

Estrutura de Dados

Aula 04 - Linguagem de
programação C – 2ª parte

Linguagem C

Laços em C:

É muito comum, em programas computacionais, termos procedimentos iterativos, ou seja, procedimentos que devem ser executados em vários passos. Vamos analisar como exemplo abaixo o cálculo do valor do fatorial de um número inteiro não negativo.

$b! = b \times (b - 1) \times (b - 2) \dots 3 \times 2 \times 1$, onde $0! = 1$

Laços em C

Mas antes disso vamos ver que a linguagem C nos ajuda nesta interação por isso podemos fazer essas interações através de laços de repetição.

Laço de repetição: while

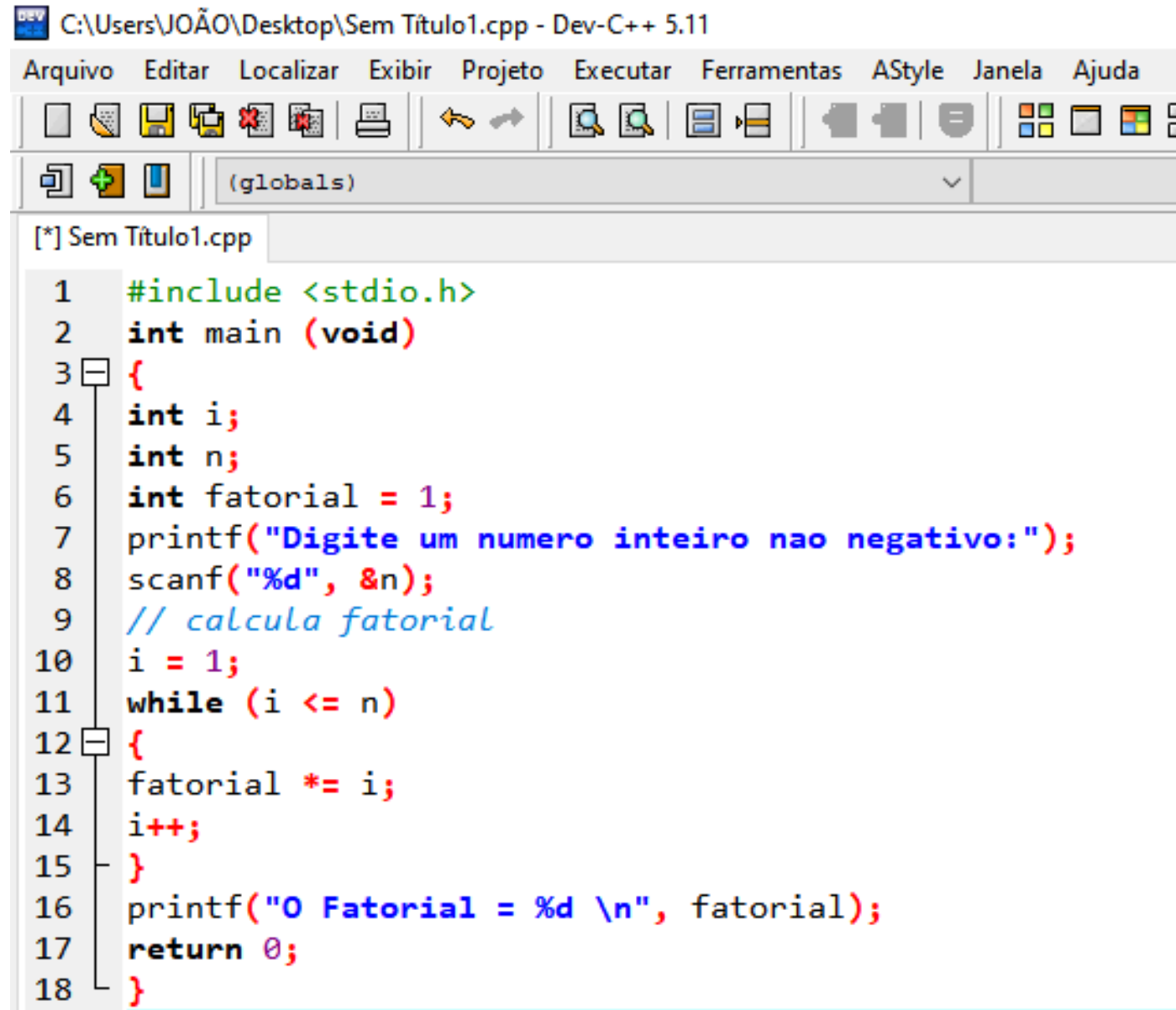
Exemplo:

```
1  int i = 1;
2  while (i <= 10) {
3      printf("%d", i++);
4  }
```

