```
1
2
3   int matriz [3] [4];
4   int l,c;
5
6   //Insere Valores
7
8   for(l=0;l<3;l++){
9           for (c=0;c<4;c++){
10          cin >> matriz [l] [c];   //matriz [l] [c] =0; posso colocar um valor fixo
11
12          }
13
14  }
15
16  //Imprime valores
17  for(l=0;l<3;l++){
18          for (c=0;c<4;c++){
19          cout << matriz [l] [c] << " ";
20
21          }
22          cout << "\n";
23  }
24
25
26  //Passagem de Parametros
27
28  int main (int argc, char *argv[]){ // int argc armazena a quantidade de argumentos,
    e o argv armazena os argumentos
29
30      cout << argv [0] << "\n\n";  //por padrão o primerio parametro é o nome do
        programa
31
32      if (argc > 1){
33          if (!strcmp (argv [1], "sol")){
34              count << "Vou ao clube.\n\n";
35          }
36
37
38
39
40      }
41
42      return 0;
43  }
44
45  //Bibliotecas de string.h
46
47  strcmp(arqc[1], "sol")
48
49  /*
50  *
51  *
52  *
53  *
54  */
55
56  //FUNÇÕES
57  //tipo de retorno de uma função, nome, argumentos
58
59
60  #include <iostream>
61
62  using namespace std;
63
64  void texto();//No C++ precisa prototipar apresentando a função  para que o C conheça
    a função Texto
65  void soma (int n1, int n2);
66  int  soma2 (int n1, int n2):
67  void tr (string tra[4]);
68
69  int main () {
70
```

```
71      int res;
72      string transp[4] ={"carro", "moto", "barco", "aviao"};
73
74
75      texto(); // Chamando a função
76      soma(15,5);
77
78      //passando valores para utilizar a funcao soma2 e armazenando o resultado em res
79      res=soma2(175,25);
80
81
82      tr(transp);
83
84
85
86      return o;
87  }
88
89  void texto (){
90
91      cout << "\n Teste de função";
92
93
94  }
95
96  //argumentos de entrada
97
98  void soma (int n1, int n2){
99
100         cout <<  "soma dos valores" << n1+n2 << "\n";
101 }
102
103 //somando e retornando um valor, qualquer valor diferente de void vamos usar o return
104 int  soma2 (int n1, int n2){
105
106
107         return n1+n2;
108 }
109
110 //passar para a função o vetor
111 void tr(string tra[4]){
112         for(int i=0; i<4; i++){
113
114             cout << tra[i] << "\n";
115         }
116
117
118 }
119
120
121 //omitir argumento
122 #include <iostream>
123
124 using namespace std;
125
126 //podemos fazer sobrecarga de funções declarando funções com nomes iguais mas seus
    argumentos são diferentes
127
128
129 //Atribuindo um valor para o argumento consigo omitir
130 void imp (string txt="");
131
132 int main () {
133
134     imp();
135
136
137     return o;
138 }
139
140 void imp (string txt){
141     cout << "\n" << txt << "\n";
142
```

```
143
144 }
145
146
147 //####ENUM###////
148
149 //Valores inteiros
150 //Por padrão cada item tem um valor, ele sempre tem uma estrutura
    sequencial...primeiro item é 0
151 //Ele é sequencial apartir do numero anterior
152
153 #include <iostream>
154
155
156 using namespace std;
157
158 int main (){
159
160     enum armas {fuzil=100, revolver=8, rifle=12, escopeta=1};
161
162     //tipo   variavel
163     armas armaSel;
164
165     armaSel = fuzil;
166
167     cout << armaSel;
168
169     return 0;
170 }
171
172
173 //// ################# STRUCT ##############///////
174
175 //Coleção de variáveis e métodos
176
177
178 #include <iostream>
179
180
181 using namespace std;
182
183
184 struct Carro{
185
186         string nome;
187         string cor;
188         int pot;
189         int valMax;
190
191 };
192
193 int main (){
194
195      //Declarar essa variável
196
197      Carro car1; //Declarando uma variável do tipo Carro
198
199      car1.nome="Tornado";
200      car1.cor="Vermelho";
201      car1.pot=450;
202      car1.velMax=350;
203
204      cout << "Nome do carro: "<<car1.nome << "\n";
205
206     return 0;
207 }
208
209
210 //Trabalhando com funções e Métodos
211
212 #include <iostream>
213
214
```

```
215  using namespace std;
216
217
218  struct Carro{
219
