulos.

5. Passou.

   Item: Evitar substituir funções por macros, já que podem causar problemas se mal planejadas.

6. Passou.

   Item: Defina bem as ligações das estruturas de dados do programa.

7. Passou.

   Item: Definir a melhor estrutura de armazenamento dinâmico a ser usado.

8. Passou.

   Item: Caso não esteja especificado, definir a melhor estrutura de armazenamento estático a ser usado.

9. Passou.

   Item: Verifique se existem bibliotecas open-source ou feitas por você que possam facilitar no design do software.

10. Passou.

    Item: Deve ser padronizada a estrutura do código em áreas.

11. Não passou. Foi corrigido esse problema passando a função “Resposta” do módulo “Vinte_Perguntas.c” para o módulo “Funcs.c”

Item: Mantenha o design consistente de cada módulo.

12. Passou.

    Item: Deve ser feito um refinamento dos módulos de acordo com o design.

13. Passou.

    Item: Deve ser sempre salva a referência inicial da memória alocada antes de manipulá-la.

14. Passou.

    Item: Desenvolva o código sempre de forma legível e autoexplicativo.

15. Passou.

    Item: Áreas pouco intuitivas devem ser comentadas.

16. Passou.

    Item: Desenvolva a 'Main' do programa de forma simples, com apenas a chamada de funções, sem ter muita lógica do programa envolvida.

17. Passou.

    Item: Caso as condições 'if' feitas sejam curtas, considere o uso do operador ternário.

18. Passou.

    Item: Faça uma interface de definições dos módulos feitos.

19. Não passou. Foi corrigido esse problema ao ajustar a função "Vinte_Perguntas" com um switch case, removendo os vários if's consecutivos desnecessários.

    Item: Ao invés de usar múltiplas condições similares, use um switch case.

20. Passou.

    Item: Para desenvolver evitando códigos repetidos antes de loops, use 'do-while'.

COMO FAZER O CÓDIGO

1. Passou.

   Item: Fazer um planejamento da forma e na ordem de desenvolvimento.

2. Passou.

   Item: Separar o código em módulos, para melhor organiza-lo e mantê-lo.

3. Passou.

   Item: Criar nomes de variáveis que fazem sentido e que sejam simples no contexto.

4. Passou.

   Item: Quando algo não estiver claro, ou não foi possível criar um nome simples de variável, comente o

código.

5. Passou.

   Item: Comentar sempre a funcionalidade da função antes de sua criação.

6. Não passou. Para corrigir esse defeito, foi refatorada a função “Vinte_Perguntas”, “Respostas”, “Constroi_Manual”, “Pergunta_Final”, removendo redundâncias e melhorando comentários e nomes de variáveis. Foi também trocado o nome da função “ConstroiNo” para “PosicaoNo”.

   Item: Sempre após desenvolver uma função que esteja funcionando, refatore ela.

7. Passou.

   Item: Evitar ao máximo usar variáveis globais, pois isso pode complicar o código caso usado de forma errada.

8. Passou.

   Item: Utilizar enumeradores para evitar usar números que podem deixar o código difícil de compreender futuramente.

9. Passou.

   Item: Organização e indentação do código para melhor legibilidade.

10. Não Passou. Foi feita a modificação nas funções “Resposta”, “Vinte_Perguntas”, “Pergunta_Final”, “Constroi_Manual”, ajustando as variáveis para unsigned.

    Item: Caso tenha uma variável inteira que é sempre positiva, use ‘unsigned’.

11. Passou.

    Item: Caso o valor de uma variável seja fixa, utilize constantes.

12. Passou.

    Item: Caso esteja usando alguma variável constante que possa eventualmente necessitar de mudanças, use o #define.

13. Não passou. Foi corrigido esse problema refatorando o nome da função “ConstroiNo” para “PosicaoNo”.

    Item: Criar nomes de funções que façam sentido e se possível que indiquem o que a função faz.

14. Passou.

    Item: Checar os tipos ideais para todas as variáveis das funções e dos módulos.

15. Passou.

    Item: Caso esteja mexendo com alocação de memória, sempre lembrar de liberá-las pós uso.

16. Passou.

    Item: Não escreva linhas muito cumpridas. Quando possível, utilize múltiplas linhas.

17. Passou.

    Item: Evite usar vetores, sempre que possível aloque dinamicamente a memória.

18. Passou.

    Item: Sempre que possível quebre uma função grande em várias curtas e simples.

19. Passou.

    Item: Analise sempre no início a melhor estrutura de dados a ser usada para o software.

