UERJ – 457

SEJA APROVADO OU MORRA TENTANDO

- Jogo do tipo RPG que é jogado no próprio terminal do computador.
- O objetivo de quem joga é conseguir ser aprovado na disciplina de LP1, em que cada decisão tomada impacta o resultado final.
- Para a construção do jogo, foram utilizados structs, ponteiros, alocação dinâmica de memória, passagem de estruturas por referência, manipulação de vetores e strings, e abertura e escrita de/em arquivos.

STRUCT

```c
typedef struct status{
    int *mochila;
    char nome [20];

} status;

void inicio (int *mochila);
void ernesto(status *jogador);
void vazio(status *jogador);
void bolado(status *jogador);
void novato(status *jogador);
void jamaicano(status *jogador);

void ordenar(int *vetor);
int busca(int *vetor, int item);
char leitura(FILE *arq);
void salvar_mochila(int *vetor);
status jogador;
```

A utilização de struct se dá na criação de um novo tipo de dado chamado de "status", em que esse dado contém dado do tipo ponteiro para inteiro que define uma mochila e um dado do tipo char que define uma string para receber o nome do jogador.

PONTEIROS E PASSAGEM DE ESTRUTURA POR REFERÊNCIA

```
if(busca(jogador.mochila, 1) == 1){// Noitebus
    ernesto(&jogador);
}

else{
    if(busca(jogador.mochila, 5) == 0){ // Ônibus vazio
        vazio(&jogador);
    }

    else if(busca(jogador.mochila, 2) && busca(jogador.mochila, 4) == 0){  // Motorista Bolado
        bolado(&jogador);
    }

    else if(busca(jogador.mochila, 3) && busca(jogador.mochila, 4) == 0){// Motorista Novato
        novato(&jogador);
    }

    else{ // Motorista Jamaicano
        jamaicano(&jogador);
    }
}
```

A utilização de ponteiros se faz em praticamente todo o código do jogo, um exemplo e dá na estrutura de condições que definem qual será o cenário que o jogador entrará, sendo esse cenário escolhido de acordo com os retornos dados por uma função de busca realizada na mochila dos jogadores, a mochila é passada pra função por referência.

ALOCAÇÃO DINÂMICA DE MEMORIA

```
printf("\n----------[OPÇÕES]----------");
jogador.mochila = (int *) malloc (3 * sizeof (int));
inicio(jogador.mochila);
getchar(); clrscr();
```

No início do jogo é necessário escolher alguns itens para levar na mochila. A mochila é criada como um ponteiro para um inteiro que tem seu tamanho redefinido por meio da alocação dinâmica feita pela função malloc. Dessa forma, a mochila pode se tornar um ponteiro para vetor de inteiro com três posições.

MANIPULAÇÃO DE VETORES E STRINGS

```
653     while ((c = fgetc(arq)) != EOF){
654 
655         if (c == '@'){
656             c = ' ';
657             printf ("%s", jogador.nome);
658         }
659 
660         printf("%c", c); // Imprime o caractere
661         fflush(stdout);  // Garante que o caractere seja mostrado imediatamente
662         sleep(30);      // Delay de 100 milissegundos entre cada caractere
663         if (c == '\n') {
664             getchar(); // Espera pelo Enter do usuário
665             clrscr();
666         }
667     }
```

Acima temos um bloco de código ao qual toda vez que houver um '@' no arquivo de texto este será substituído pelo nome do jogador

```
576 void ordenar(int *vetor)
577 {
578     int key, j;
579 
580     for(int i =1; i<3; i++)
581     {
582         key = vetor[i];
583         j = i-1;
584         while( key < vetor[j] && j >=0)
585         {
586             vetor[j + 1] = vetor[j];
587             j--;
588 
589         }
590         vetor[j+1] = key;
591 
592     }
593 
594 }
595 
```

O bloco de código acima realiza a ordenação do vetor, o que facilita ao fazer as comparações para decidir os caminho do jogo

SALVAR E CARREGAR DADOS DO JOGO

```c
void salvar_mochila(int *vetor)
{
    FILE *p;
    p = fopen("save", "a");
    fprintf (p, "\nJogador: %s\nMochila:\n", jogador.nome);

    for(int i =0; i <3; i++){

        switch(vetor[i]){
            case 1:
                fprintf(p, "óculos\n");
                break;
            case 2:
                fprintf(p, "água\n");
                break;
            case 3:
                fprintf(p, "Sanduiche\n");
                break;
            case 4:
                fprintf(p, "Celular\n");
                break;
            case 5:
                fprintf(p, "GameBoy\n");
                break;
        }
    }
    fprintf(p, "\n");
    fclose (p);
}
```

```c
if (menu == 3){


    p = fopen("save", "r");
    if (p == NULL) {

    printf("Save inexistente.\n");
    return 1;
    }
    while ((c = fgetc(p)) != EOF) {
        printf("%c", c);
    }
    getchar(); clrscr();
    fclose(p);
}
```

Para salvar dados gerados em cada execução do jogo, assim como para abrir os textos da história, são utilizados arquivos.

Com isso, a quantidade de linhas do código é reduzida consideravelmente.

A leitura dos arquivos se faz a cada vez que o jogador realiza decisões no jogo, em que a decisão resulta em algum arquivo determinado a ser aberto.

A cada decisão também é aberto um arquivo em modo de escrita, para assim salvar quais foram as decisões do jogador.

ILUSTRAÇÕES DO JOGO

MOTORISTA BOLADO

MOTORISTA JAMAICANO

MOTORISTA NOVATO

ÔNIBUS VAZIO

15/06/2024

NOITEBUS

PROF. "LAISSA"