

Centro Universitário Estácio

Campus Parangaba

Ciência da Computação

Desenvolvimento de uma campanha para personagem em um RPG.

Introdução à Programação Estruturada em C.

Carlos Ryan Sousa Guimarães – 201903284481 Narihel Kauã de Lima Souza – 202204137372 Rinaldo Uchoa Pontes Junior – 202203065981

Thayssa Kelly Sousa Osterno – 202203208195

Professor(a): Maria Edineuda Teixeira Pinto

Fortaleza – CE 2022.1

# Introdução

A ideia de montar o próprio RPG (role-playing game), que consiste em um jogo em que os jogadores assumem o papel de personagens imaginários em um mundo fictício, é uma forma de testar as habilidades adquiridas na linguagem de programação em C, trabalhando todos os comandos de forma lúdica.

O intuito de montar esse código refere-se, principalmente, à flexibilidade que a linguagem C oferece para montar qualquer tipo de projeto. Dessa forma, é possível implementar o RPG, com toda sua complexidade, nessa linguagem através da lógica de programação. Neste código, utiliza-se o universo de Shrek como ilustração.

# Fluxograma

Diagrama

Descrição gerada automaticamente

Legenda:

Processamento de dados

Decisões

Ínicio ou Fim do programa

Entrada de dados com o uso do teclado

Fluxo de dados

1. **Ficha dos Jogadores**

Texto

Descrição gerada automaticamente com confiança média

**Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente**

**Interface gráfica do usuário, Texto

Descrição gerada automaticamente**

**Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente**

1. **Características e Funcionamento do Programa**

O código foi criado apartir da utilização das seguintes **bibliotecas**:

**<stdio.h>:** é responsável pela entrada e saída dos dados, ou seja, a existência das funcões printf() e scanf(), que fazem a saída e a leitura dos dados, respectivamente, não são propriamente de C, mas fazem parte dessa biblioteca;

**<stdlib.h>:** é propriamente o cabeçalho do código, ela possui funções envolvendo alocação de memória, controle de processos, conversões, entre outras.

**<time.h>:** apresenta funções e tipos de dados para manipular informações de tempo e datas.

**<locale.h>:** a utilização dda biblioteca locale.h e da função setlocale() configurada adequadamente vai garantir que caracteres como **“ç”** e acentuação sejam exibidos normalmente em nosso programa.

Neste código, utilizamos as seguinte **estruturas de repetição**:

**While:** uma das condições que podemos usar é o **enquanto**, ou seja, enquanto a expressão booleana for verdadeira o algoritmo executa o bloco proposto, por fim é necessário que algo dentro do bloco altere a condição.

**Do While:** permite entrar no laço novamente para depois testar a condição, contrária a estrutura de repetição anterior.

Dentre as **estruturas condicionais**, utilizamos as seguintes:

**Comando if:** expressa uma condição, é representado por “se”.

**Comando if-else:** quando existe um conjunto de instruções a ser executado, caso o valor da condição seja falso, utiliza-se esse comando.

Os tipos de variáveis são:

**Int:** representa um número inteiro;

**Float:** representa um número real;

**Char:** define um caractere. Ademais, quando há um conjunto de caracteres forma-se uma string, ou seja, texto.

1. **Ambiente de Desenvolvimento**

Neste programa, foi utilizado do Bloodsheld Dev C++

Há várias razões da escolha, outras alternativas poderiam ser o Code Blocks, Visual Studio e entre

outros. Mas devido há Universidade Estácio nos conceder essa proximidade com o Programado Dev C++,

diversos alunos tiveram essa familiaridade com a interface do compilador.

Alguns das vantagens que notamos ao utilizar o Dev C++:

* Interface simples, fácil e amigável à novos programadores;
* Possibilidade de alterar fontes e temas (Como o modo escuro, totalemente agrádavel à quem passa horas no computador;
* Leveza e portabilidade do programa;

**Texto

Descrição gerada automaticamente**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

🡨Aqui podemos ver as diversas opções de compilar o seu código, além de verificar erros de sintaxe, este que indica linha e muitas vezes lhe explica o que está faltando.

# Texto Descrição gerada automaticamente

No exemplo abaixo, realizei um erro proposital ao retirar o “;” do final de uma instrução e ao compilar o programa indica a linha e diz o que está faltando: “É esperado um ‘;’ antes do ‘Scanf’.”

