Funções

SSC300- Linguagens de Programação e Aplicações

Profa Vânia de Oliveira Neves

ICMC/USP – São Carlos

Slides baseados no material gentilmente cedido pela
Profa Simone Senger Souza

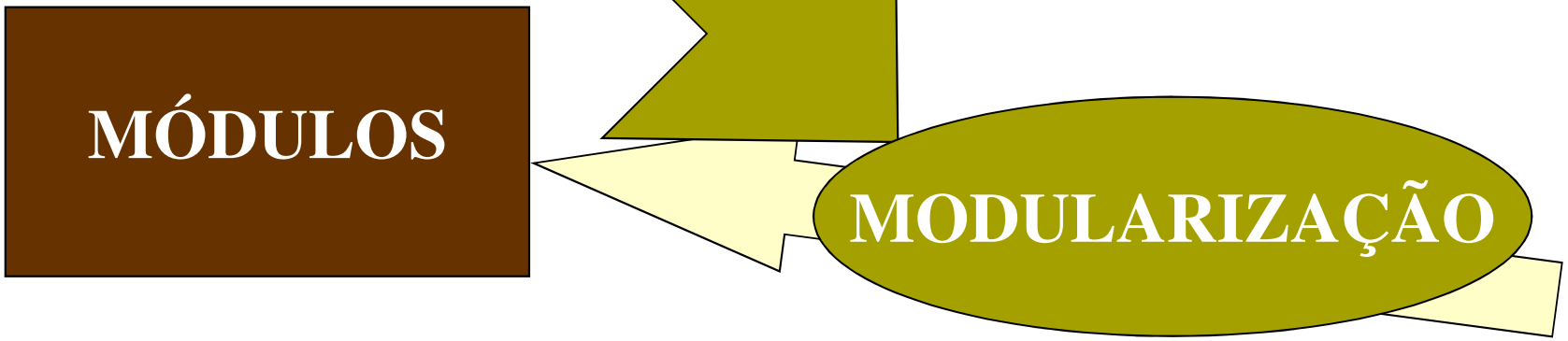
MODULARIZAÇÃO

Um problema complexo é melhor abordado se for dividido primeiramente em vários subproblemas

MODULARIZAÇÃO

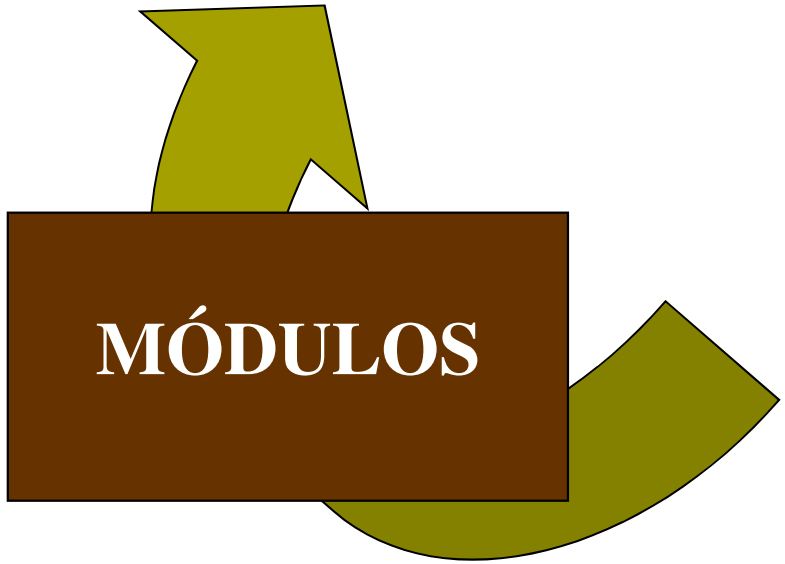
MODULARIZAÇÃO

Um problema complexo é melhor abordado se for dividido primeiramente em vários subproblemas



MODULARIZAÇÃO

A implementação desses Módulos é feita através de FUNÇÕES



Características das Funções

- Devem realizar uma única funcionalidade no programa.
  - possuem "código independente" e devem ser definidos separadamente no programa
    - Na linguagem C a definição das funções deve ocorrer antes da função que irá utiliza-lo
- São declarados no início do programa
  - Prototipação de funções

Estrutura de uma função

```
tipo da função nome da função(tipo
  var1, ..., tipo varn)
{
  declaração de variáveis locais
  corpo da função
  return(variável de retorno);
}
```

Prototipação da função

Declaração da função:

```
tipo da função nome da função(tipo1,
  ..., tipon);
```

Estrutura de uma função

- Retorna um único valor
- Na definição da função devem ser declarados:
  - o tipo de todos os parâmetros
  - o tipo do valor que a função retorna
  - todas as variáveis utilizadas internamente no subprograma (variáveis locais)
- Para utilizar a função no programa principal basta colocar seu nome (identificador) e os parâmetros reais.