220          string nome;
221          string cor;
222          int pot;
223          int valMax;
224          int vel;
225
226          void  insere (string stnome, string stcor, int stpot, int stvelmax){
227
228              nome=stnome;
229              cor=stcor;
230              pot=stpot;
231              velMax=stvelmax;
232              vel=0;
233
234          }
235
236          void mostra (){
237                  cout << "Nome.................. :" << nome <<"\n";
238                  cout << "Cor.................. :" << cor<<"\n";
239                  cout << "Potencia.................. :" << pot <<"\n";
240                  cout << "Velocidade.................. :" << vel <<"\n";
241                  cout << "Velocidade Máxima.................. :" << velMax <<"\n";
242
243          }
244
245          //Mudar a velocidade
246          //verifica se a velocidade não é maior que velocidade máxima
247          void mudaVel (int mv){
248              vel=mv;
249
250              if(vel > velMax){
251
252                  vel=velMax;
253              }
254              if(vel <0){
255
256                  vel=0;
257              }
258
259
260          }
261
262  };
263
264
265  int main (){
266
267       //Declarar essa variável
268
269       Carro car1,car2; //Declarando uma variável do tipo Carro
270
271
272       car1.insere("Tornado", "Vermelho",450,350);
273       car2.insere("Tornado", "Vermelho",450,350);
274
275       car1.mostra();
276
277       car1.mudaVel(150);
278       car1.mostra();
279
280      return 0;
281  }
282
283
284  ///############Continuando STRUCT CRIANDO UM ARRAY
285
286
287  #include <iostream>
```

```
288
289
290   using namespace std;
291
292
293   struct Carro{
294
295           string nome;
296           string cor;
297           int pot;
298           int valMax;
299           int vel;
300
301           void  insere (string stnome, string stcor, int stpot, int stvelmax){
302
303               nome=stnome;
304               cor=stcor;
305               pot=stpot;
306               velMax=stvelmax;
307               vel=0;
308
309           }
310
311           void mostra (){
312                   cout << "Nome.................. :" << nome <<"\n";
313                   cout << "Cor.................. :" << cor<<"\n";
314                   cout << "Potencia.................. :" << pot <<"\n";
315                   cout << "Velocidade.................. :" << vel <<"\n";
316                   cout << "Velocidade Máxima.................. :" << velMax <<"\n";
317
318           }
319
320           //Mudar a velocidade
321           //verifica se a velocidade não é maior que velocidade máxima
322           void mudaVel (int mv){
323               vel=mv;
324
325               if(vel > velMax){
326
327                   vel=velMax;
328               }
329               if(vel <0){
330
331                   vel=0;
332               }
333
334
335           }
336
337   };
338
339
340   int main (){
341
342       //um Objeto do tipo Carro cria um Array Carro de 5 posições
343
344        Carro *carros=new Carro[5];
345        Carro car1,car2, car3, car4, car5; //Declarando uma variável do tipo Carro
346
347        //cada carro vai ter uma posição
348
349        carros [0] =car1; carros [1] =car2; carros [2] =car3; carros [3] =car4; carros
         [4] =car5;
350
351        carros [0].insere("Tornado", "Vermelho",450,350);
352        carros [1].insere("Tornado", "Vermelho",450,350);
353        carros [2].insere("Tornado", "Vermelho",450,350);
354        carros [3].insere("Tornado", "Vermelho",450,350);
355        carros [4].insere("Tornado", "Vermelho",450,350);
356
357
358        for(int i=0; i<5; i++){
359               carros[i].mostra();
```

```
360
361         }
362
363
364         return 0;
365     }
366
367
368     //###############PONTEIROS
369
370
371     // Não armazena o endereço de uma variavel, mas o endereço de outra variável, ele
        está apontando.
372     //Ponteiro tem que ser da mesma variável
373
374     /*
375     End                     Tipo                    valor           nome
376     1000                        int                 4                   num
377
378
379     int *pn;          //criando  ponteiro
380     pn=&num;      //Associei o endereço num com o ponteiro
381     cout << pn;    //Endereço;
382     cout << *pn;
383
384
385     */
386
387     int main (){
388
389     string veiculo="Carro";
390     string *pv; // criou o endereço;
391
392     pv=&veiculo; // Ponteiro PV recebe o endereço da variavel veículo
393
394     cout << pv; //Endereço da memória Ram;
395     cout << &veiculo;
396
397     *pv="Moto";
398
399
400     }
401
402
403     // Manipulando Arrays e vetores
404
405
406     int *p;
407     int vet[10];
408
409     p=vet; //atribuindo o primeiro elemento do vetor p=&vetor[0];
410
411     p=&vetor[1];
412     cout<< "\n" << p << "\n";
413
414     *(p+=1);//incrementando 1 para o próxima posição
415     *p=10; // 10 atribuído a primeira posição do vetor
416     cout << "\n" << vetor[0]<<"\n";
417
418     *(p+=1); //incrementando para a segunda posição
419     *p=20;
420     cout << "\n" << vetor[1] << "\n";
421
422
423
424     void iniVetor (float *v);
425
426     int main(){
427
428     float vetor[5];
429     iniVetor(vetor); // Não precisa especificar o endereço
430     }
431
```

```
432   for(int i=0; i<5; i++){
433
434       cout<< vetor[i] << "\n";
435   }
436
437   void iniVetor (float *v){
438       v[0]=0;
439       v[1]=0;
440       v[2]=0;
441       v[3]=0;
442       v[4]=0;
443   }
444
445
446   ////////////////////////////////Alocação dinamica
447
448
449   #include <iostream>
450   #include <stdio.h> // Para função gets
451   #include <stdlib.h> //Para função malloc
452
453
454   //Malloc de memóry alocation, alocar exatamente o que é necessário para reservar um
      espaço de memória
455
456   int main (){
457
458           char *vnome;
459           vnome=(char*)malloc(sizeof(char)) //indicar o tamanho do bloco, retorna
              void, porém a variável nome é char, tapecast para converter o retorno/
              sizeof retorna o tamanho do elemento (char) 1 byte
460           vnome=(char*)malloc(sizeof(char)+1)//Soma a alocação de memória
461
462
463           gets(vnome);
464
465
466
467
468
469       return 0;
470
471
472
473   }
474
475   //////////////Recursividade
476   #include <iostream>
477
478   using namespace std;
479
480   int main (){
481
482           contador(20);
483
484
485
486           return 0;
487
488
489   }
490
491   void contador(int num, int cont){
492       cout << cont << "\n";
493       if(num > cont){
494           contador(num, ++cont);
495
496       }
497
498   }
499
500
501   ////////////////////////Recursividade Fatorial
```

```cpp

#include <iostream>

using namespace std;

int main (){

        int val,res;

        val=8;
        res=fatorial(val);
        cout << "Fatorial de " << val ": " << res;

        cout << "\n\nFibonacci com " << val <<" valores: ";
        for(int i=0; i<val;i++){
            cout << fibonacci (i+1) << " ";

        }

        cout << "\n\n";

        return 0;
}

int fatorial (int n) {
    if(n==0){

        return 1;
    }
    return n*fatorial(n-1);

}

int fibonacci(int n){
    if (n==1 || n==2){
        return 1;

    }else{
        return fibonacci(n-1)+fibonacci(n-2);
    }

}


////////////Pilha /Stack - Um tubo, o primeiro elemento inserido é o ultimo elemento a ser trabalhado, o ultimo elemento inserido é o primeiro a ser trabalhado

/*

            |  |
            |_|

*/


#include <iostream>
#include <stack>

using namespace std;

int main (){

        stack <string> cartas;

        cartas.push("Rei de copas"); // inserir elemento na pilha
        cartas.push("Rei de Espadas");
        cartas.push("Rei de Ouros");
        cartas.push("Rei de Paus");