Ou

```
1  int i = 10;
2  while (i <= 10) {
3      printf("%d", i--);
4  }
5  |
```

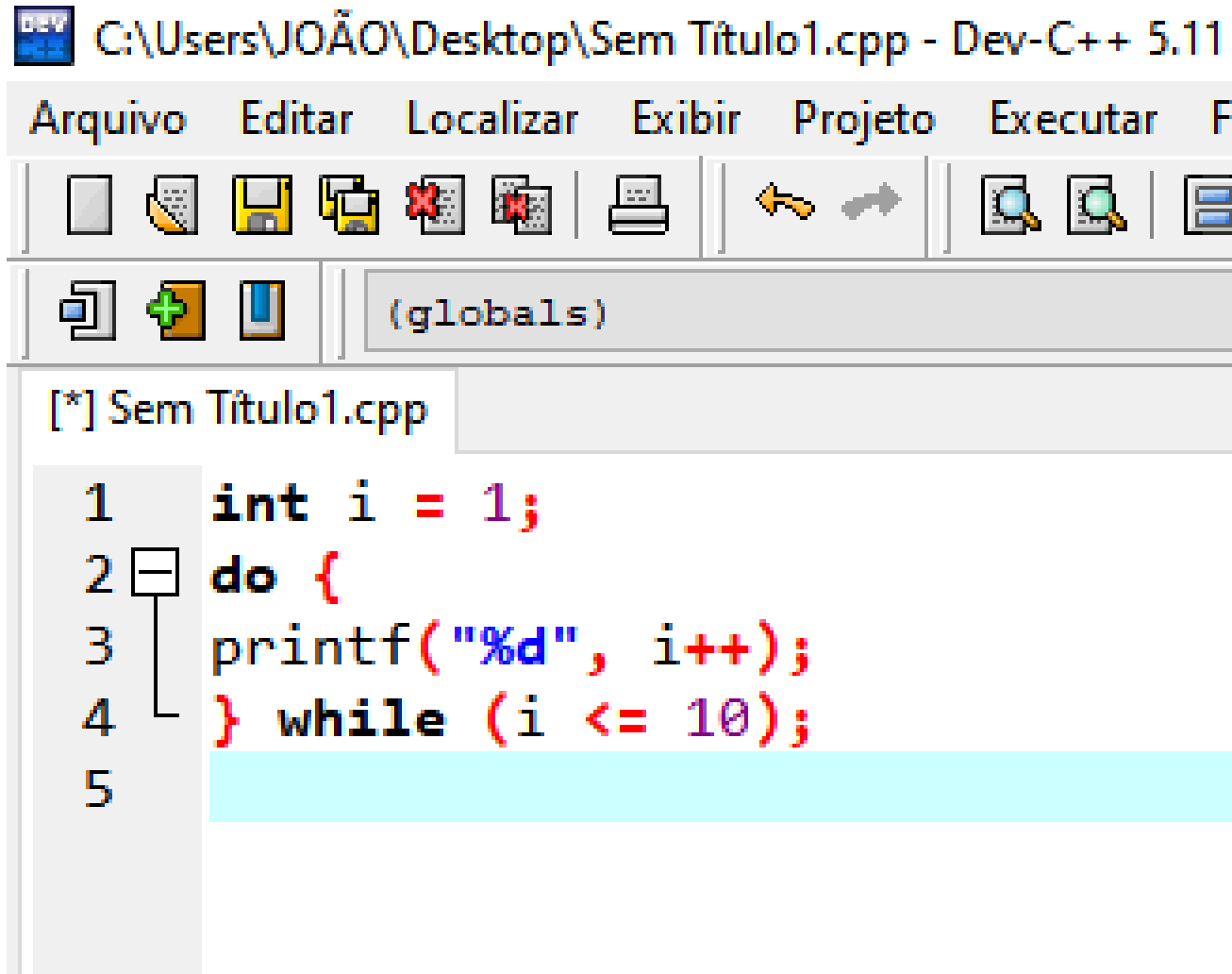
Vamos ver a solução do exemplo do fatorial com o laço While:



The screenshot shows the Dev-C++ 5.11 IDE interface. The title bar indicates the file path: C:\Users\JOÃO\Desktop\Sem Título1.cpp - Dev-C++ 5.11. The menu bar includes: Arquivo, Editar, Localizar, Exibir, Projeto, Executar, Ferramentas, AStyle, Janela, Ajuda. The toolbar contains icons for file operations (new, open, save, print, delete), editing (undo, redo), and execution (run, step-through). The variable explorer on the left shows a list of variables: i, n, and fatorial, all of type int. The main editor window displays the following C++ code:

```
[*] Sem Título1.cpp
1  #include <stdio.h>
2  int main (void)
3  {
4      int i;
5      int n;
6      int fatorial = 1;
7      printf("Digite um numero inteiro nao negativo:");
8      scanf("%d", &n);
9      // calcula fatorial
10     i = 1;
11     while (i <= n)
12     {
13         fatorial *= i;
14         i++;
15     }
16     printf("O Fatorial = %d \n", fatorial);
17     return 0;
18 }
```

Laço de repetição: Do



The screenshot shows the Dev-C++ 5.11 IDE interface. The title bar indicates the file path: C:\Users\JOÃO\Desktop\Sem Título1.cpp - Dev-C++ 5.11. The menu bar includes Arquivo, Editar, Localizar, Exibir, Projeto, Executar, and Ferramentas. The toolbar contains icons for file operations (new, open, save, print, etc.) and editing (undo, redo, find, etc.). The variable declaration section shows (globals). The main editor window displays the code for Sem Título1.cpp:

```
1  int i = 1;
2  do {
3      printf("%d", i++);
4  } while (i <= 10);
5
```

The code implements a do-while loop that prints the value of i from 1 to 10. The loop body is highlighted in light blue.

Laço de repetição: For

Exemplo:

DEV C++ C:\Users\JOÃO\Desktop\Sem Título1.cpp - Dev-C++ 5.11

Arquivo Editar Localizar Exibir Projeto Executar Ferramentas AStyle Janela Ajuda

(globals)

[*] Sem Título1.cpp

```
1 for (i=1;i<=10;i++){  
2     printf("\n %d", i);  
3 }  
4
```

Como ficaria o algoritmo do Fatorial já mostrado utilizando o For?

DEV C++ C:\Users\JOÃO\Desktop\Sem Título1.cpp - Dev-C++ 5.11

Arquivo Editar Localizar Exibir Projeto Executar Ferramentas AStyle Janela Ajuda

(globals)

[*] Sem Título1.cpp

```
1  #include <stdio.h>
2  int main (void)
3  {
4      int i;
5      int n;
6      int fatorial = 1;
7      printf("Digite um número inteiro nao negativo:");
8      scanf("%d", &n);
9      // calculo do fatorial
10     for (i = 1; i <= n; i++)
11     {
12         fatorial *= i;
13     }
14     printf("O Fatorial = %d \n", fatorial);
15     return 0;
16 }
```

Como exemplo podemos incrementar o algoritmo acima já analisado criando uma função em C e mostrando seu fatorial:

Vamos ver o que acontece na memória:


```
Arquivo  Editar  Localizar  Exibir  Projeto  Executar  Ferramentas  AStyle  Janela  Ajuda

Sem Título1.cpp

1  #include <stdio.h>
2  int fat (int n);
3  int main (void)
4  {
5      int n = 0;
6      int r;
7      printf("Digite um numero inteiro nao negativo");
8      scanf("%d",&n);
9      r = fat ( n );
10     printf("Fatorial de %d = %d \n", n, r);
11     return 0;
12 }
13 //criando uma função de para calcular o fatorial
14 int fat (int n)
15 {
16     int fatorial = 1.0;
17     while (n != 0)
18     {
19         fatorial *= n;
20         n--;
21     }
22     return fatorial;
23 }
```

Vamos falar de vetores em C ?

