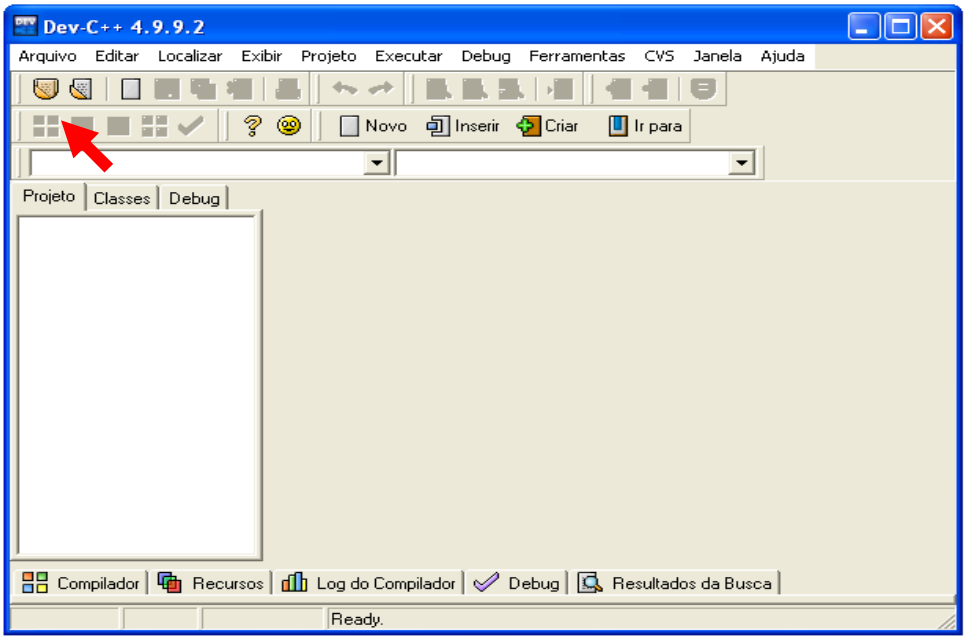
Apostila C Básico

1. Construção e Executar de um Programa no Ambiente Dev-C++

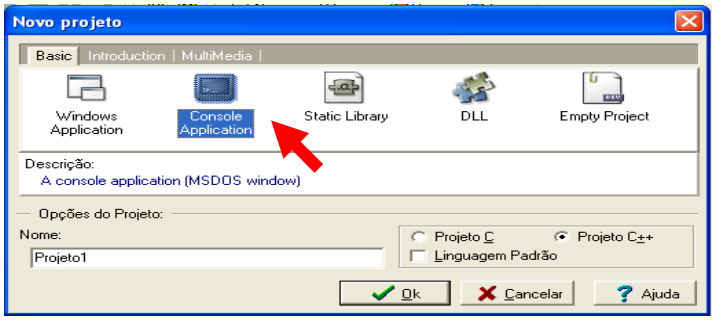
A seguir, iremos mostrar os passos básicos para executar um programa que exibe a tabuada de um número informado pelo usuário.

1. Se você está com o ambiente do Dev-C++ aberto, pule para o próximo passo. Caso ontrário, inicie o ambiente Dev-C++ clicando em Iniciar >> Programas >> Bloodshed Dev-C++ >> Dev-C++;

2. Com o ambiente do Dev-C++ aberto, a primeira coisa a ser feita é criar um projeto. Um projeto no Dev-C++ é o ponto central para gerenciar seus diferentes códigos fontes. Este lhe ajuda a navegar pelos seus programas e permite a configuração de diferentes parâmetros. Por exemplo, configurar se o seu programa será uma aplicação baseada em janelas (Windows Application) ou em caracteres (Console Application). Para criar um projeto você deve clicar em Arquivo >> Novo >> Projeto ou clicar botão Projeto (ver figura abaixo);