        cout << "Tamanho da pilha:" << cartas.size() << "\n";

        cout<<"Carta do topo: " << cartas.top() << "\n";

```

```
574             cartas.pop(); // retira o elemento da pilha que está no topo
575
576             cout << "Tamanho da pilha:" << cartas.size() << "\n";
577
578             cout<<"Nova carta do Carta do topo: " << cartas.top() << "\n"; // Vert carta
                do topo
579
580
581             return 0;
582     }
583
584
585     //////Pilhas Método Empt
586
587     #include <iostream>
588     #include <stack>
589
590     using namespace std;
591
592     int main (){
593
594             stack <string> cartas;
595
596             if (cartas.empty()){ //empty retorna verdadeiro ou falso
597
598                 cout<< "Pilha vazia\n\n";
599
600             }else{
601
602                 cout << "Pilha com cartas\n\n";
603
604
605             }
606
607             //Pode ser usado o size também para verificar se existe elementos na pilha
608
609             if (cartas.size()==0){ //empty retorna verdadeiro ou falso
610
611                 cout<< "Pilha vazia\n\n";
612
613             }else{
614
615                 cout << "Pilha com cartas\n\n";
616
617
618             }
619
620             //excluir elementos
621             while(!cartas.empty()){ // Enquanto cartas não for vazio
622
623                 cartas.pop();
624
625             }
626
627             cartas.push("Rei de copas"); // inserir elemento na pilha
628             cartas.push("Rei de Espadas");
629             cartas.push("Rei de Ouros");
630             cartas.push("Rei de Paus");
631
632             cout << "Tamanho da pilha:" << cartas.size() << "\n";
633
634             cout<<"Carta do topo: " << cartas.top() << "\n";
635
636             cartas.pop(); // retira o elemento da pilha que está no topo
637
638             cout << "Tamanho da pilha:" << cartas.size() << "\n";
639
640             cout<<"Nova carta do Carta do topo: " << cartas.top() << "\n"; // Vert carta
                do topo
641
642
643             return 0;
644     }
```

```
645
646 ////////////////////////Filas****** /Queue
647 /*
648
649             _     _
650               |   |
651             _|    |_
652
653 */          //Primeiro elemento que entra é o primeiro elemento que sai, ultimo a entrar ultimo a sair
654 #include <iostream>
655 #include <queue>
656
657 using namespace std;
658
659 int main (){
660
661     /*
662         empty
663         size
664         front -  Na frente da fila
665         back -
666         push
667         pop
668     */
669
670     queue <string> cartas;
671
672         cartas.push("Rei de copas"); // inserir elemento na pilha
673         cartas.push("Rei de Espadas");
674         cartas.push("Rei de Ouros");
675         cartas.push("Rei de Paus");
676
677         cout << "Tamanho da fila:" << cartas.size() << "\n";
678         cout<< "Primeira Carta: " << cartas.front() << "\n";
679         cout<< "Primeira Carta: " << cartas.back() << "\n\n";
680
681
682
683         //excluir elementos
684         while(!cartas.empty()){ // Enquanto cartas não for vazio
685
686
687             cout<< "Primeira Carta: " << cartas.front() << "\n";
688             cartas.pop(); //sempre vai remover a carta que está na frente da fila
689         }
690
691
692         return 0;
693 }
694
695 /////////////////POO //// C++
696
697 #include <iostream>
698
699 using namespace std;
700
701
702 class Aviao{
703
704     public:
705         int vel=0;
706         int velMax;
707         string tipo;
708         void ini(int tp); //Prototipar a Função, criando um metodo
709
710     private:
711
712 };
713
714 void Aviao::ini(int tp){//tipo 1=jato, 2 = monomotor 3=Planador
715
716     if(tp==1){
```

```
717             this->velMax = 800; //Para dizer que velmax pertence a classe avião usamos
                "this" (está)
718             this->tipo = "Jato";
719
720         }else if (tp==2){
721             this->velMax=350;
722             this->tipo="Monomotor";
723
724         }
725     }
726
727     int main (){
728
729
730             Aviao *av1=new Aviao(); //No c++ precisa definir o ponteiro, instancio o
                objeto dessa classe
731
732             av1->ini(1);
733
734             cout << av1->vel; // O "."  nesse exemplo av1.vel mostra uma propriedade do
                Objeto, mas no C++ precisamos alterar a sintaxe, no c++ é "->"
735
736             return 0;
737
738     }
739
740     //*************************************************************************************
    ********************
741     ///////////Aula 45 - POO, Classe em arquivo externo - Arquivo principal.cpp
742
743
744
745     #include <iostream>
746     #include "arquivo.h"
747
748     using namespace std;
749
750
751         Aviao *av1=new Aviao(1);//Informa o tipo de avião, ou seja passando o parametro
752         Aviao *av2=new Aviao(3);
753         Aviao *av3=new Aviao(2);
754
755         av1->imprimir();
756         av2->imprimir();
757
758
759
760     //************************************************Segundo arquivo.h
761     //ou usar namespace std;
762     class Aviao{
763
764         public:
765             int vel=0;
766             int velMax;
767             std::string tipo;
768             Aviao(int tp); //Prototipar a Função, criando um metodo
769             void imprimir();
770
771         private:
772
773     };
774
775     Aviao::Aviao(int tp){//tipo 1=jato, 2 = monomotor 3=Planador
776
777         if(tp==1){
778             velMax = 800; //Aqui não usamos o usamos "this" por causa da declaração do
                nome Aviao(int tp);
779             tipo = "Jato";
780
781         }else if (tp==2){
782             velMax=350;
783             tipo="Monomotor";
784
```

```
785         }else if (tp==3){
786             velMax=180;
787             tipo="Planador";
788     }
789
790
791
792     void Aviao::imprimir(){
793
794         std::cout<< "Tipo: " << tipo <<endl;
795         std::cout<< "Velocidade máxima: " << velMax <<endl;
796
797     }
798
799     /////GET E SET*****************************************************#46
800     //TRABALHANDO COM PARAMETROS PRIVATE
801
802     //ARquivo principal
803
804         Veiculo *v1=new Veiculo();
805         //v1->velMax=300; // Está dando erro por que o velMax é privado
806         cout<<v1->getVelMax();
807         cout<<v1->velMax;
808
809         v1->setLigado(1);
810
811
812     //Arquivo .harderr
813
814     class Veiculo{
815
816         public:
817
818             int vel;
819             int tipo;
820             Veiculo(int tp);//Método construtor é o método que é chamado quando
                instancio um objeto da classe
821             int getVelMax(); //Mesmo tipo da variável
822             bool getLigado();
823             void setLigado(int l);
824
825         private:
826             std::string nome;
827             int velMax;
828             bool ligado;
829             void setVelMax(int vm);
830     };
831
832     bool Veiculo::getLigado(){
833         return ligado;
834     }
835
836     void Veiculo::setLigado(int l){
837         if(l==1){
838             ligado=true;
839         }else if(l==0){
840             ligado=false;
841         }
842     }
843
844     int Veiculo::getVelMax(){
845         return VelMax;
846     }
847
848     void Veiculo::setVelMax(int vm){
849         velMax=vm;
850     }
851
852     Veiculo::Veiculo(int tp){//1 = Carro 2 = Aviao 3 = Navio
853
854         tipo = tp;
855         if(tipo==1){
856             nome="Carro";
```

```
            setvelMax(200);
        }else if(tipo==2){
            nome="Aviao";
            setvelMax(800);
        }if(tipo==3){
            nome="Navio";
            setvelMax(120);
        }


    }

    //////Aula 47**********************************************************************Herança



    //Arquivo Principal


        Moto *v1=new Moto();
        Carro *v2=new Carro();

        cout<<v1->rodas;
        v1->imp();



    //Classes Arquivo .harder


    class Veiculo{

        public:

            int vel;
            int blind;
            int rodas;
            void setTipo(int tp);
            void setVelMax(int vm);
            void setArma(bool ar);
            void imp();

        private:

            int tipo;
            int velMax;
            bool arma;
    };

    void Veiculo::imp(){
        std::cout<<"Tipo Veículo:"<< tipo;
    }

    void Veiculo:setTipo(int tp){
        tipo=tp;
    }

    void Veiculo:setVelMax(int vm){
        velMax=vm;
    }

    void Veiculo:setArma(bool ar){
        arma=ar;
    }

    class Moto::public Veiculo{//Criando uma classe que herda os elemntos da classe veículo

        public:
            moto();
```

```
928
929 };
930
931 Moto::Moto(){ //não foi declado os elementos
932     vel=0;
933     blind=0;
934     rodas=2;
935     setTipo(1);
936     setVelMax(120);
937     setArma(false);
938 }
939
940
941 class Carro::public Veiculo{//Criando uma classe que herda os elemntos da classe
    veículo
942
943     public:
944         Carro();
945
946
947 };
948
949 Carro::Carro(){
950
951     vel=0;
952     blind=0;
953     rodas=4;
954     setTipo(2);
955     setVelMax(120);
956     setArma(false);
957
958
959 }
960
961
962
963
964
965
966 ///////VECTOR
967
968 #include <vector>
969
970 int main(){
971
972 vector<int> num; // Criado um vetor sem tamanho
973 vector<int> num(5); // Com tamanho
974 vector<int> num2; // Criado um vetor sem tamanho
975
976
977 int tam,i;
978
979 num.push_back(10); //Inserir no final do vetor
980
981 tam = num.size(); //armazena o tamanho do vector
982
983 for(i=0;i<tam.size();i++){
984     cout << num[i];
985
986 }
987
988 num1.swap(num2); //troca de valores entre vetores
989 num1.front(); //retorna o primeiro elemento
990 num1.back(); //retorna o primeiro elemento
991 num1.at(tam/2);//retorno o valor do meio
992
993 num1.insert(num1.begin(),8888);//begin -> inicio, valor a inserir
994 num1.insert(num1.begin()+1,88);//begin -> inicio próxima posição indicando o +1
995 num1.insert(num1.end()-1,88); //penultima posição
996 num1.insert(num1.end(),88); //ultima posição
997 num1.erase(num1.end()-1);//eliminar o elemento
998 clear();limpa tudo
999
```

```
1000    while(!num1.empty(){
1001        num1.pop_back;
1002
1003    }//retire até que o array fique vazio
1004
1005    }
1006
1007    //Interator
1008
1009    vector<string>produtos = {"mouse", "teclado"};
1010    vector<string>::iterator it; //apontar o elemento
1011
1012    it=produtos.begin(); //primeiro elemento
1013    it=produtos.end()-1; //ultimo elemento
1014
1015    //advance
1016    //next
1017    //prev
1018
1019    advance(it,3); //avançar
1020
1021    cout <<*next(it,3)<<endl // avançar
1022    cout <<*prev(it,3)<<endl //anda para a esquerda
1023
1024    cout << *it << endl; // colocar como ponteiro
1025
1026    for(it=produtos.begin(); it!=produtos.end();it++){
1027        cout << *it << endl;
1028
1029    }
1030
1031
1032    ///*************************************************Conversão
1033
1034
1035    strupr();
1036
1037    #include <stdio.h>
1038    #include <stdlib.h>
1039
1040    // procedimento que converte uma string para maiúsculo
1041    void maiusculo(char s1[], char s2[]){
1042        int i = 0;
1043        while(s1[i] != '\0'){
1044            s2[i] = toupper(s1[i]);
1045            i++;
1046        }
1047        s2[i] = '\0'; // caracteer que indica o fim da string
1048    }
1049
1050    // procedimento que converte uma string para minúsculo
1051    void minusculo(char s1[], char s2[]){
1052        int i = 0;
1053        while(s1[i] != '\0'){
1054            s2[i] = tolower(s1[i]);
1055            i++;
1056        }
1057        s2[i] = '\0'; // caracteer que indica o fim da string
1058    }
1059
1060    int main() {