**6. Implementação**

Texto

Descrição gerada automaticamente

Primeiramente, utilizamos o setlocale para permitir o uso de letras especiais do nosso português. Assim, podemos começar nosso programa criando as váriaveis de começo, que futuramente iremos utilizar.

As que possuem o nome “int”, são váriaveis para números inteiros onde aí que entra os atributos como Força, agilidade, inteligencia e etc. O Char é para entrada de um conjunto de caracteres, nestes que escolheremos o Nome do personagem e quais habilidades você irá desejar.

**Texto

Descrição gerada automaticamente**

Na execução do programa, realizamos primeiramente a criação do seu personagem, utilizei meu próprio nome apenas por questão de exemplo. E assim podemos prosseguir com o uso do “printf” que este vai apresentar uma mensagem na tlea pedindo para você escolher a sua raça que deseja. Assim, em seguida, usamos o “scanf” para você usúario inserir um dado.

O uso do “\n” é muito importante para organização dos textos, assim podendo pular linhas e separar para não causar confusão.

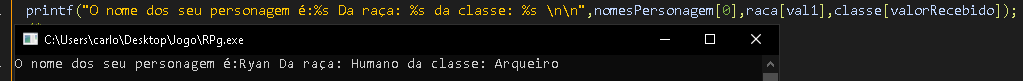
Uso do ‘goto’:

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

Aqui podemos ver que ele irá perguntar se você deseja aquela escolha mesmo, então se você desejar voltar, aperte 1 e assim o goto irá realizar sua função de retonar a certo ponto onde foi definido o “selectRaca” lá em cima.

Utilizaremos essas mesmas funções tanto para seleção de Raças como a seleção de Classes.



Este ‘printf’ em especial, exibirá em sequênca o seu nome, raça e classe com o uso de strings “%s”.

Texto

Descrição gerada automaticamente

Devido minha escolha do exemplo ter sido um Humano, o programa irá apresentar um texto exclusivo para minha escolha. Se estivesse escolhido o Orc, outro texto totalmente diferente irá aparecer para descrever sua raça.

Logo em sequência, podemos ver que ele irá pedir para você distrubir seus pontos, estes que tem o limite de apenas 10 para escolher.

Distribuição de pontos:

Texto

Descrição gerada automaticamente

Cada raça terá suas próprias caracteristicas para distribuir pontos, usamos o Do While para definir a execução da distribuição.

Se você inserir mais de 10 em cada uma, o programa dirá que ultrapassou o limite ou caso não tenha atribuido todas, qe você precisa inserir corretamente e bem distribuida.

Isso tudo ocorrerá enquanto você distribuir do jeito certo, senão, iremos executar um goto para retornar para a distribuição de pontos.

Criação de habilidades:

Texto

Descrição gerada automaticamente

Aqui, podemos ver simples usos do ‘printf’ e ‘scanf’, utilizaremos de nossa criatividade para criação de habilidades diversas.

Por fim, exibiremos suas escolhas e daremos um ‘system(“pause”);’ para o programa não passar muito tempo para a próxima parte do programa e logo em seguida um ‘system(“cls”) para limpar a tela.

Narração e Escolhas:

Texto

Descrição gerada automaticamente

Aqui iremos para a parte menos técnica e mais criativa, utilizaremos do Printf para descrever a história e realizar dialógos de NPC’s (Non-Playabe Characters) conforme quisermos.

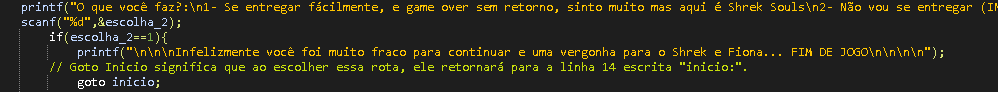
Como a temática se passa no universo do Shrek, tentamos ao máximo estudar e conhecer melhor os detalhes. O primeiro elemento foi a ambientação, por ser um ambiente medieval e fantasioso, temos essa chance de se aproveitar de diversos detalhes, tais como fadas, orgros, cavaleiros, tavernas e etc.