20. Passou.

    Item: Use um padrão consistente para a nomeação de variáveis, funções e módulos.

COMO TESTAR O CÓDIGO

1. Passou.

   Item: Utilizar um framework de testes.

2. Passou.

   Item: Não usar impressões na tela (printf) para testes.

3. Passou.

   Item: Os testes devem ser feitos com o pensamento de todas as situações de entrada.

4. Passou.

   Item: Os testes devem abranger bem o código.

5. Passou.

   Item: Devem ser feitos testes para analisar situações de erro e parâmetros inválidos.

6. Passou.

   Item: Fazer o desenvolvimento orientado a testes.

7. Passou.

   Item: Saiba bem quando usar as funções de teste do framework.

8. Passou.

   Item: Verifique a porcentagem do código que foi testada.

9. Passou.

   Item: Faça testes rigorosos em partes críticas/sensíveis do código.

10. Passou.

    Item: Faça testes para todas as condições do código.

11. Passou.

    Item: Nomeie os testes de forma a especificar o que está sendo testado.

12. Passou.

    Item: Separe os testes por grupos específicos de parâmetros a serem testados.

13. Passou.

    Item: Execute o programa no Valgrind com os parâmetros '—track-origins=yes -v' para testar as variáveis e as funções de liberação da memória.

14. Passou.

    Item: Usar um módulo separado para os testes.

15. Passou.

    Item: Quando surgir uma possibilidade ainda não pensada, faça o teste.

16. Passou.

    Item: Comente antes ou dentro do teste detalhadamente sobre o que está sendo testado.

17. Não passou. Foram adicionadas no código assertivas de entrada e saída.

    Item: Utilize junto ao framework de testes, bibliotecas que possam auxiliar a encontrar erros.

18. Não passou. Foi adicionado uma assertiva dentro da função "PosicaoNo" para verificar o caractere '\0' no final da string concatenada.

    Item: Caso seu programa mecha com concatenação e manipulações de strings, teste se o final da string contém o caractere '\0'.

19. Passou.

    Item: Sempre teste seu programa para vazamentos de memória.

20. Não passou. O programa necessita do input do usuário para testar algumas de suas funções, não é possível mudar isso.

    Item: Evite ao máximo testes que necessitem de comandos do usuário para passarem.

COMO DEPURAR O CÓDIGO

1. Passou.

   Item: Caso alguma função não aparenta estar executando uma parte do código, use breakpoints com o GBD para depurar.

2. Passou.

   Item: Use ferramentas de análise estática para depurar bugs e comportamentos inesperados.

3. Passou:

   Item: Caso alguma variável esteja alterando seu valor de forma não esperada, analise-a usando o watch do GDB.

4. Passou.

Item: Sempre depure seu programa com o Valgrind.

5. Passou.

   Item: Caso encontre dificuldades em visualizar a execução do programa pelo GDB, use o DDD.

6. Passou.

   Item: Procure uma IDE com compilador interno ou com funções de ajuda para depurar.

7. Passou.

   Item: Utilize o DBX para depurar acessos inválidos.

8. Passou.

   Item: Caso encontre dificuldades em visualizar a depuração pelo Valgrind, use o 'MemProf'.

9. Não passou. Foram adicionadas assertivas em todas as funções.

   Item: Usar assertivas para reduzir o escopo de erro e ajudar na depuração.

10. Passou.

    Item: Utilize a função 'Step' do GDB ao depurar uma função com outras chamadas de funções.

11. Passou.

    Item: Use o comando 'backtrace' para visualizar parte do código executado e as linhas futuras no GDB.

12. Passou.

    Item: Utilize o teste de 'core' do GDB para depurar falhas de segmentação.

13. Passou.

    Item: Não use 'printf' para depurar o código.

14. Passou.

    Item: Utilize a ferramenta ideal para cada tipo de problema que se quer depurar.

15. Passou.

    Item: Ao depurar funções que aparentam estar muito complexas ou desorganizadas, considere refazer partes da função.

16. Passou.

    Item: Quando possível, refatore uma função em várias funções menores, para depurar suas partes.

17. Passou.

    Item: Analise as mensagens de warnings e erros do compilador para depurar o programa.

18. Passou.

    Item: Cheque bem a lógica das assertivas usadas para depurar.

19. Passou.

Item: Ao encontrar um bug em uma chamada de função, cheque bem os argumentos de entrada na depuração.

20. Passou.

    Item: Ao depurar, caso encontre múltiplos erros iguais, em uma mesma sequência de código, considere arrumar essa sequência e torna-la uma nova função.