

ATIVIDADE EXTRACURRICULAR III

RELATÓRIO PROJETO JOGO DA SENHA

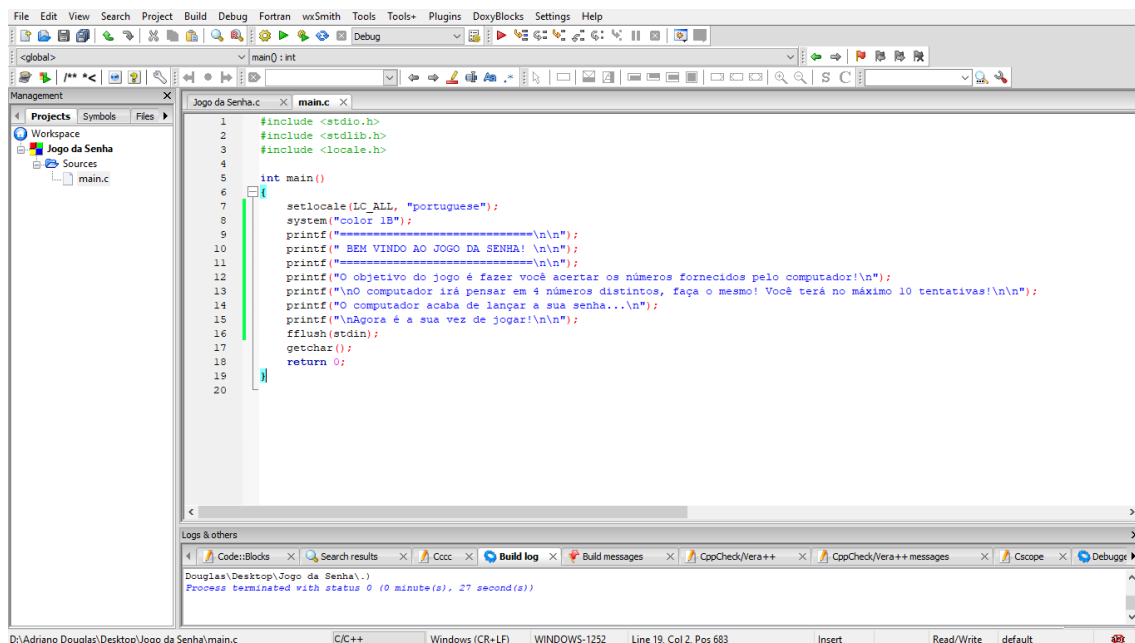
Alunos:

Adriano Douglas Reis Rosa.

Rafael de Souza Damasceno.

Primeiramente foi definida a cor de fundo e a cor do texto do jogo, para isso foi utilizado o comando `system("color 1B")` onde o 1 significa o código da cor de fundo e a letra B corresponde ao código da cor do texto, foi criado também uma tela de boas-vindas explicando para o usuário as principais regras do jogo, nessa parte utilizamos basicamente alguns comandos `printf("")`.

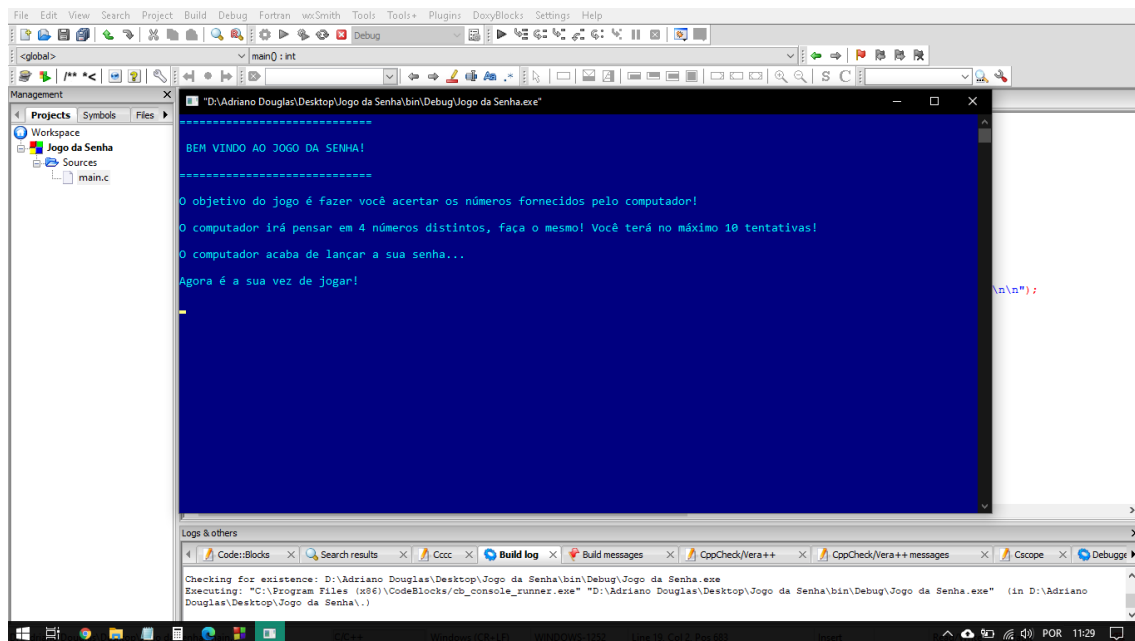
`setlocale(LC_ALL, "portuguese");` // é chamado para utilizar caracteres especiais.



The screenshot shows a C++ IDE with the following code in `main.c`:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <locale.h>
4
5 int main()
6 {
7     setlocale(LC_ALL, "portuguese");
8     system("color 1B");
9     printf("===== \n\n");
10    printf(" BEM VINDO AO JOGO DA SENHA! \n\n");
11    printf("===== \n\n");
12    printf("O objetivo do jogo é fazer você acertar os números fornecidos pelo computador!\n");
13    printf("O computador irá pensar em 4 números distintos, faça o mesmo! Você terá no máximo 10 tentativas!\n\n");
14    printf("O computador acaba de lançar a sua senha...\n");
15    printf("Agora é a sua vez de jogar!\n\n");
16    fflush(stdin);
17    getch();
18    return 0;
19 }
20
```

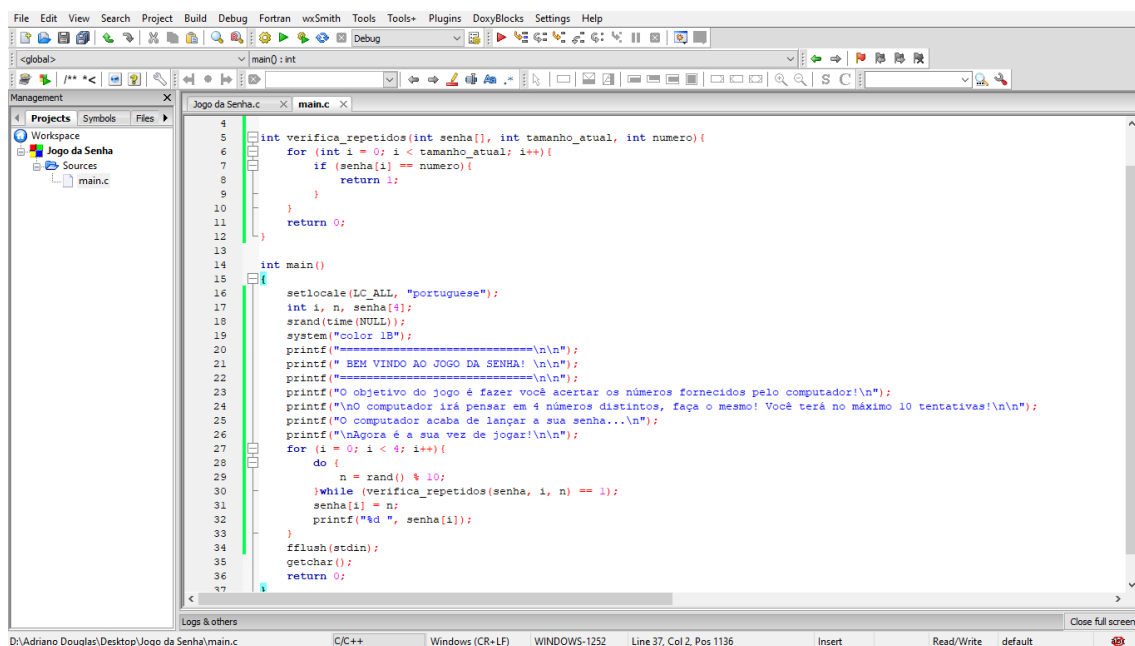
The IDE interface includes a menu bar (File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, Help), a toolbar, a Project Explorer on the left showing 'Jogo da Senha' and 'main.c', and a bottom status bar indicating 'C/C++', 'Windows (CR+LF)', 'WINDOWS-1252', 'Line 19, Col 2, Pos 683', and 'Read/Write default'.



Em seguida foi criada uma função para gerar números aleatórios sem repetição, esses números foram guardados em um vetor `senha[4]`, que será os números que o usuário deve acertar.

`srand(time(NULL))` // é chamado para que seja possível gerar novos números aleatórios.

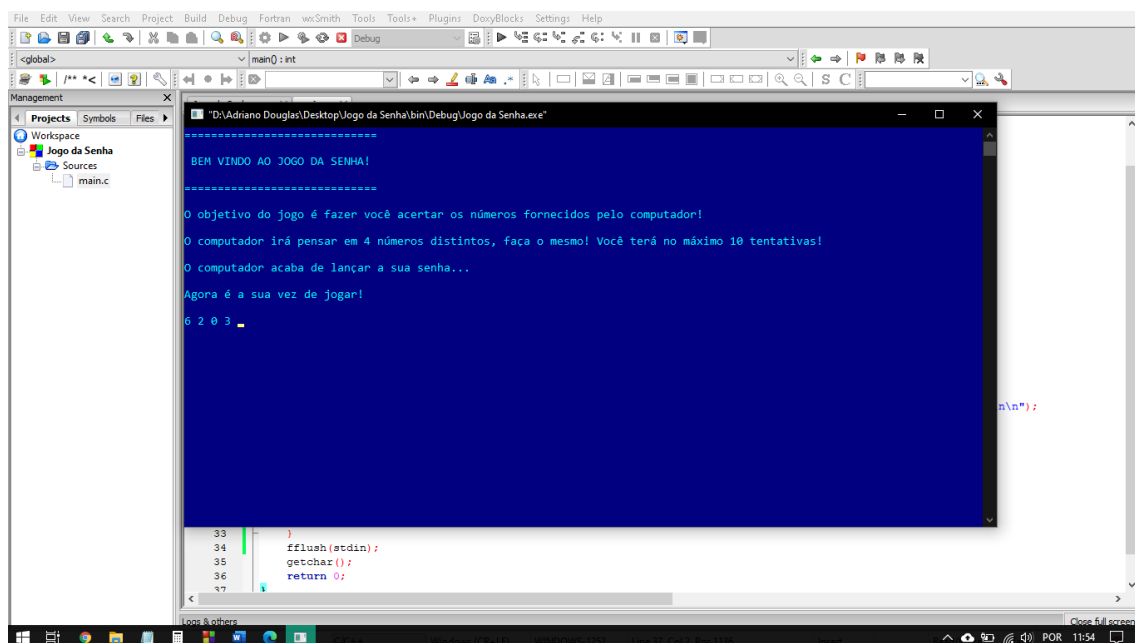
`rand() % 10 + 1` // é responsável por gerar números aleatórios de 1 até 10.



A função `verifica_repetidos` funciona da seguinte forma:

Serão passados três parâmetros, o vetor a ser verificado, o índice atual que corresponde ao número de posições já preenchidas no vetor e um número que vai ser procurado até essa posição do vetor, se esse número já fizer parte do vetor a função retornara o valor 1, senão a função irá retornar o valor 0.

No método main(), dentro de um for foi inserido um do while que vai permitir que números aleatórios sejam sorteados até que a função verifica_repetidos retorne o valor 0, pois se ela retornar o valor 1 significa que aquele número que acaba de ser gerado já faz parte do vetor senha e então deve ser gerado outro número até que a função retorne 0, dessa forma no fim da execução do for todos os números do vetor senha[4] serão distintos.

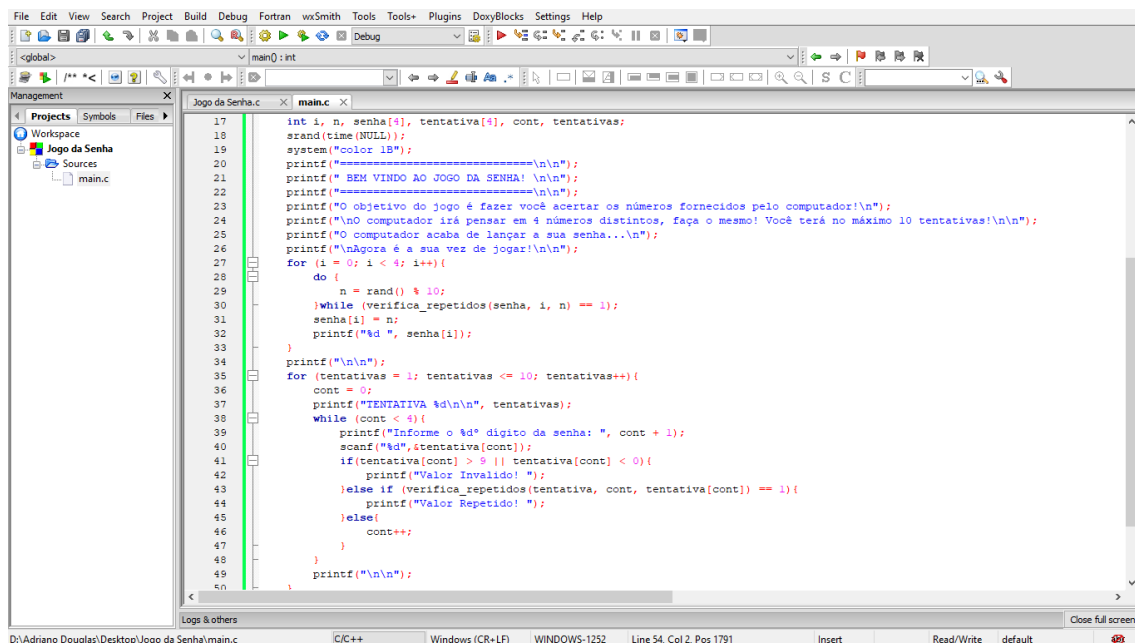


```
33 }
34 fflush(stdin);
35 getch();
36 return 0;
37 }
```

Depois de criarmos a tela de boas-vindas e gerar números aleatórios sem repetir, criamos um for iniciando com 1 e rodando até completar 10, com isso o usuário teria 10 tentativas de acertar a senha. Para guardar os dígitos de cada tentativa do usuário criamos um vetor tentativa[4] onde cada índice desse vetor irá guardar o valor que o usuário digitar, o objetivo disso era depois verificar cada posição do vetor tentativa[4] com o vetor senha[4] e gerar o resultado dessa tentativa em outro vetor.

Sabendo que o usuário não podia digitar números maiores que 9 e menores que 0, e também esses valores precisavam ser distintos, criamos um contador contendo sendo iniciado com 0 a cada nova tentativa do usuário, dentro de um while

colocamos duas condições, a primeira era: se o o valor informado pelo usuário for maior que 9 e menor que 0, seria exibido a mensagem “Valor invalido!” e dessa forma o usuário teria que informar outro número até que a condição seja feita, a outra condição era a seguinte: se o número digitado pelo usuário já tivesse sido informado anteriormente ele teria que informar outro número, para isso aproveitamos a função `verifica_repetidos`, e passamos como paramentos o vetor `tentativa[4]`, a posição `cont` e o número a ser procurado, ou seja a `tentativa[cont]`, se esse valor já estivesse sido informado a função retornaria 1 e a mensagem “Valor repetido!” apareceria, então seria pedido um novo número, dessa forma o `while` rodaria até o `cont` completar 4, pois enquanto as duas condições não forem cumpridas o contador não iria aumentar.



```
17 int i, n, senha[4], tentativa[4], cont, tentativas;
18 srand(time(NULL));
19 system("color 1B");
20 printf("=====\n\n");
21 printf(" BEM VINDO AO JOGO DA SENHA! \n\n");
22 printf("=====\n\n");
23 printf("O objetivo do jogo é fazer você acertar os números fornecidos pelo computador!\n");
24 printf("\nO computador irá pensar em 4 números distintos, faça o mesmo! Você terá no máximo 10 tentativas!\n\n");
25 printf("O computador acaba de lançar a sua senha...\n");
26 printf("\nAgora é a sua vez de jogar!\n\n");
27 for (i = 0; i < 4; i++){
28     do {
29         n = rand() % 10;
30     } while (verifica_repetidos(senha, i, n) == 1);
31     senha[i] = n;
32     printf("%d ", senha[i]);
33 }
34 printf("\n\n");
35 for (tentativas = 1; tentativas <= 10; tentativas++){
36     cont = 0;
37     printf("TENTATIVA %d\n\n", tentativas);
38     while (cont < 4){
39         printf("Informe o %dº dígito da senha: ", cont + 1);
40         scanf("%d", &tentativa[cont]);
41         if (tentativa[cont] > 9 || tentativa[cont] < 0){
42             printf("Valor Invalido! ");
43         } else if (verifica_repetidos(tentativa, cont, tentativa[cont]) == 1){
44             printf("Valor Repetido! ");
45         } else {
46             cont++;
47         }
48     }
49     printf("\n\n");
50 }
```

```
-----
O objetivo do jogo é fazer você acertar os números fornecidos pelo computador!
O computador irá pensar em 4 números distintos, faça o mesmo! Você terá no máximo 10 tentativas!
O computador acaba de lançar a sua senha...
Agora é a sua vez de jogar!
6 4 3 0

TENTATIVA 1
Informe o 1º dígito da senha: 1
Informe o 2º dígito da senha: 2
Informe o 3º dígito da senha: 1
Valor Repetido! Informe o 3º dígito da senha: -3
Valor Invalido! Informe o 3º dígito da senha: 2
Valor Repetido! Informe o 3º dígito da senha: 35
Valor Invalido! Informe o 3º dígito da senha: 3
Informe o 4º dígito da senha: 4

TENTATIVA 2
Informe o 1º dígito da senha: 46
                                     cont++;
47                                     }
48                                     }
49                                     printf("\n\n");
50                                     }
51                                     }
```

Depois das verificações feitas para o computador e para o usuário, agora era a parte principal do algoritmo, onde depois de ele verificar novamente as funções estabelecidas, ela irá retornar para o usuário, informações sobre o andamento do jogo.

De início, o compilador iria mostrar a sequência fornecida pelo usuário através de um for. Logo em seguida, os valores indicados por cada função iam ser chamada e verificando o que se pede. Depois de todo o percurso feito por cada função, a variável resultado iria fornecer uma sequência de números (0,1,-1), onde cada um tinha um significado para os números fornecidos do usuário em relação a senha do computador.

Para responder os significados dos números, foi criado um sistema de dicas, uma vez que o usuário até a sua tentativa 5 não tivesse acertado a solução dada, o programa irá mostrar os padrões dos números:

- 1 para cada valor que não fizer parte da senha;
- 0 para cada valor correto e na posição errada;
- 1 para cada valor correto e na posição correta.

main.c - Code::Blocks 17.12

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global> main(): int

Management

Start here main.c

```
78 }else if (verifica_repetidos(tentativa, cont, tentativa[cont])){
79     printf("Valor Repetido! ");
80 }else{
81     cont++;
82 }
83
84 printf("\nSua tentativa: ");
85 for (i = 0; i < 4; i++){
86     resultado[i] = -2;
87     printf("%d\t", tentativa[i]);
88 }
89 nao_faz_parte(senha, tentativa, resultado);
90 posicao_certa(senha, tentativa, resultado, &resp);
91 posicao_errada(resultado);
92 printf("\nResultado: ");
93 for (i = 0; i < 4; i++){
94     printf("%d\t", resultado[i]);
95 }
96 if ((tentativas == 5) && (resp != 4)){
97     printf("\n\nDICAS:\n");
98     printf("a) 1 significa que o valor está na posição correta\n");
99     printf("b) 0 significa que o valor está na posição errada\n");
100     printf("c) -1 significa que o valor não faz parte da senha ");
101 }
102 printf("\n\n");
103 if (resp == 4){
104     break;
105 }
106 }
107 if (resp == 4){
```

Logs & others

C:\Users\rafael\OneDrive\Documentos\codigos em c\1 periodo\jogo da se\C/C++ Windows (CR+LF) WINDOWS-1252 Line 105, Col 10, Pos 3375 Insert Read/Write default 13:47 23/06/2020

main.c [Jogo da Senha] - Code::Blocks 17.12

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global> main(): int

Management

Jogo da Senha.c main.c

```
72 while (cont < 4){
73     printf("Informe o %dº dígito da senha: ", cont + 1);
74     scanf("%d", &tentativa[cont]);
75     if (tentativa[cont] > 9 || tentativa[cont] < 0){
76         printf("Valor Inválido! ");
77     }else if (verifica_repetidos(tentativa, cont, tentativa[cont]) == 1){
78         printf("Valor Repetido! ");
79     }else{
80         cont++;
81     }
82 }
83 printf("\nSua tentativa: ");
84 for (i = 0; i < 4; i++){
85     resultado[i] = -2;
86     printf("%d\t", tentativa[i]);
87 }
88 nao_faz_parte(senha, tentativa, resultado);
89 posicao_certa(senha, tentativa, resultado, &resp);
90 posicao_errada(resultado);
91 printf("\nResultado: ");
92 for (i = 0; i < 4; i++){
93     printf("%d\t", resultado[i]);
94 }
95 if (tentativas == 5 && resp != 4){
96     printf("\n\nDICAS:\n");
97     printf("a) 1 significa que o valor está na posição correta\n");
98     printf("b) 0 significa que o valor está na posição errada\n");
99     printf("c) -1 significa que o valor não faz parte da senha ");
100 }
101 printf("\n\n");
102 if (resp == 4){
```

Logs & others

D:\Adriano Douglas\Desktop\Jogo da Senha\main.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 119, Col 2, Pos 3661 Insert Read/Write default 14:09

```
main.c [Jogo da Senha] - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
global>
main() : int
Management
Projects
Workspace
Jogo da Senha
Sources
main
TENTATIVA 4
Informe o 1º dígito da senha: 1
Informe o 2º dígito da senha: 2
Informe o 3º dígito da senha: 3
Informe o 4º dígito da senha: 4
Sua tentativa: 1 2 3 4
Resultado: -1 -1 0 -1
TENTATIVA 5
Informe o 1º dígito da senha: 1
Informe o 2º dígito da senha: 2
Informe o 3º dígito da senha: 3
Informe o 4º dígito da senha: 4
Sua tentativa: 1 2 3 4
Resultado: -1 -1 0 -1
DICAS:
a) 1 significa que o valor está na posição correta
b) 0 significa que o valor está na posição errada
c) -1 significa que o valor não faz parte da senha
TENTATIVA 6
Informe o 1º dígito da senha:
printf("c) -1 significa que o valor não faz parte da senha\n");
printf("\n\n");
if (tentativa == 4) {
100
101
102
}
Log & others
C/C++ Windows (CR+LF) WINDOWS-1252 Line 96, Col 35, Pos 3000 Insert Read/Write default
```

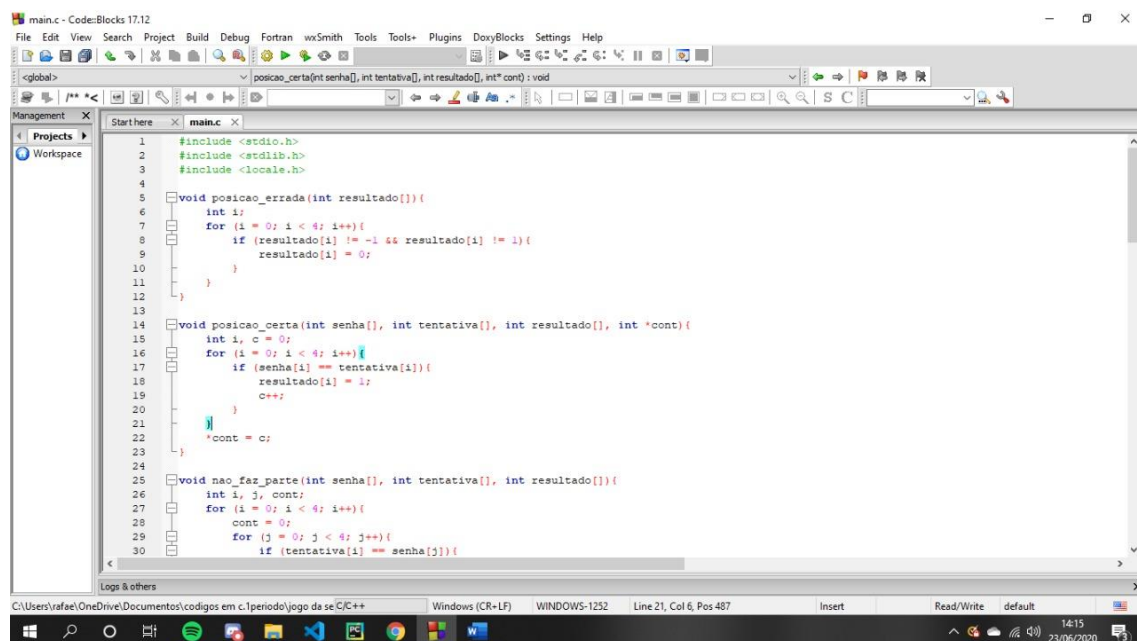
Para o sucesso da parte principal do algoritmo, era de extrema importância saber otimizar as funções utilizadas no programa. Para um código mais limpo e simples de ser lido por qualquer programador, estabelecemos uma série de funções que foram escolhidas para trabalharem separadamente, favorecendo um desempenho melhor.

Primeiramente, foi criada a função `nao_faz_parte`, onde como propriamente dita o nome da mesma, ela fazia a verificação dos números fornecidos pelo usuário, dos quais aqueles que não estivessem presentes, o vetor de resultado, que tinha o mesmo tamanho do vetor tentativa, recebia "-1", e ficava bem esclarecido que o se o -1 aparecesse abaixo do número fornecido, significaria que esse número não existia. Para o funcionamento da mesma, foi estabelecida uma estrutura de repetição simples usando duas estruturas de repetição `for`. O primeiro guardava a posição da tentativa, e o segundo as posições da senha do computador. No final, era montado um contador, onde se o mesmo continuasse valendo 0, significaria que o valor emitido pelo usuário, não fazia parte da senha do computador.

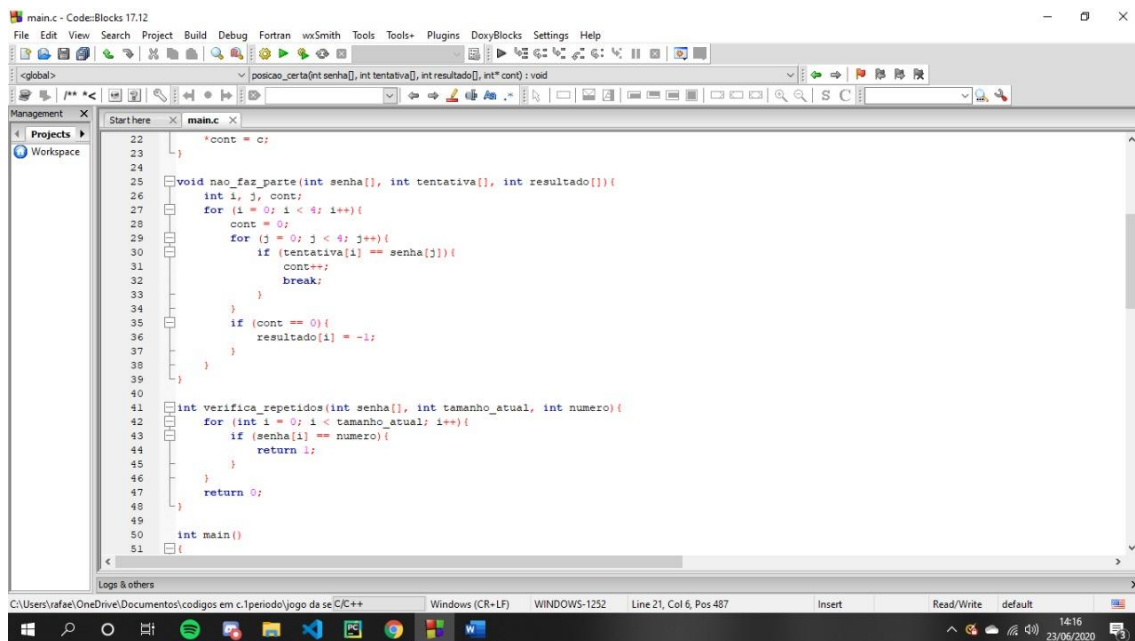
Logo após, era preciso uma nova criação de uma função que ficaria responsável para a verificação das posições dos números e seus respectivos lugares propriamente certos. A função `posicao_certa` foi dentre todas, a mais simples de se entender, onde ele simplesmente verificava posição por posição das senhas e tentativas se ambas eram iguais. Para essa condição de igualdade entre os

vetores, foi criado um simples if, onde se ele fosse verdadeiro, o vetor resultado recebia como valor 1, e aplicando na função principal ele aloca logo abaixo da tentativa, significando que a posição estava certa.

Por fim, era preciso criar uma função que verificava se algum número poderia estar nas 2 funções, mas em posições completamente erradas. Com a ajuda das funções criadas anteriormente, ficou fácil acrescentar ao valor resultado = 0, uma vez que tal número fornecido não se encaixava nas demais funções, logicamente poderia retornar ao resultado o valor 0. Mas para ter uma explicação mais didática a aceitável, criamos a função posicao_errada que funcionava de uma maneira simples onde que a partir de um laço de repetição for e outro laço de condição if, caso o resultado fosse diferente de -1 e diferente de 1, ele ia fornecer para o vetor resultado, o valor 0.



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <locale.h>
4
5 void posicao_errada(int resultado[]) {
6     int i;
7     for (i = 0; i < 4; i++) {
8         if (resultado[i] != -1 && resultado[i] != 1) {
9             resultado[i] = 0;
10        }
11    }
12 }
13
14 void posicao_certa(int senha[], int tentativa[], int resultado[], int *cont) {
15     int i, c = 0;
16     for (i = 0; i < 4; i++) {
17         if (senha[i] == tentativa[i]) {
18             resultado[i] = 1;
19             c++;
20         }
21     }
22     *cont = c;
23 }
24
25 void nao_faz_parte(int senha[], int tentativa[], int resultado[]) {
26     int i, j, cont;
27     for (i = 0; i < 4; i++) {
28         cont = 0;
29         for (j = 0; j < 4; j++) {
30             if (tentativa[i] == senha[j]) {
```

```
22 *cont = 0;
23
24 void nao_faz_parte(int senha[], int tentativa[], int resultado[]) {
25     int i, j, cont;
26     for (i = 0; i < 4; i++) {
27         cont = 0;
28         for (j = 0; j < 4; j++) {
29             if (tentativa[i] == senha[j]) {
30                 cont++;
31                 break;
32             }
33         }
34         if (cont == 0) {
35             resultado[i] = -1;
36         }
37     }
38 }
39
40 int verifica_repetidos(int senha[], int tamanho_atual, int numero) {
41     for (int i = 0; i < tamanho_atual; i++) {
42         if (senha[i] == numero) {
43             return i;
44         }
45     }
46     return 0;
47 }
48
49 int main()
50 {
51     ...
52 }
```

```
}
printf("\nSua tentativa: ");
for (i = 0; i < 4; i++) {
    resultado[i] = -2;
    printf("%d\t", tentativa[i]);
}
nao_faz_parte(senha, tentativa, resultado);
posicao_certa(senha, tentativa, resultado, &resp);
posicao_errada(resultado);
printf("\nResultado:      ");
for (i = 0; i < 4; i++) {
    printf("%d\t", resultado[i]);
}
```

Para saber se o usuário ganhou nos aproveitamos a função `posicao_certa`, para isso usamos uma variável contadora dentro da função, toda vez que um valor estiver na posição correta a variável contadora aumenta, isso por referencia porque a função `posicao_certa` não tem retorno, dentro do método `main()`, foi criada a variável `resp`, e passamos ela para a função, se o seu valor for 4 significa que o usuário ganhou e então o for se encerra com um `break`, senão as tentativas continuam até que se complete 10, e se no final das 10 tentativas o usuário não acertar ou seja a variável `resp` for menor que 4 o programa é encerrado e o jogador perde.

Função `posicao_certa` com contador por referência:

```

3
4 void posicao_certa(int senha[], int tentativa[], int resultado[], int *cont){
5     int i, c = 0;
6     for (i = 0; i < 4; i++){
7         if (senha[i] == tentativa[i]){
8             resultado[i] = 1;
9             c++;
10        }
11    }
12    *cont = c;
13}
14

```

Para ficar mais bonito, caso o jogador ganhasse mudamos a cor do texto para roxo e caso ele perdesse mudamos a cor para vermelho, usando também a função system("color").

```

93 printf("\nResultado: ");
94 for (i = 0; i < 4; i++){
95     printf("%d\t", resultado[i]);
96 }
97 if (tentativas == 5 && resp != 4){
98     printf("\n\nDICAS:\n\n");
99     printf("a) 1 significa que o valor está na posição correta\n");
100    printf("b) 0 significa que o valor está na posição errada\n");
101    printf("c) -1 significa que o valor não faz parte da senha ");
102 }
103 printf("\n\n");
104 if (resp == 4){
105     break;
106 }
107 if (resp == 4){
108     system("color 5");
109     printf("Você ganhouuuu!!! ");
110 }else{
111     system("color C");
112     printf("GAME OVER!\n\nSenha correta: ");
113     for (i = 0; i < 4; i++){
114         printf("%d ", senha[i]);
115     }
116 }
117 fflush(stdin);
118 getch();
119 return 0;
120 }
121 }
122

```

```

0 computador irá pensar em 4 números distintos, faça o mesmo! Você terá no máximo 10 tentativas!
0 computador acaba de lançar a sua senha...
Agora é a sua vez de jogar!
8 2 5 1

TENTATIVA 1
Informe o 1º dígito da senha: 8
Informe o 2º dígito da senha: 2
Informe o 3º dígito da senha: 1
Informe o 4º dígito da senha: 5
Sua tentativa: 8 2 1 5
Resultado: 1 1 0 0

TENTATIVA 2
Informe o 1º dígito da senha: 8
Informe o 2º dígito da senha: 2
Informe o 3º dígito da senha: 5
Informe o 4º dígito da senha: 1
Sua tentativa: 8 2 5 1
Resultado: 1 1 1 1
Você ganhouuuu!!!

```

main.c [Jogo da Senha] - Code::Blocks 17.12

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

global>

"D:\Adriano Douglas\Desktop\Jogo da Senha\bin\Debug\Jogo da Senha.exe"

Management

Projects

Workspace

Jogo da Senha

Sources

TENTATIVA 9

Informe o 1º dígito da senha: 1
Informe o 2º dígito da senha: 2
Informe o 3º dígito da senha: 3
Informe o 4º dígito da senha: 4

Sua tentativa: 1 2 3 4
Resultado: -1 -1 -1 1

TENTATIVA 10

Informe o 1º dígito da senha: 1
Informe o 2º dígito da senha: 2
Informe o 3º dígito da senha: 3
Informe o 4º dígito da senha: 4

Sua tentativa: 1 2 3 4
Resultado: -1 -1 -1 1

GAME OVER!

Senha correta: 9 5 8 4

10 tentativas!\n\n");

```
77 if(tentativa[cont] > 9 || tentativa[cont] < 0){
78     printf("Valor Invalido! ");
79 }else if (verifica_repetidos(tentativa, cont, tentativa[cont]) == 1){
80     printf("Valor Repetido! ");
81 }
```

Logs & others

C/C++ Windows (CR+LF) WINDOWS-1252 Line 70, Col 20, Pos 1972 Insert Read/Write default

POR 14:44