1061        char str1[] = "Ola. Bom dia.";
1062        char str2[500];
1063
1064        printf("String original: %s\n", str1);
1065
1066        maiusculo(str1, str2);
1067        printf("String maiuscula: %s\n", str2);
1068
1069        minusculo(str1, str2);
1070        printf("String minuscula: %s\n", str2);
1071
1072        return 0;
```

```
}


//////////////////////////////////////////////////DATA

#include <stdio.h>
#include <time.h>

int main(void) {
    time_t mytime;
    mytime = time(NULL);
    struct tm tm = *localtime(&mytime);
    printf("Data: %d/%d/%d/\n", tm.tm_mday, tm.tm_mon + 1, tm.tm_year + 1900);
}

struct tm {
    int tm_sec;   // Indica os segundos de 0 a 59
    int tm_min;   // Indica os minutos de 0 a 59
    int tm_hour;  // Indica as horas de 0 a 24
    int tm_mday;  // Indica os dias do mês de 1 a 31
    int tm_mon;   // Indica os meses do ano de 0 a 11
    int tm_year;  // Indica o ano a partir de 1900
    int tm_wday;  // Indica o dia da semana de 0 (domingo) até 6 (sábado)
    int tm_yday;  // Indica o dia do ano de 1 a 365
    int tm_isdst; // Indica o horário de verão se for diferente de zero
};



///***Biblioteca Chrono

// é utilizado na versão 11 do C++
#include <chrono>
#include <ctime>

using namespace chrono;

minutes m(1);
seconds s(1);
minutes m=duration_cast<minutes>(s); //converter segundos em minutos para que
funcione corretamente

hours
milliseconds
microseconds
nanoseconds

cout<< s.count() << endl; // contagem

using chrono::system_clock; // acessar o relógio do sistema
duration<int,ratio<60*60*24>> um_dia(1);// duraration é o periodo em segundos

system_clock::time_point hoje=system_clock::now(); // criando um ponto no tempo
system_clock::time_point amanha=hoje + um_dia;
system_clock::time_point ontem=hoje - um_dia;

time_t tt;

tt=system_clock::to_time(hoje);
cout << "Hoje: " << ctime(&tt) << endl;

tt=system_clock::to_time(amanha);
cout << "amanhã: " << ctime(&tt) << endl;


//Tempo
steady_clock::time_point t1 = steady_clock::now(); //tempo
cout << "Imprimindo 1500 estrelas: "<<endl;

for (int i=0; i<1500; i++){
    cout <<"*";
}

```

```
1145  steady_clock::time_point t2 = steady_clock::now(); // outro tempo
1146
1147  duration<double> tempo = duration_cast<duration<double> > (t2-t1);//duration_cast a
      conversão para double
1148
1149  cout <<"Tempo de trabalho: "<< tempo.count() << "segundos";
1150
1151
1152
1153  ////////////////////////////////////////BIBLIOTECA CTIME E TIME.H
1154
1155
1156  time_t t;
1157  struct tm*infotempo; // implementado em ctime
1158
1159  time(&t);
1160  infoTempo=localtime(&t); // receber o tempo decorrido e converter para a estrutura tm
1161
1162  cout << asctime(infoTempo)<<endl; //asctime converte para string
1163
1164  cout<< infoTempo->tm_mday;
1165
1166  ////////////////////////////////////////////
1167
1168  //Em utilizar o ctime não é necessário  implementar a struct como no exemplo acima
1169  time_t t;
1170
1171  time(&t);
1172
1173  cout<<ctime(&t)<<endl;
1174
1175  ////////////////////////////////// strftime formata a saída
1176
1177
1178  time_t t;
1179  struct tm*infoTempo; // implementado em ctime
1180  char buffer[80];
1181
1182  time(&t);
1183  infoTempo=localtime(&t); // receber o tempo decorrido e converter para a estrutura tm
1184
1185
1186
1187  strftime(buffer,80, "Hora: %I : %M" , infoTempo);//resultado, tamanho, como eu quero
      que realize a formatação
1188
1189  cout << buffer<<endl; //asctime converte para string
1190
1191  ///
1192  clock_t c; //tempo
1193  size_t tam; //Armazena tamanho
1194  struct tm * stinfo;
1195  time_t t;
1196
1197
1198  ////////////////////*** Criação de Arquivo
1199
1200  #include <fstream>
1201
1202  //ofstream, ifstream, fstream
1203
1204
1205  //Criar um arquivo
1206
1207  ofstream arquivo;
1208
1209  arquivo.open("impressora.txt");
1210
1211  arquivo << "cfb Cursos";//armazenar no arquivo
1212
1213
1214  arquivo.close();
```