Texto

Descrição gerada automaticamente

Na escolha de rotas, temos a chance de fazer várias ramificações de escolhas e consequências, mas por se tratar de um one-shot (Um teste resumido da ideia), teremos escolhas mais simplificadas.

Com o uso dos ‘If’, ‘Else If’ e ‘Else’, podemos elaborar várias formas de decisões, como podemos ver na foto acima, escolher uma das três opções, terá resultados diferentes.



Novamente podemos ver o uso do ‘Goto’ para retornar até a linha determinada pelo “inicio”.

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média Após a decisão, usamos o system(“pause”) e system(“cls”) novamente para limpar a tela e passar para o início do combate.

O Combate:

Texto

Descrição gerada automaticamente

Essa foi a parte mais complicada para se fazer... O combate se passa dentro de um Do While, ou seja, ele vai se repetir até que tal condição seja alcançada.

Nesse caso, podemos traduzir tudo para: Faça, enquanto a vida do cavaleiro for maior que zero ou a vida do jogador for maior que zero.

Aqui utilizaremos do recurso ‘srand’ que irá gerar um numéro aleatório entre 1 e 10, onde maior que 6 é acerto e menor que 6 é um erro no combate. Caso tenha um acerto no ataque, iremos atualizar a vida do jogador ou do inimigo subtraindo sua vida com o dano base do jogo que é 10, agora caso haja um erro no ataque, aparecerá uma mensagem afirmando que errou e irá passar o turno para o inimigo.

Tudo isso até que a vida de alguém chegue a zero.

1. **Resultados Experimentais**

Introdução:

Texto

Descrição gerada automaticamente

Primeiro resultado vimos a introdução com texto, onde a cada ‘printf’, teremos um system(“pause”).

Nome e escolhas do personagem

Texto

Descrição gerada automaticamente

Aqui já podemos inserir o nome do nosso personagem e logo depois pedirá a raça, classe que você deseja e se você confirma.

Texto

Descrição gerada automaticamente Atributos:

Os atributos como podemos ver, devem ser distribuidos nos respectivos lugares, no exemplo, depositei 5 em força, 2 em agilidade, 1 em inteligência (um pouco burro), 0 em constituição e 2 em carisma (orc carismatico?).

Texto

Descrição gerada automaticamente

Como podemos ver, caso você distribua mais do que 10, ele avisa e repete o processo pra que faça direito.

Ou até mesmo caso não coloque nada, ele avisa e pede novamente.

Escolha de habilidades

Texto

Descrição gerada automaticamente

Aqui temos a liberdade de deixar a criatividade rolar solta (podendo escolher até mesmo essas que eu, um cara de 20 anos nas costas tiver a coragem de escrever)

A narração:

Texto

Descrição gerada automaticamente

A narração descrevendo o ambiente funcionando perfeitamente com vários printf e por fim pedindo para você fazer uma escolha de rota, onde cada uma levará para um dialógo ou acontecimento diferente.

1

Texto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente

3

Texto

Descrição gerada automaticamente

2

Texto

Descrição gerada automaticamenteUso do goto na prática:

4

Texto

Descrição gerada automaticamente O tão esperado combate:

O combate é básicamente uma repetição até que alguém morra, por ser um jogo de turno, é bom ter paciência.

Como podemos ver, temos a escolha com ‘If’ se quer atacar, ao atacar, ele pede pra você rolar o dado (digitando qualquer coisa) e ele irá retonar um número aleatório e uma consequência. E sucesso, ao tirar menos de 2, você recebe uma mensagem humilhante

Texto

Descrição gerada automaticamente

Em seguida temos o turno do inimigo e ele falhou, ou seja, tirou menos de 6. Resolvi não colocar o printf para o que ele tirou, pela emoção.

Texto

Descrição gerada automaticamente

Logo depois, rolamos os dados de novo e dessa vez acertamos, assim diminuindo a vida do inimigo.

Texto

Descrição gerada automaticamente

Dessa vez o inimigo teve a sorte de te acertar e diminuindo assim sua vida.

Sua vida base é 100, enquanto o ataque inimigo é 10.

Texto

Descrição gerada automaticamente

Depois de rolar bastante os dados, podemos ver que minha vida caiu para 19 pois tive bastante azar no combate de inicio, mas posteriormente consegui vencer por ser o primeiro inimigo.

1. **Conclusão**

Código do Jogo:

