

TRABALHO

Projeto 1 & 2

Artur José Gomes Pereira

Nº 2040415

João José da Costa Cabral

Nº 2020919

**Tecnologias e Programação de Sistemas de
Informação**

UNIDADE CURRICULAR:

Programação Orientada a Objetos

DOCENTE:

Nélio Gaspar

ESCOLA SUPERIOR DE TECNOLOGIAS E GESTÃO

Cofinanciado por:

Índice

.....	0
Introdução.....	2
Análise do exercício 1.....	3
Exercício 1 Programa.....	4
Análise do exercício 2.....	9
Exercício 2 Programa.....	10
Conclusão.....	18
Web grafia.....	19

Introdução

Este relatório é referente ao trabalho solicitado pelo professor Luís Gaspar. Este trabalho consistia da realização de 2 programas ambos desenvolvidos em C++ com o intuito de ser avaliado na parte pratica da disciplina de Programação Orientada a Objetos.

O primeiro programa era um emulador do sistema de uma máquina de pagamento de um estacionamento com as condições de que a máquina só aceitaria notas como forma de pagamento e só devolveria moedas como troco.

O segundo programa era um sistema para simular o Euromilhões. Este programa questionava se o utilizador pretendia efetuar uma aposta simples ou múltipla e ainda tinha a possibilidade de efetuar uma aposta gerada de forma aleatória pelo sistema.

Este trabalho como um todo tinha como objetivos utilizar conhecimentos adquiridos em sala de aula, aumentar a nossa capacidade de análise de problemas e demonstrar o uso corretos de estruturas de dados.

Análise do exercício 1

O problema apresentado no exercício 1 consiste em uma máquina de pagamento de estacionamento, na qual um valor entre 0,10€ e 10€ é gerado aleatoriamente para assumir a quantidade a pagar numa máquina que só recebe notas, e para devolver o troco ao utilizador, somente em moedas.

De forma a resolver o problema nós criamos um programa que indica aleatoriamente o valor a pagar pelo utilizador, após o utilizador efetuar o pagamento do valor apresentado, o nosso sistema irá validar se o valor introduzido está entre o intervalo permitido, após validação se o valor introduzido for permitido pelo sistema o programa continua, no caso de o valor introduzido estiver fora do intervalo apresentado o mesmo indica uma mensagem para introduzir um valor correto, a seguir à inserção de um valor correto, o sistema verifica se é necessário devolver o troco, caso seja necessário o sistema calcula o valor do troco e entrega ao utilizador e depois termina, se não for necessário calcular troco o programa termina.

MÁQUINA DE PAGAMENTO DE UM ESTACIONAMENTO

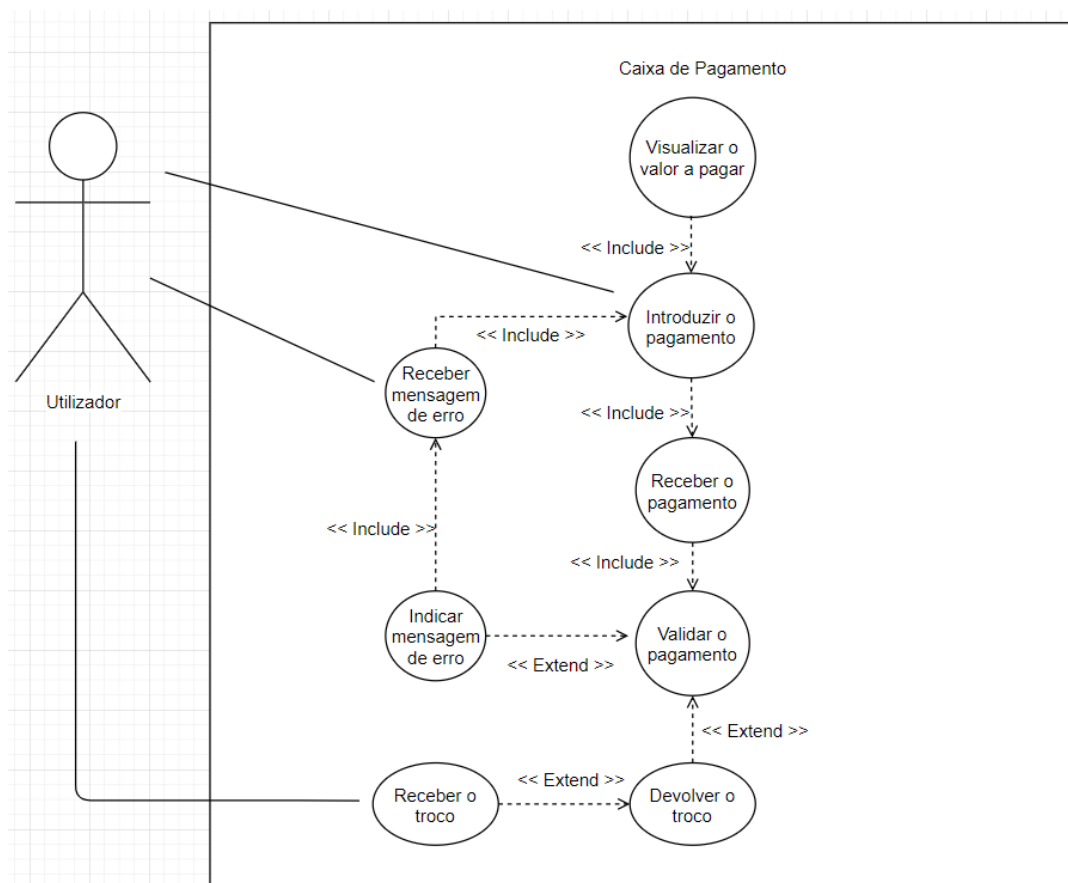


Figura 1

Exercício 1 Programa

Ao iniciarmos o programa o mesmo têm a seguinte aparência:

```
=====
                        Caixa de Pagamento
                        Apenas sao Aceites Notas
                        {5 , 10 , 20 , 50 , 100 , 200 , 500}
Valor a Pagar: 3.36
=====
Insira valor para o pagamento: ->
```

Figura 2

Utilizamos um método de alinhamento através da quantidade de caracteres existente numa linha cmd no tamanho padrão:

```
void print( Position pos, string text, int linelength ){
    int spaces = 0;
    switch( pos )
    {
        case CENTRE: spaces = ( linelength - text.size() ) / 2; break;
        case RIGHT : spaces =  linelength - text.size()      ; break;
    }
    if ( spaces > 0 ) cout << string( spaces, ' ' );
    cout << text << '\n';
}
```

Figura 3

Também utilizamos o texto introduzido de forma a formatar e a centrar a informação.

O nosso programa gera automaticamente o valor a pagar como pode ser verificado na primeira foto utilizando o código que pode verificar em baixo:

```

void menu(){
    const int LINELENGTH = 118;
    string header( LINELENGTH, '=' );

    double value;

    for (amountToPay = 0; amountToPay < 10;){
        srand(time(NULL));
        amountToPay = double(rand() % 1000);
        value = amountToPay / 100;
    }

    cout << header << "\n";

    print(CENTRE, "Caixa de Pagamento", LINELENGTH);
    cout << "\n";

    print(CENTRE, "Apenas sao Aceites Notas", LINELENGTH);
    print(CENTRE, "{5 , 10 , 20 , 50 , 100 , 200 , 500}", LINELENGTH);
    cout << "\n";

    cout << "Valor a Pagar: " << value << "\n";

    cout << header << '\n';
}

```

Figura 4

Após o sistema criar um valor de pagamento apresenta esse valor e depois aguarda pela introdução de um valor de pagamento pelo utilizador:

```

void inputData(){
    for (int i = 0; i < 1;) {
        cout << "Insira valor para o pagamento: ->";

        while(!(cin >> payment)) {
            cout << "Valor nao aceitavel. \nInsira valor para o pagamento: ->";
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
        }

        if (verify(payment) == 0){
            cout << "Valor incorreto." << "\n";
        } else if (verify(payment) == 1){
            cout << "Valor insuficiente." << "\n";
        } else {
            i += 1;
        }
    }
}

```

Figura 5

O nosso código nesta função pede o pagamento e recebe o valor do mesmo.

Após o utilizador introduzir o pagamento confirma se é um pagamento que é valido para introdução, caso não seja indica um erro e no caso de ser um valor insuficiente também refere o mesmo.

Validação do valor introduzido:

```
int verify(double number) {  
    double confirmValue[7] = {5,10,20,50,100,200,500};  
    int exists = false;  
  
    for (int i = 0; i < 7; i++) {  
        if(confirmValue[i] == number) {  
            if ((number * 100) < amountToPay) {  
                exists = 1;  
                return exists;  
                break;  
            } else {  
                exists = 2;  
                return exists;  
                break;  
            }  
        }  
    }  
    exists = 0;  
    return exists;  
}
```

Figura 6

Valor introduzido não é valido:

```
Insira valor para o pagamento: ->asasasaa  
Valor nao aceitavel.
```

Figura 7

Valor introduzido é insuficiente:

```
Valor a Pagar: 8.64  
=====
```

```
Insira valor para o pagamento: ->5  
Valor insuficiente.  
Insira valor para o pagamento: ->
```

Figura 8

Após a inserção do valor correto o nosso código verificar se é necessário entregar troco na seguinte função:

```
void getChange(){
    int paymentInCents = payment * 100;
    change = paymentInCents - amountToPay;
    cout << "O seu troco e de : ->" << change / 100;

    int cents[8] = {200 , 100 , 50 , 20 , 10 , 5 , 2 , 1};
    int rest = 0;
    int amount = 0;

    cout << "\n";

    for (int i = 0; i < 8 ; i++){
        double now = cents[i];
        amount = change / now;
        rest = change - (amount * now);
        change = rest;

        if (amount > 0){
            if (amount == 1) {
                switch (i){
                    case 0:
                        cout << amount << " moeda de " << (cents[i] / 100) << " euros" << "\n";
                        break;
                    case 1:
                        cout << amount << " moeda de " << (cents[i] / 100) << " euro" << "\n";
                        break;
                    case 7:
                        cout << amount << " moeda de " << cents[i] << " centimo" << "\n";
                        break;
                    default:
                        cout << amount << " moeda de " << cents[i] << " centimos" << "\n";
                        break;
                }
            } else {
                switch (i){
                    case 0:
                        cout << amount << " moedas de " << (cents[i] / 100) << " euros" << "\n";
                        break;
                    case 1:
                        cout << amount << " moedas de " << (cents[i] / 100) << " euro" << "\n";
                        break;
                    case 7:
                        cout << amount << " moedas de " << cents[i] << " centimo" << "\n";
                        break;
                    default:
                        cout << amount << " moedas de " << cents[i] << " centimos" << "\n";
                        break;
                }
            }
        }
    }
}
```

Figura 9

Caso seja necessário o programa entrega o troco se não for necessário o programa termina.

Após a introdução da matéria das classes durante as nossas aulas em vez de termos as funções simplesmente no programa decidimos criar uma classe de forma a organizar melhor o nosso código.


```

class moneyFunctions {
private:
    int verify(double number) {

public:
    void inputData() {
    void getChange() {
};

```

Figura 10

Na imagem a cima temos a forma como organizamos as funções em que colocamos a função de verificar os valores introduzidos em privado e as restantes em publico.

Uma vez que está tudo organizado como pretendíamos a nossa função Main apenas chama as funções da classe e as funções do menu.

```

main() {
    menu();
    cout << "\n";

    moneyFunctions park;
    park.inputData();
    park.getChange();

    cout << "Press anykey to exit.";
    cin.ignore();
    cin.get();
}

```

Figura 11

Análise do exercício 2

O problema que nos foi apresentado no exercício 2 consiste na criação de um programa que simula o Euromilhões em que o sistema questiona ao utilizador qual o tipo de aposta que o mesmo pretende realizar.

De forma a resolver o problema criamos um menu que apresenta opções ao utilizador em que o mesmo deve seleccionar o que pretende.

O utilizador indica se pretende realizar uma aposta simples ou múltipla e ainda têm a possibilidade de efetuar uma aposta de forma aleatória, no boletim de aposta pode também realizar de 1 a 5 chaves de aposta com os números de 1 a 50 e as estrelas de 1 a 9.

Após introduzir a informação no programa o mesmo apresenta a informação ao utilizador.

PROGRAMA DO EUROMILHOES

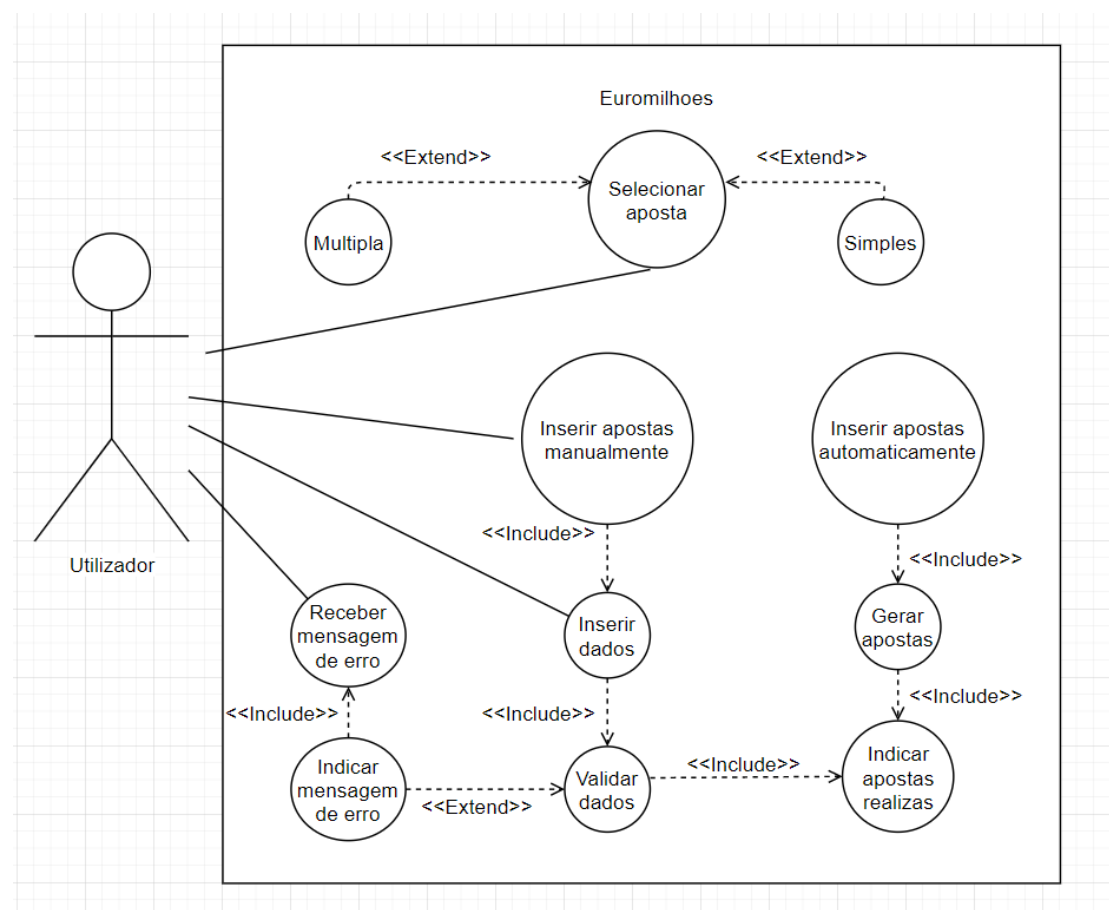


Figura 12

Exercício 2 Programa

Neste exercício decidimos aproveitar algum do código que já criado anteriormente, nomeadamente a criação do menu, utilizamos o mesmo código com alguns reajustes de forma a enquadrar no pretendido em que o resultado final foi este:

```
=====
                        Euromilhoes
                Seleccione o tipo de aposta a realizar:

                        Aposta Simples
                          (1)

                        Aposta Multipla
                          (2 a 5)
=====
Quantidade: ->
```

Figura 13

Após a inicialização do programa temos 2 opções uma vez que existem até ao momento dois tipos de aposta a Simples e a Múltipla.

A função abaixo demonstra as opções que o cliente têm:

```
int askInsertType() // Pergunta ao utilizador se os bilhetes irao ser inseridos de forma automatica ou manual
{
    char type;
    system("cls");

    do
    {
        if (cont > 0)
        {
            cout << "Valor introduzido incorreto!\n";
        }
        cout << header << "\n";

        print(CENTRE, "Euromilhoes", LINELENGTH);
        cout << "\n";

        print(CENTRE, "M -> Inserir Manualmente", LINELENGTH);
        print(CENTRE, "A -> Inserir Automaticamente", LINELENGTH);
        cout << header << "\n";

        cout << "Opcao: ->";
        cin >> type;

        if ((type == 'a') || (type == 'A') || (type == 'm') || (type == 'M'))
        {
            invalid = false;
            if ((type == 'm') || (type == 'M'))
            {
                return 0;
            }
            else
            {
                return 1;
            }
        }
        else
        {
            invalid = true;
            cont += 1;
        }
    } while (invalid);
}
```

Figura 14

Ao selecionar a opção que pretende é apresentado o seguinte menu:

```
Euromilhoes
M -> Inserir Manualmente
A -> Inserir Automaticamente
=====
Opcao: ->
```

Figura 15

Em que o cliente ao escolher a opção pode selecionar jogar por ele mesmo efetuando a escolha dos números ou pela maquina.

Criamos uma função em que pergunta ao utilizador por 5 números e 2 estrelas e depois define esses valores para um determinado objeto:

Números:

```
void newTicket()
{
    system("cls");

    cout << header << "\n";
    print(CENTRE, "Euromilhoes", LINELENGTH);
    cout << "\n";
    print(CENTRE, "Numeros: ", LINELENGTH);
    cout << header << "\n";
    for (int i = 0; i < 5; i++)
    {
        int userInput = true;
        while (userInput)
        {
            cout << "Number " << i + 1 << ": ";
            cin >> this->Numbers[i];
            if (cin.fail() || this->Numbers[i] <= 0 || this->Numbers[i] > 50 || this->NumberExists(this->Numbers[i], i) == true)
            {
                cin.clear();
                cin.ignore();
                cout << "Essa opcao nao valida"
                    << "\n";
            }
            else
            {
                userInput = false;
            }
        }
    }
}
```

Figura 16

Estrelas:

```
system("cls");

cout << header << "\n";
print(CENTRE, "Euromilhoes", LINELENGTH);
cout << "\n";
print(CENTRE, "Estrelas: ", LINELENGTH);
cout << header << "\n";
for (int i = 0; i < 2; i++)
{
    int userInput = true;
    while (userInput)
    {
        cout << "Star " << i + 1 << ": ";
        cin >> this->Stars[i];
        if (cin.fail() || this->Stars[i] <= 0 || this->Stars[i] > 12 || this->StarExists(this->Stars[i], i))
        {
            cin.clear();
            cin.ignore();
            cout << "Essa opcao nao valida"
                  << "\n";
        }
        else
        {
            userInput = false;
        }
    }
}
```

Figura 17

Selecionando a manual, o cliente tem de inserir 5 números e 2 estrelas de forma a validar os números e estrelas introduzidos nos criamos duas funções dentro da classe Ticket que verifica se o ultimo numero inserido já existe dentro do array pretendido e se já existir envia um erro ao cliente.

```
class Ticket
{
private:
    bool NumberExists(int number, int currentPositon)
    {
        for (int i = 0; i < currentPositon; i++)
        {
            if (this->Numbers[i] == number)
            {
                return true;
            }
        }

        return false;
    }

    bool StarExists(int number, int currentPositon)
    {
        for (int i = 0; i < currentPositon; i++)
        {
            if (this->Stars[i] == number)
            {
                return true;
            }
        }

        return false;
    }

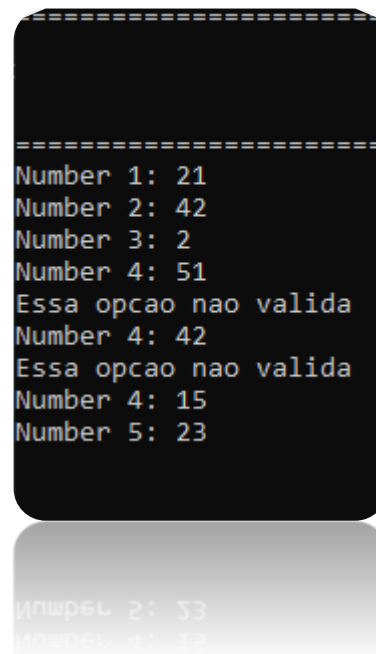
public:
    int Numbers[5];
    int Stars[2];
    int *NumbersPtr[5];
    int *StarsPtr[2];

    Ticket()
    {
        // ...
    }
};
```

Figura 18

No caso de algum numero esteja fora do intervalo (números até 50 e estrelas até 12) é apresentada a seguinte mensagem de erro como podem verificar no código da função newTicket() em cima demonstrada.

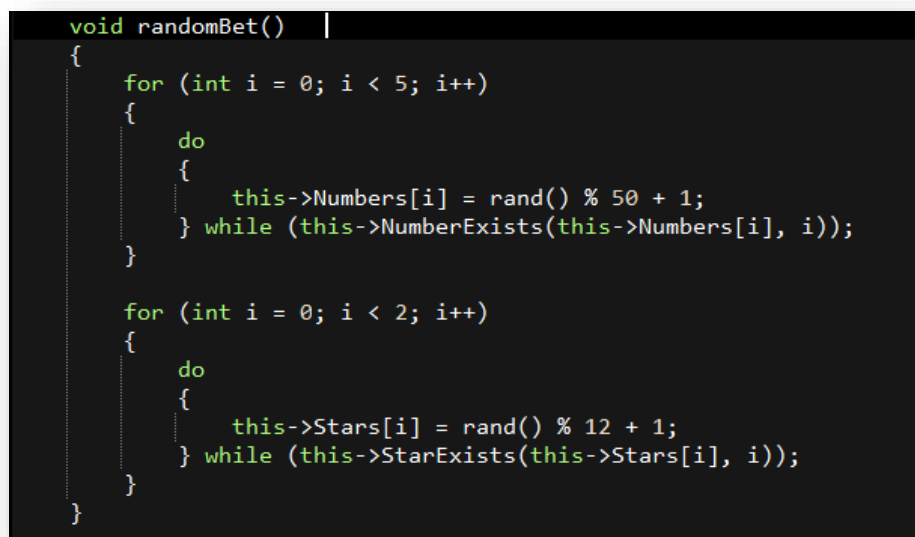
Demonstração de Output com erro:



```
=====
Number 1: 21
Number 2: 42
Number 3: 2
Number 4: 51
Essa opcao nao valida
Number 4: 42
Essa opcao nao valida
Number 4: 15
Number 5: 23
```

Figura 19

Uma vês que no Euromilhões existe também a possibilidade de jogar pela maquina criamos uma função para que faça uma ou mais que uma aposta aleatória:



```
void randomBet()
{
    for (int i = 0; i < 5; i++)
    {
        do
        {
            this->Numbers[i] = rand() % 50 + 1;
        } while (this->NumberExists(this->Numbers[i], i));
    }

    for (int i = 0; i < 2; i++)
    {
        do
        {
            this->Stars[i] = rand() % 12 + 1;
        } while (this->StarExists(this->Stars[i], i));
    }
}
```

Figura 20

Na aposta Múltipla o raciocínio anterior é o mesmo no entanto foi criada uma nova função chamada `amountOfKeys()`, que valida se o número de chaves introduzidas está dentro do intervalo e se estiver dentro do intervalo deixa o programa avançar:

```
void amountOfKeys() // Pergunta ao utilizador a quantidade de bilhetes deseja inserir
{
    int i;
    quantity = 0;
    do
    {
        system("cls");
        cout << header;

        print(CENTRE, "Euromilhoes", LINELENGTH);
        cout << "\n";

        print(CENTRE, "Selecione o tipo de aposta a realizar:\n", LINELENGTH);
        print(CENTRE, "Aposta Simples ", LINELENGTH);
        print(CENTRE, "(1) \n", LINELENGTH);
        print(CENTRE, "Aposta Multipla", LINELENGTH);
        print(CENTRE, "(2 a 5)\n", LINELENGTH);

        cout << header;
        if (cont == 1)
        {
            cout << "Ultrapassaria o limite de chaves\n";
        }
        else if (cont == 2)
        {
            cout << "Valor inserido nao aceite\n";
        }

        cout << "Quantidade: ->";

        while (!(cin >> quantity))
        {
            cout << "Valor introduzido incorreto!\nQuantidade: ->";
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            i += 1;
        }
    }
}
```

Parte 1

```

        if (i > 4)
        {
            i = 0;
            system("cls");
            cout << header;

            print(CENTRE, "Euromilhoes", LINELENGTH);
            cout << "\n";

            print(CENTRE, "Chaves a inserir", LINELENGTH);
            print(CENTRE, "Maximo 5", LINELENGTH);

            cout << header;
            cout << "Valor introduzido incorreto!\nQuantidade: ->";
        }
    }

    if (quantity + amount > 5)
    {
        cont = 1;
    }
    else if (quantity > 5 || quantity < 1)
    {
        cont = 2;
    }

} while (quantity > 5 || quantity < 1 || quantity + amount > 5);

amount = amount + quantity;
}

```

Parte 2

Exemplo de output ao introduzir 10:

```
Ultrapassaria o limite de chaves  
Quantidade: ->
```

Figura 21

A baixo podem verificar também a função que criamos de forma a que os bilhetes depois sejam transmitidos ao cliente através do menu:

```
void printTicket()
{
    cout << "\nNumbers: ";
    for (int i = 0; i < 5; i++)
    {
        cout << this->Numbers[i] << " ";
    }
    cout << "\n";
    cout << "Stars: ";
    for (int i = 0; i < 2; i++)
    {
        cout << this->Stars[i] << " ";
    }
}
};
```

Figura 22

E a nossa ultima função é um simples menu que mostra ao utilizador após a seleção de opções:

```
void consultMenu() // Menu simples para o final do programa
{
    int opp = 0, i;

    cout << header;

    print(CENTRE, "Euromilhoes", LINELENGTH);
    cout << "\n";

    print(CENTRE, "Consultar Chave", LINELENGTH);

};

};
```

Figura 23

No main temos a inicialização e parte do "corpo" do nosso programa em que chamada as nossa funções:

```
Ticket bets[5]; // cria array de objetos
Ticket winnerTicket; // cria a chave vencedora
winnerTicket.randomBet(); // define os valores da chave
int opt, inputError;

srand(time(NULL));

amountOfKeys(); // bilhetes a inserir

if (askInsertType() == 0) // askInsertType == 0 quer dizer inserir os valores a mão / askInsertType == 1 quer dizer inserir os valores automaticamente
{
    for (int i = 0; i < quantity; i++)
    {
        bets[i].newTicket(); // define os valores na determinada posicao do array
    }
    if (quantity > 1)
    {
        system("cls");
        print(CENTRE, "Chaves inseridas", LINELENGTH);
    }
    else
    {
        system("cls");
        print(CENTRE, "Chave inserida", LINELENGTH);
    }
}
else
{
    for (int i = 0; i < quantity; i++)
    {
        bets[i].randomBet(); // define os valores na determinada posicao do array
    }
    if (quantity > 1)
    {
        system("cls");
        print(CENTRE, "Chaves inseridas", LINELENGTH);
    }
    else
    {
        system("cls");
        print(CENTRE, "Chave inserida", LINELENGTH);
    }
}
```

Figura 24

```

consultMenu();
cout << "Chave vencedora: ";
winnerTicket.printTicket();
cout << "\n";
cout << header;

for (int x = 0; x < amount; x++) // estes for's vao passar por todas as posicoes dos arrays e compara quantos numeros iguais existe entre a chave vencedora e as inseridas
{
    soma = 0;
    for (int i = 0; i < 5; i++)
    {
        for (int y = 0; y < 5; y++)
        {
            if (winnerTicket.Numbers[i] == bets[x].Numbers[y])
            {
                soma += 1;
            }
        }
    }
    for (int i = 0; i < 2; i++)
    {
        for (int y = 0; y < 2; y++)
        {
            if (winnerTicket.Stars[i] == bets[x].Stars[y])
            {
                soma += 1;
            }
        }
    }
    switch (soma)
    {
        case 7: // Primeiro premio == todos os numeros e estrelas iguais
            cout << "\n\nBilhete " << x + 1 << ": PRIMEIRO PREMIO";
            bets[x].printTicket();
            break;
        case 6:
            cout << "\n\nBilhete " << x + 1 << ": PREMIO MUITO ALTO";
            bets[x].printTicket();
            break;
        case 5:
            cout << "\n\nBilhete " << x + 1 << ": PREMIO MEDIO ALTO";
            bets[x].printTicket();
            break;
        case 4:
            cout << "\n\nBilhete " << x + 1 << ": PREMIO ALTO";
            bets[x].printTicket();
            break;
        case 3:
            cout << "\n\nBilhete " << x + 1 << ": PREMIO MEDIO";
            bets[x].printTicket();
            break;
        case 2:
            cout << "\n\nBilhete " << x + 1 << ": PREMIO BAIXO";
            bets[x].printTicket();
            break;
        default:
            cout << "\n\nBilhete " << x + 1 << ": SEM PREMIO";
            bets[x].printTicket();
            break;
    }
}
}

```

Figura 25

E aqui podem verificar as nossas ultimas linhas de código em que mostra o nosso menu final e em que verifica se o cliente tem premio com a chave introduzida.

Output do resultado final em que foi seleccionado 5 apostas múltiplas de forma automática:

```

Euromilhoes
Consultar Chave

Chave vencedora:
Numbers: 42 18 35 1 20
Stars: 5 7
=====

Bilhete 1: PREMIO BAIXO
Numbers: 4 35 1 13 26
Stars: 3 11

Bilhete 2: PREMIO BAIXO
Numbers: 8 19 42 47 1
Stars: 6 8

Bilhete 3: SEM PREMIO
Numbers: 17 28 1 29 26
Stars: 4 3

Bilhete 4: SEM PREMIO
Numbers: 2 35 23 25 9
Stars: 8 1

Bilhete 5: SEM PREMIO
Numbers: 10 36 8 49 46
Stars: 5 2
=====

```

Figura 26

Conclusão

Após concluirmos estes dois programas, chegamos a conclusão que, de forma a ser eficiente, é necessária organização e uma definição de objetivos a atingir.

Para poder trabalhar em equipa é necessária comunicação entre os dois programadores e partilha de informação entre ambos, apesar de algumas dificuldades iniciais em definir como íamos realizar os programas e após alguma pesquisa sobre o que queríamos realizar conseguimos realizar ambos os programas solicitados.

Finalizamos este relatório com a certeza de ter aumentado os nossos conhecimentos sobre C++, que com certeza será útil durante toda a nossa futura jornada como programadores, temos a agradecer ao professor Nélío Gaspar por todos os conhecimentos que nos foram transmitidos durante este semestre.

Web grafia

https://www.w3schools.com/cpp/cpp_constructors.asp

<https://www.geeksforgeeks.org/constructors-c/>

<http://www.cplusplus.com/doc/tutorial/arrays/>

https://www.w3schools.com/cpp/cpp_arrays.asp

<https://stackoverflow.com/questions/21489160/c-putting-objects-into-arrays>

<http://www.cplusplus.com/forum/general/30450/>

<http://www.cplusplus.com/reference/cstdlib/rand/>

<http://www.cplusplus.com/reference/random/>

<http://www.cplusplus.com/reference/vector/vector/>

<http://www.cplusplus.com/doc/tutorial/operators/>

<http://www.cplusplus.com/doc/tutorial/arrays/>

<http://www.cplusplus.com/doc/tutorial/classes/>

<http://www.cplusplus.com/doc/tutorial/inheritance/>

<https://stackoverflow.com/questions/6219878/stack-overflow-c>

<https://pt.stackoverflow.com/questions/tagged/c%2B%2B>