O vetor é uma estrutura de dados linear que necessita de somente um índice para que seus elementos sejam endereçados. É utilizado para armazenar uma lista de valores do mesmo tipo, ou seja, o tipo vetor permite armazenar mais de um valor em uma mesma variável.

Exemplo de vetores:

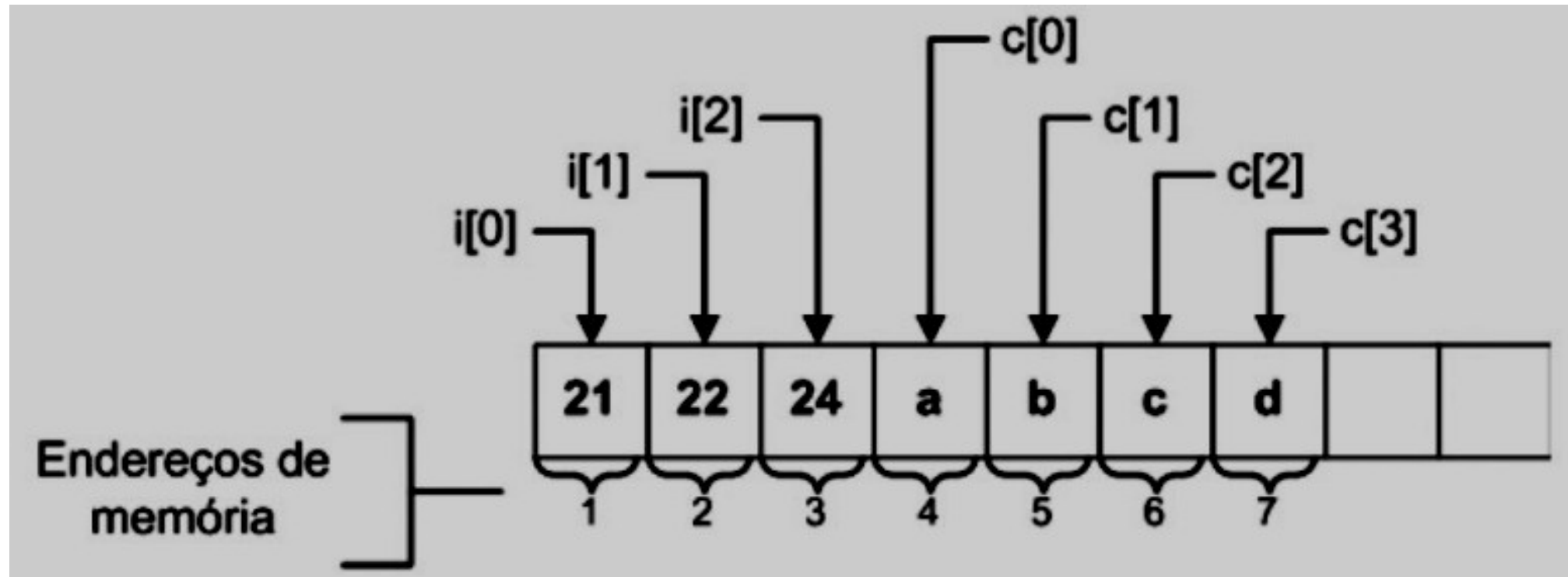
```
int vetorX[10]; //Vetor de inteiros
```

obs: vetorX[0] é o primeiro elemento e
vetorX[9] o último.

exemplo:

```
int x[5];  
x[0]=12;  
x[1]=13;  
x[2]=34;  
x[3]=72;  
x[4]=01;
```

Um exemplo o que esta acontecendo na memoria do computador:



Algumas observações:

características de um vetor:

- Alocação estática (deve-se conhecer as dimensões da estrutura no momento da declaração em C)
 - Estrutura homogênea
 - Alocação sequencial (bytes contíguos)
 - Inserção/Exclusão
 - Realocação dos elementos;
 - Posição de memória não liberada;
- Vamos ver como funciona um vetor simples:

Exemplo pratico:

C:\Users\JOÃO\Desktop\Sem Título1.cpp - Dev-C++ 5.11

Arquivo Editar Localizar Exibir Projeto Executar Ferramentas AStyle Janela Ajuda

(globals)

[*] Sem Título1.cpp

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main(){
4      float notas[2];
5      printf("Digite a primeira nota:");
6      scanf("%f",&notas[0]);
7      printf("Digite a segunda nota:");
8      scanf("%f",&notas[1]);
9      printf("O aluno tirou as notas %.1f e %.1f.",notas[0],notas[1]);
10     return 0;
11 }
12
```

Exercício 01:

Faça um algoritmo que receba valores inteiros de em um vetor de tamanho 3x2 e preencha um vetor inteiro de tamanho 6. Imprima o vetor preenchido.

DEV C++ C:\Users\JOÃO\Desktop\Sem Título1.cpp - Dev-C++ 5.11

Arquivo Editar Localizar Exibir Projeto Executar Ferramentas

(globals)

[*] Sem Título1.cpp

```
1  #include<stdio.h>
2  #include<math.h>
3  int main(){
4      int vetor[6], i;
5      for (i = 0; i < 6; i++){
6          scanf("%d",&vetor[i]);
7          printf("%d \n",vetor[i]);
8      }
9      return 0;
10 }
11
```

Exercício 02:

Faça um cadastro de várias matrículas de alunos da faculdade e armazene-os em um vetor até o mesmo ser preenchido por 18 matrículas. Esses números são distintos, ou seja, o vetor não armazenará valores repetidos.



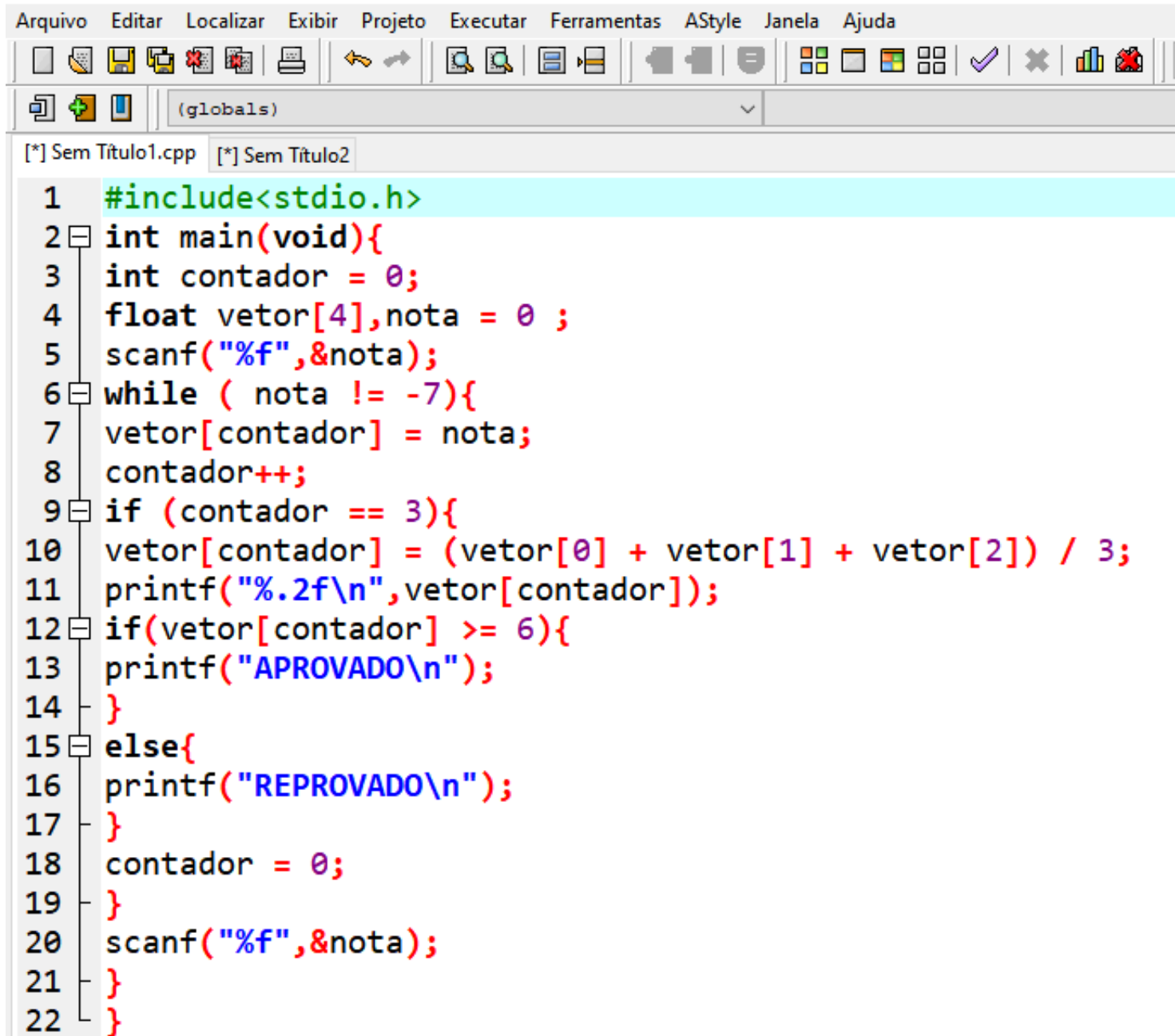
[*] Sem Título1.cpp

```
1  #include<math.h>
2  #include<stdio.h>
3  #include<string.h>
4  int main(){
5      int vetor[18], numero, cont, posicao = 0 ;
6      while (posicao < 18){
7          scanf("%d",&numero);
8          if (posicao == 0){
9              vetor[posicao] = numero;
10             printf("%d\n",vetor[posicao]);
11             posicao++;
12         }
13         else{
14             for(cont = 0; (cont < posicao)&&(vetor[cont] != numero); cont++);
15             if (cont >= posicao){
16                 vetor[posicao] = numero;
17                 printf("%d\n",vetor[posicao]);
18                 posicao++;
19             }
20         }
21     }
22     return 0;
23 }
```

Exercício 03:

Criar 4 vetores, o primeiro com a nota da primeira prova, o segundo com a nota da segunda prova e o terceiro, no quarto vetor com a média das 3 primeiras notas, e imprima o resultado “APROVADO” para aqueles que obtiverem uma média igual ou acima de 7, e “REPROVADO” para quem obtiverem uma média abaixo de 7.

OBS.: Saia do laço quando a primeira nota for igual a -7.



```
1  #include<stdio.h>
2  int main(void){
3      int contador = 0;
4      float vetor[4], nota = 0 ;
5      scanf("%f",&nota);
6      while ( nota != -7){
7          vetor[contador] = nota;
8          contador++;
9      if (contador == 3){
10         vetor[contador] = (vetor[0] + vetor[1] + vetor[2]) / 3;
11         printf("%.2f\n",vetor[contador]);
12     if(vetor[contador] >= 6){
13         printf("APROVADO\n");
14     }
15     else{
16         printf("REPROVADO\n");
17     }
18     contador = 0;
19 }
20     scanf("%f",&nota);
21 }
22 }
```

Vamos falar sobre matrizes em C ?

Também chamado de vetores multidimensionais, esses tipos de vetores são divididos em linhas e colunas de dados. Como por exemplo nesta declaração:

```
float mat[2][2]; //Tabela com 2 linhas e 2 colunas
```

Matriz M

i/j

	0	1
0		
1		

Também chamado de vetores multidimensionais, esses tipos de vetores são divididos em linhas e colunas de dados.

```
int x[2][5]; //Tabela com 2 linhas e 5 colunas
```

Uma melhor visualização seria :

<Variável(x) >	Coluna 0	Coluna 1	Coluna 2	Coluna 3	Coluna 4
Linha 0	2	54	60	23	44
Linha 1	4	63	7	11	55

Um exemplo de utilização dessa matriz seria para um programa para uma loja de departamentos com 2 filiais, cada uma vendendo 3 itens, poderia incluir um vetor declarado como abaixo:

```
int vendas [2][3];
```

Cada elemento `vendas [i] [j]` representa a quantidade do item `j` vendida na filial `i`, declarando como demonstrado abaixo:

C:\Users\JOÃO\Desktop\Sem Título1.cpp - Dev-C++ 5.11

Arquivo Editar Localizar Exibir Projeto Executar Ferramentas AStyle Janela Ajuda

(globals)

Sem Título1.cpp [*] Sem Título2

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(){
4     int vendas [2][3];
5     int i,j;
6     for(i=0;i<=1;i++){
7         printf("Filial 0-: %d \n",i+1);
8         for(j=0;j<=2;j++){
9             printf("Digite o item 0-: %d:",j+1);
10             scanf("%d",&vendas[i][j]);
11         }
12         system("cls");
13     }
14     printf("Itens cadastrados com sucesso!!!");
15     return 0;
16 }
```

Exercício

Construa um sistema de controle de estoque de uma pequena padaria. Seu programa deve oferecer um menu para (a) cadastrar um novo produto; (b) aumentar o estoque de um produto cadastrado (quando há compras); (c) diminuir o estoque de um produto cadastrado (quando há vendas); e (d) imprimir os produtos cadastrados e suas características. De cada produto deseja-se cadastrar: nome (tamanho máximo 50 caracteres) e preço (número real) além, da quantidade em estoque. Assuma que a padaria venderá no máximo, 100 produtos diferentes.

OBS: Para facilitar a localização de um produto (para as funções (b) e (c)), use a posição dos mesmos no vetor que representa o estoque como código do produto, informando-a ao fim do cadastro. Então, peça ao usuário para digitar o código do produto que deseja aumentar/diminuir o estoque.

Tipos de dados abstratos

As propriedades lógicas de um tipo de dado é o tipo de dado abstrato, ou TDA.

Fundamentalmente, um tipo de dado significa um conjunto de valores e uma sequência de operações sobre estes valores.

Este conjunto e estas operações formam uma construção matemática que pode ser implementada usando determinada estrutura de dados do hardware ou do software.

A expressão "tipo de dado abstrato" refere-se ao conceito matemático básico que define o tipo de dado.

Como podemos definir novos tipos de dados mas antes disso temos que pensar em
A estrutura da informação propriamente dita.
Na linguagem de programação C representamos estrutura da seguinte forma:

```
struct coordenada {  
int x;  
int y;  
}
```

Tipo de dados abstrato

A palavra reservada para isso é a
struct.

```
struct coordenada {  
    int x;  
    int y;  
}  
  
struct coordenada coord;  
coord.x = 3;
```

Com base no exemplo acima represente novas estrutura de informações, e imprima na tela do console esse dados, lembrando que tem que ser 15 tipos novos.

Como visto nos slide anterior vamos agora criar novas estruturas abstratas de dados, para isto utilizamos a palavra reservada *TYPEDEF*:

```
typedef struct coordenada {  
    int x;  
    int y;  
} Coordenada;  
Coordenada c;  
c.x = 10;
```

Ponteiros

Para começar temos que analisar três propriedades que um programa deve manter quando armazena dados:

- onde a informação é armazenada;
- que valor é mantido lá;
- que tipo de informação é armazenada.

A definição de uma variável simples obedece a estes três pontos. A declaração prove o tipo e um nome simbólico para o valor. Também faz com que o programa aloque memória para o valor e mantenha o local internamente.

Ponteiros operador de endereço: &

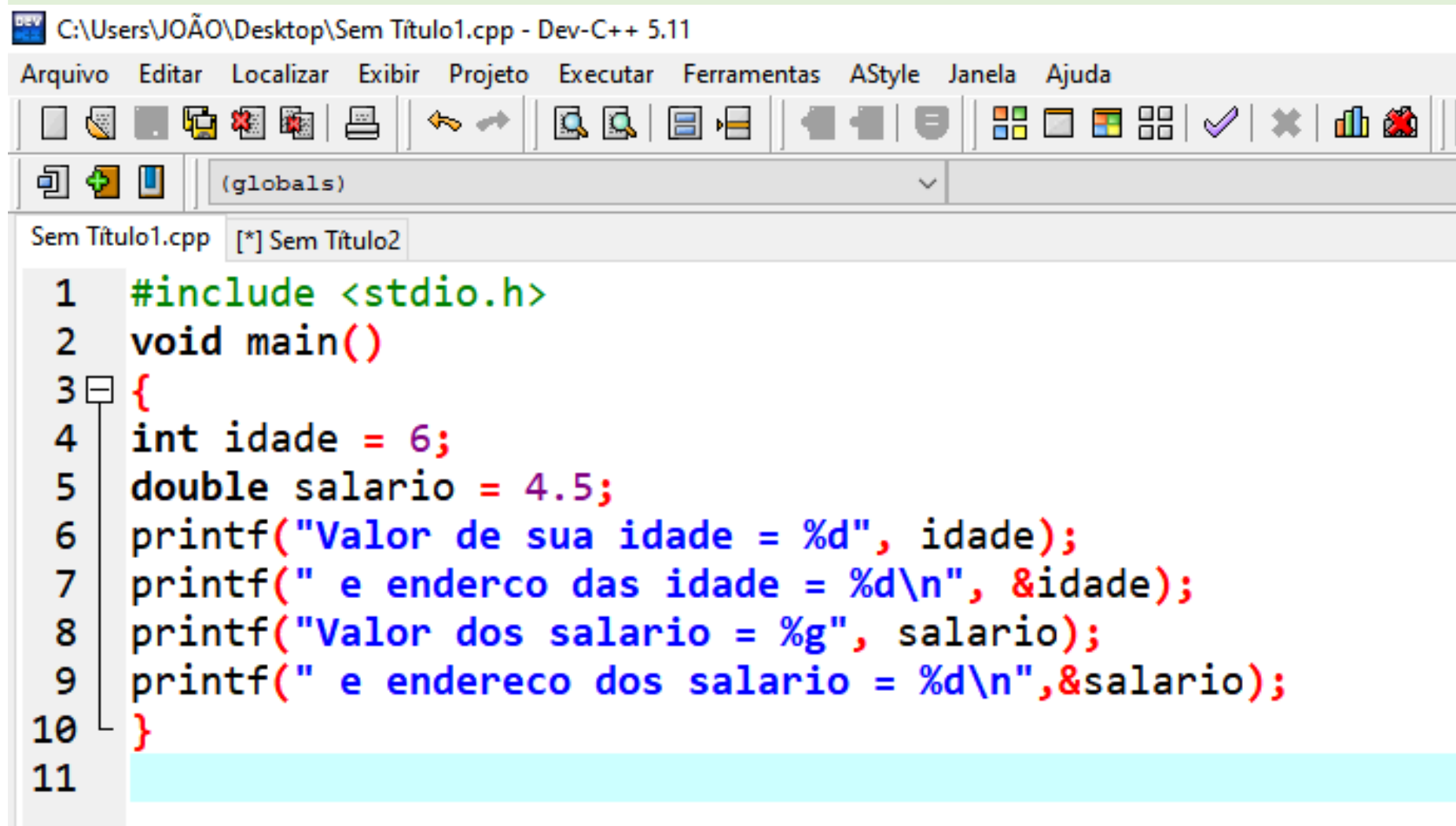
Segunda estratégia baseada em ponteiros, que são variáveis que armazenam endereços ao invés dos próprios valores.

Mas antes de discutir ponteiros, vejamos como achar endereços explicitamente para variáveis comuns.

Aplique o operador de endereço, &, a uma variável para pegar sua posição; por exemplo, se *notas* e uma variável, *¬as* e seu endereço.

Ponteiro operador de endereço: &

Exemplo:



The screenshot shows the Dev-C++ 5.11 IDE interface. The title bar indicates the file path: C:\Users\JOÃO\Desktop\Sem Título1.cpp - Dev-C++ 5.11. The menu bar includes Arquivo, Editar, Localizar, Exibir, Projeto, Executar, Ferramentas, AStyle, Janela, and Ajuda. The toolbar contains various icons for file operations, editing, and execution. The project explorer on the left shows a folder named (globals). The main editor window displays the following C++ code:

```
1  #include <stdio.h>
2  void main()
3  {
4      int idade = 6;
5      double salario = 4.5;
6      printf("Valor de sua idade = %d", idade);
7      printf(" e enderco das idade = %d\n", &idade);
8      printf("Valor dos salario = %g", salario);
9      printf(" e endereco dos salario = %d\n",&salario);
10 }
11
```