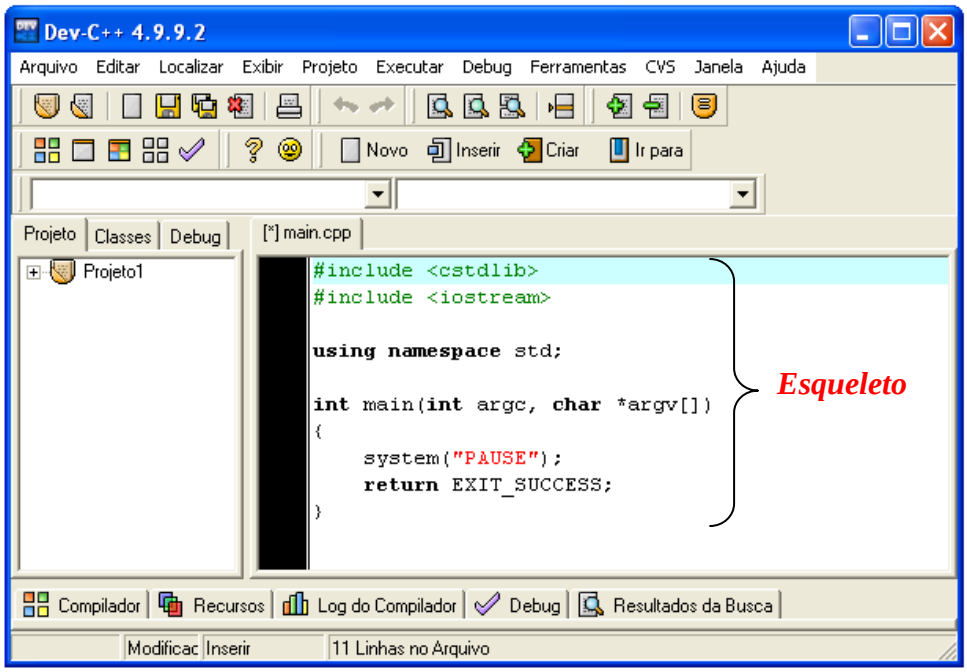


3. A seguir, irá aparecer uma janela para você escolher o tipo de projeto a ser usado. Como este é um curso introdutório sobre programação, nós só iremos trabalhar com projetos com interface a caractere. Por isso escolha a opção "Console Application" (ver figura abaixo) e depois clique em OK;



4. O próximo passo é definir o nome do seu projeto. Escolha um diretório, digite um nome significativo (ex: Tabuada.dev) e clique em *Save*;

5. Depois de realizado os passos anteriores, irá aparecer uma janela (ver figura abaixo) com um trecho de código gerado automaticamente pelo Dev-C++. Esse trecho de código irá funcionar como um esqueleto para construção dos nossos programas, por isso, de preferência, não o modifique, só acrescente o código do programa que iremos fazer;



6. O código fonte a ser acrescentado deve vir após a primeira chave que fica depois da palavra reservada main (ver figura abaixo). Antes de digitar o seu programa, salve-o com um nome significativo. Para isso, clique em Arquivo >> Salvar ou botão *Salvar*. O código a ser digita é:

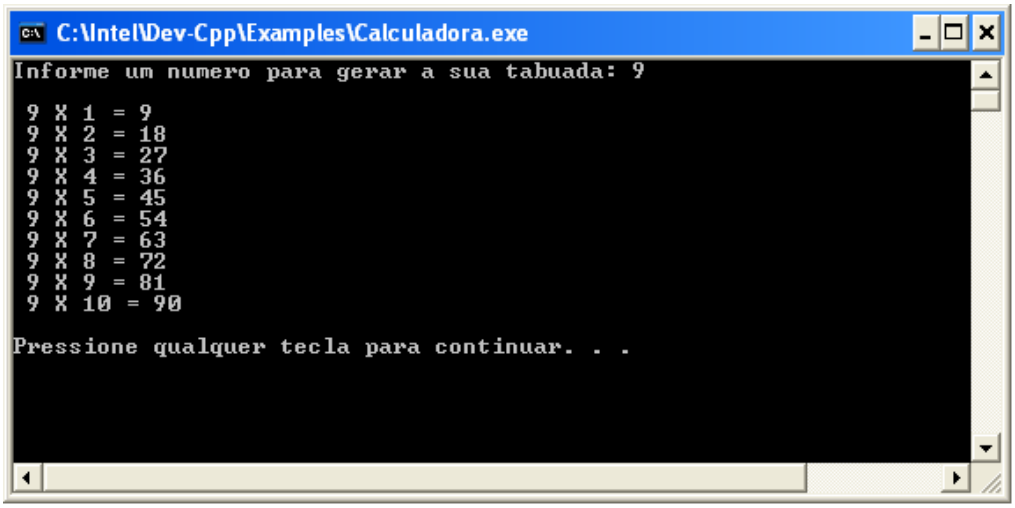
```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
   //Isso é um comentário - o seu código deve começar a partir destes ponto
   int num, n, tab;
   printf("Informe um numero para gerar a sua tabuada: ");
   scanf("%d",&num);
   for(n=1; n<=10;n++)
   {
       tab = num * n;
       printf("\n %d X %d = %d", num, n, tab);
   }
   printf("\n\n");
   system("PAUSE");
   return EXIT_SUCCESS;
}
```

7. Após concluir a digitação, podemos compilar o programa. Para isso, clique em Executar >> Compilar ou no botão Compilar. Se não houver erro, você deve visualizar uma janela indicando que o programa está livre de erros. Se houver erros no programa, uma lista com tais erros é apresentada na janela principal;

8. Considerando que o seu programa foi compilado com sucesso, o próximo passo é executá-lo. Para isso, clique em Executar >> Executar ou no botão Executar. Se tudo transcorreu corretamente, você irá informar um número e será exibida a tabuada deste número (ver a figura abaixo);



9. Pressione qualquer tecla para retornar ao ambiente do Dev-C++.

2. Convertendo algoritmos para a Linguagem C

2.1 Exemplo de algoritmo de seqüência

Faça um algoritmo para ler um preço unitário, uma quantidade e calcular e exibir o preço total.

Pseudo-código	Linguagem C
1. Algoritmo ExemploInstruçõesEntradaSaida 2. Real : preçoUnit, preçoTot; 3. Inteiro : qtd; 4. Início 5. Escreva(“informe preço”); 6. Leia(preçoUnit); 7. Escreva(“informe quantidade”); 8. Leia(qtd); 9. preçoTot ← preçoUnit * qtd; 10. Escreva(“preço total = ”,preçoTot); 11. Fim.	1. #include <cstdlib> 2. #include <iostream> 3. using namespace std; 4. int main(int argc, char *argv[]) 5. { 6. float precoUnit, precoTot; 7. int qtd; 8. printf("informe preço \n"); 9. scanf("%f", &precoUnit); 10. printf("informe quantidade \n"); 11. scanf("%d", &qtd); 12. precoTot = precoUnit * qtd; 13. printf("preço total = %.2f \n",precoTot); 14. system("PAUSE"); 15. return EXIT_SUCCESS; 16. }

Equivalência entre as linhas	
2	6
3	7
4	5
5	8
6	9
7	10
8	11
9	12
10	13
11	16

Algumas observações:

- As linhas 1, 2, 3, 4, 5, 14, 15 e 16 são automaticamente inseridas pelo Dev-C++ quando cria-se um novo projeto;

- Em C, a função *main* representa o programa principal que inicia e termina com abre chaves"{" e fecha chaves "}", respectivamente. Todo programa deve ter uma única função *main*;

- Em C, uma chave aberta "{" equivale ao início de um bloco de comandos e a chave fechada "}" equivale ao fim de um bloco de comandos;

- Observe que ao final de cada linha do programa C há um ponto-e-vírgula ";" indicando a conclusão do comando;

- Em C, como na maioria das linguagens, os dados são divididos em tipos: inteiro, real, caracter, etc. Abaixo segue uma lista dos tipos básicos de dados permitidos em C.

Tipo	**Intervalo**	**Observação**
char	-128 a 127	número muito pequeno e caracter ASCII
int	-32768 a 32767	contador, controle de laço
float	3.4e-38 a 3.4e38	real (precisão de 7 dígitos)
double	1.7e-308 a 1.7e308	científico (precisão de 15 dígitos)

- Exemplos de declarações válidas para variáveis (usar as mesmas regras para nomear identificadores):
  - int i;
  - int x,y,z;
  - char letra;
  - float nota_1,nota_2,media;
  - double num;

- As funções *printf()* e *scanf()* permitem, respectivamente, escrever no vídeo e ler do teclado o valor de variáveis. Estas funções são equivalentes aos comandos Escreva e Leia em pseudo-código. Perceba nestas funções que aparecem alguns caracteres precedidos pelo símbolo de %. Estes caracteres são string de controle que são usadas para especificar o formato do tipo de dado que será exibido (*printf*) ou armazenado (*scanf*) pelas variáveis. As principais string de controle são:
  - %c – para o tipo de dado Caracter;
  - %d – para o tipo de dado Inteiro;
  - %f – para o tipo de dado Real;
  - %s – para o tipo de dado String (não visto até o momento);

- Antes de um *scanf()* para leitura de um caractere, recomenda-se o uso da função *fflush(stdin),* a qual limpa da memória o último ENTER pressionado. Uma alternativa para ler um caractere sem usar *fflush(stdin)* é usar a função *getche()* ou *getch()* da biblioteca *conio.h.* A primeira função lê e exibe o caractere digitado, enquanto que a segunda lê e não exibe o caractere digitado. Em ambas não é necessário digitar ENTER. Testes os exemplos abaixo;

Errado – Sem fflush()	Certo – Com fflush()	Certo – Sem fflush() e com getche()

`#include <cstdlib>`
`#include <iostream>`
`using namespace std;`
`int main(int argc, char *argv[])`
`{`
`  int x;`
`  char y;`
`  printf("tecle um número \n");`
`  scanf("%d",&x);`
`  printf("tecle um caractere \n");`
`  scanf("%c",&y);`
`  printf("x = %d e y= %c \n",x,y);`
`  system("PAUSE");`
`  return EXIT_SUCCESS;`
`}`
	`#include <cstdlib>`
	`#include <iostream>`
	`using namespace std;`
	`int main(int argc, char *argv[])`
	`{`
	`  int x;`
	`  char y;`
	`  printf("tecle um número \n");`
	`  scanf("%d",&x);`
	`  fflush(stdin);`
	`  printf("tecle um caractere \n");`
	`  scanf("%c",&y);`
	`  printf("x = %d e y= %c \n",x,y);`
	`  system("PAUSE");`
	`  return EXIT_SUCCESS;`
	`}`
		`#include <cstdlib>`
		`#include <iostream>`
		`#include <conio.h>`
		`using namespace std;`
		`int main(int argc, char *argv[])`
		`{`
		`  int x;`
		`  char y;`
		`  printf("tecle um número \n");`
		`  scanf("%d",&x);`
		`  printf("tecle um caractere \n");`
		`  y = getche();`
		`  printf("x = %d e y= %c \n",x,y);`
		`  system("PAUSE");`
		`  return EXIT_SUCCESS;`
		`}`

- Além das string de controle, você também pode usar operadores especiais para tornar a exibição do *printf* mais sofisticada. A seguir é apresentado os principais operadores especiais suportados por *printf*. Note que usamos o operador especial \n no nosso exemplo anterior.
  - \b - Retrocesso
  - \n - Linha Nova
  - \t - Tabulação Horizontal
  - \’ - Apóstrofo
  - \" - Aspas
  - \\ - Barra invertida
  - %% - Percentual

- Outro recurso especial do *printf* é o uso de números que delimitam a quantidade de casas numéricas a serem usadas para exibir um número. Um exemplo deste recurso é apresentado na linha 13 do primeiro exemplo desta seção. Nesta linha, o delimitador “%.2f” formata a variável precoTot para que esta tenha apenas 2 casas decimais. A seguir são exibidos outros exemplos de uso do printf para formatação da saída de números;

Programa em C – Formatação para números inteiros	Resultado exibido na tela

`#include <cstdlib>`
`#include <iostream>`
`using namespace std;`
`int main(int argc, char *argv[])`
`{`
`  int pec = 10 ;`
`  printf("O reajuste foi de %2d%%\n",pec);`
`  printf("O reajuste foi de %4d%%\n",pec);`
`  printf("O reajuste foi de %6d%%\n",pec);`
`  system("PAUSE");`
`  return EXIT_SUCCESS;`
`}`
	`O reajuste foi de 10%`
	`O reajuste foi de   10%`
	`O reajuste foi de     10%`
	`Press any key to continue . . .`

Programa em C – Formatação para números reais	Resultado exibido na tela

`#include <cstdlib>`
`#include <iostream>`
`using namespace std;`
`int main(int argc, char *argv[])`
`{`
`  float taxa = 35;`
`  printf("a taxa foi de %4.2f%%\n",taxa);`
`  printf("a taxa foi de %4.4f%%\n",taxa);`
`  printf("a taxa foi de %4.6f%%\n",taxa);`
`  system("PAUSE");`
`  return EXIT_SUCCESS;`
`}`
	`a taxa foi de 35.00%`
	`a taxa foi de 35.0000%`
	`a taxa foi de 35.000000%`
	`Press any key to continue . . .`

- Cada variável a ser lida com scanf deverá ser precedida pelo operador “&”, o qual representa o "endereço" de memória que a variável será armazenada. Contudo, não se preocupe com isso agora, só não esqueça de colocar o operador “&” antes de cada variável. Para a leitura de uma string do teclado, também pode-se usar a função gets() – veremos isso na seção 5;

- O sinal de atribuição em C é o = (igual). Para usar a operação de igualdade fazemos == (igualigual);

- Para realizar operações aritméticas a linguagem C tem os seguintes operadores (ver quadro abaixo). Contudo, além destes operadores, existe o operador % (chamado operador de módulo), o qual retorna o resto da divisão inteira entre dois números. O operador % é igual ao operador MOD que aprendemos em pseudo-código;

    - \+ (adição);
    - \- (subtração);
    - * (multiplicação);
    - / (divisão);
    - % (resto da divisão inteira).

Programas:

(1) Leia nome, idade e salário de um funcionário e exiba os mesmos dados, contudo o salário deve ser reajustado em 12%;

(2) Leia a base e a altura de um triângulo. Em seguida, escreva a área do mesmo;

(3) Leia uma temperatura em Fahrenheit e exiba o equivalente em Celsius;

(4) Leia uma distância em km, o preço da gasolina em reais e exiba quantos litros de gasolina o carro irá consumir e quanto será gasto em reais. Considere que o carro faz 12 km/l de gasolina;

(5) Leia um valor inteiro em segundos e, depois convertê-lo, mostre-o no formato hh:mm:ss.

2.2 Exemplo de algoritmo de decisão simples

Faça um algoritmo para ler um número e escrever X>0 quando este número for positivo.

Pseudo-código	Linguagem C
1. Algoritmo ExemploDecisãoSimples 2. Inteiro : x; 3. Início 4. Escreva(“Digite um valor”); 5. Leia (x); 6. Se (x > 0) Então 7. Escreva (“X > 0”); 8. Fim.	1. #include <cstdlib> 2. #include <iostream> 3. using namespace std; 4. int main(int argc, char *argv[]) 5. { 6. int x; 7. printf("Digite um valor \n"); 8. scanf("%d", &x); 9. if (x>0) 10. printf("X>0 \n"); 11. system("PAUSE"); 12. return EXIT_SUCCESS; 13. }
Equivalência entre as linhas	
2	6
3	5
4	7
5	8
6	9
7	10
8	13

Algumas observações:

- As linhas 1, 2, 3, 4, 5, 11, 12 e 13 são automaticamente inseridas pelo Dev-C++ quando cria-se um novo projeto;

- A sintaxe de uma estrutura de decisão simples é a seguinte:

```
if (condição)
{
   conjunto de comandos do bloco verdade
}
```

- No exemplo acima, se a condição não for satisfeita, então o fluxo de execução do programa sai da estrutura de decisão sem executar nada;

- O operador de igualdade na linguagem C é dado pelo operador == (igual igual). Além deste operador, a linguagem C tem mais 5 operadores relacionais (ver quadro abaixo).
  - > maior que
  - < menor que
  - >= maior ou igual a
  - <= menor ou igual a
  - != diferente de

- Além dos operadores aritméticos e relacionais, a linguagem C tem 3 operadores lógicos (ver abaixo). Ressalta-se que devido ao fato da linguagem C não possuir um tipo de dado booleano, as operações lógicas irão retornar apenas os inteiros 0 e 1 para representar, respectivamente, os valores falso e verdadeiro.
  - && E
  - || OU
  - ! NÃO

2.2 Exemplo de algoritmo de decisão simples

Faça um algoritmo para ler um número e escrever X>0 quando este número for positivo ou X<=0 quando este for negativo ou igual a zero.

Pseudo-código	Linguagem C
1. Algoritmo ExemploDecisãoComposta 2. Inteiro: x; 3. Início 4. Escreva("Digite um valor"); 5. Leia (x); 6. Se (x > 0) Então 7. Escreva ("X > 0"); 8. Senão 9. Escreva ("X <= 0"); 10. Fim.	1. #include <cstdlib> 2. #include <iostream> 3. using namespace std; 4. int main(int argc, char *argv[]) 5. { 6. int x; 7. printf("Digite um valor \n"); 8. scanf("%d", &x); 9. if (x>0) 10. printf("X>0 \n"); 11. else 12. printf("X<=0 \n"); 13. system("PAUSE"); 14. return EXIT_SUCCESS; 15. }

Equivalência entre as linhas	
2	6
3	5
4	7
5	8
6	9
7	10
8	11
9	12
10	15

Algumas observações:

- A sintaxe de uma estrutura de decisão composta é a seguinte:

```
if (condição)
{
   conjunto de comandos do bloco verdade
}
else
{
   conjunto de comandos do bloco falso
}
```

- Se a condição não for satisfeita, então o fluxo de execução do programa procura o *else* posterior ao *if,* e executa o bloco de comandos do *else*.

Programas

(1) Leia um número inteiro e mostre uma mensagem indicando se este número é par ou ímpar, e se é positivo ou negativo;

(2) Leia a idade de um nadador e exiba sua categoria segundo as regras: A(5 até 7); B(8 até 10); C(11 até 13); D(14 até 18) e E(Idade > 18);

(3) Leia uma milhar e informe se esse número é palíndromo. Exemplos de números palíndromos: 9889, 7337 e 2002.

2.3 Exemplo de algoritmo de repetição condicional

Faça um algoritmo para ler vários números até que 0 seja digitado (escrever "X = 0"). Para cada número, informar se ele é positivo (escrever "X > 0") ou negativo (escrever "X < 0").

Pseudo-código

```
1. Algoritmo ExemploEnquantoFaça-TesteNoInicio
2. Inteiro : x;
3. Início
4.     Escreva("Digite um valor");
5.     Leia (x);
6.     Enquanto (x<>0) faça
7.           Se (x > 0) Então
8.              Escreva ("X > 0");
9.           Senão
10.             Escreva ("X < 0");
11.          Escreva("Digite um valor");
12.          Leia (x);
13.    Fim Enquanto;
14.    Escreva ("X = 0");
15. Fim.
```

Linguagem C

```
1.  #include <cstdlib>
2.  #include <iostream>
3.  using namespace std;
4.  int main(int argc, char *argv[])
5.  {
6.      int x;
7.      printf("Digite um valor \n");
8.      scanf("%d", &x);
9.      while (x!=0){
10.         if (x > 0)
11.             printf("X > 0 \n");
12.         else
13.             printf("X < 0 \n");
14.         printf("Digite um valor \n");
15.         scanf("%d", &x);
16.     }
17.     printf("X = 0 \n");
18.     system("PAUSE");
19.     return EXIT_SUCCESS;
20. }
```

Equivalência entre as linhas

2	6
3	5
4	7
5	8
6	9
7	10
8	11
9	12
10	13
11	14
12	15
13	16
14	17
15	20

Algumas observações:

- As linhas 1, 2, 3, 4, 5, 18, 19 e 20 são automaticamente inseridas pelo Dev-C++ quando cria-se um novo projeto;

- A sintaxe do laço *while* é a seguinte:

  ```
  while (condição)
  {
          conjunto de comandos do laço
  }
  ```

- O laço *while* se repete enquanto a sua condição for verdadeira. Como o teste do laço é feito no início, então, se no primeiro teste a condição for falsa, o laço não é executado nenhuma vez. Se a condição for verdadeira, o conjunto de comandos do *while* é executado uma vez e a condição do laço é avaliada novamente. Entenda o exemplo a seguir;

  ```
  n = 0;
  while (n < 100) //este laço será executado 100 vezes
  {
      n=n+1;
  }
  ```

2.4 Exemplo de algoritmo de repetição contada

Faça um algoritmo para ler 20 números e informar a quantidade de números positivos que foram digitados.

Pseudo-código

```
1. Algoritmo ExemploParaFaça-RepetiçãoContada
2. inteiro : x, i, contP;
3. Inicio
4.     contP←0;
5.     Para i←1 até 20 faça
6.        Escreva("Digite um valor");
7.        Leia (x);
8.        Se (x > 0) Então
9.           contP ← contP + 1;
10.    Fim Para;
11.    Escreva ("positivos = ",contP);
12. Fim.
```

Linguagem C

```
1.  #include <cstdlib>
2.  #include <iostream>
3.  using namespace std;
4.  int main(int argc, char *argv[])
5.  {
6.      int x, i, contP;
7.      contP = 0;
8.      for (i=1; i<=20; i++){
9.         printf("Digite um valor \n");
10.        scanf("%d", &x);
11.        if (x > 0)
12.           contP++;
13.    };
14.    printf("positivos = %d \n", contP);
15.    system("PAUSE");
16.    return EXIT_SUCCESS;
17. }
```

Equivalência entre as linhas

2	6
3	5
4	7
5	8
6	9
7	10
8	11
9	12
10	13
11	14
12	17

Algumas observações:

- As linhas 1, 2, 3, 4, 5, 15, 16 e 17 são automaticamente inseridas pelo Dev-C++ quando cria-se um novo projeto;

- A sintaxe do laço *for* é a seguinte:

```
for (inicialização; condição; incremento)
{
        conjunto de comandos do laço
}
```

onde:

- inicialização é uma expressão de inicialização do contador.
- condição é uma expressão lógica de controle do laço.
- incremento é uma expressão de incremento do contador.

• No exemplo acima, podemos observar que foi utilizado a expressão i++ e contP++. Tais expressões são formas resumidas de se escrever i = i + 1 e contP = contP + 1, respectivamente. Ou seja, o operador ++ equivale a operação de fazer o incremento de 1 à variável. O oposto do operador ++ é o operador – –. Assim, i – – é igual a i=i – 1.

Programas

(1) Leia um número e escreva o seu somatório

(2) Leia um número e escreva o seu fatorial. Lembrar que fatorial de (0!) é 1.

(3) Leia dois números inteiros, n e m, e calcule a soma dos números entre n e m.

(4) Leia um conjunto de 100 números inteiros positivos e escreva o maior e o menor deles.

(5) Leia um número inteiro e escreva se ele é primo.

(6) Leia um número inteiro e positivo e escreva o numero inteiro que mais se aproxima da sua raiz quadrada.