DEFINIÇÃO DE FUNÇÃO

Exemplo

- Dados dois números N e K, calcular a Combinação

$$C_{N,K} = \frac{N!}{K!(N-K)!}$$

- Com a definição de uma função **fat (X)** que calcula o fatorial de um dado X, o cálculo da Combinação fica:

CNK ← **fat (**N**)** / (**fat (**K**)** * **fat (**N-K**)**)

Escopo de variáveis

- Significa a visibilidade de uma variável perante os diversos subprogramas integrantes do programa (ou algoritmo)
  - Variável global: declarada no início do algoritmo ou programa (fora dos subprogramas e programa principal) e é visível por todos.
  - Variável local: declarada dentro de um subprograma e somente visível dentro do mesmo.

```
void troca(int a, int b)
{
  int aux;
  aux = a;
  a = b;
  b = aux;
}
void main()
{
  int  a, b;
  a = 10; b = 20;
  troca(a,b);
  printf("a = %d e b = %d", a,b);
}
```

```
void troca(int a, int b)
{
  int aux;
  aux = a;
  a = b;
  b = aux;
}
void main()
{
  int  a, b;
  a = 10; b = 20;
  troca(a,b);
  printf("a = %d e b = %d", a,b);
}
```

a, b e aux são variáveis locais de troca()

a e b são variáveis locais do alg. princ.

```
void troca(int a, int b)
{
  int aux;
  aux = a;
  a = b;
  b = aux;
}
void main()
{
  int  a, b;
  a = 10; b = 20;
  troca(a,b);
  printf("a = %d e b = %d", a,b);
}
```

a, b e aux são variáveis locais de troca()

Não realiza a troca!

Não é possível retornar dois valores

a e b são variáveis locais do alg. princ.

```
// variáveis globais
int  a, b;
void troca(void)
{
  int aux;
  aux = a;
  a = b;
  b = aux;
}
void main(void)
{
  a = 10; b = 20;
  troca();
   printf("a = %d e b = %d", a,b);
}
```

a e b são variáveis globais

aux é a única variável local

Realiza a troca com o uso de variável global

```
 // variáveis globais
int  a, b;
void troca(void)
{
  int aux;
  aux = a;
  a = b;
  b = aux;
}
void main(void)
{
  a = 10; b = 20;
  troca();
   printf("a = %d e b = %d", a,b);
}
```

Não é a melhor solução para o problema!!

Escopo de variáveis

- A variável global existe durante toda a execução do programa
  - Interessante quando a mesma é utilizada por várias funções
  - Muitas variáveis globais podem prejudicar a execução do programa (overflow)
  - Os parâmetros enviados para as funções ajudam no entendimento da finalidade da função
    - O uso de variável global deve ser analisado.

Como Transformar um Programa em Subprogramas

Programa para verificar se um número é primo


```
void main ()
{
int n, c, i;
   scanf("%d", &n);
   c = 1;  //true
   i = 2;  //divisor do número
   while ( (i < n)  &&  c )
       if (n % i == 0)  //resto da divisão for zero
            c = 0;
       else
            i++;

if (c)
       printf("O Numero e Primo" );
 else
       printf("O Numero nao e Primo");
 }
```

Procura por um divisor diferente de 1 e do próprio número

Programa para verificar se um número é primo


```
void main ()
{
int n, c, i;
   scanf("%d", &n);
   c = 1;  //true
   i = 2;  //divisor do número
   while ( (i < n)  &&  c )
      if (n % i == 0)  //resto da divisão for zero
          c = 0;
      else
          i++;

if (c)
      printf("O Numero e Primo" );
 else
      printf("O Numero nao e Primo");
 }
```

Este trecho do programa verifica se o número é primo

OBJETIVO PRINCIPAL

Programa para verificar se um número é primo


```
void main ()
{
int n, c, i;
   scanf("%d", &n);
   c = 1;  //true
   i = 2;  //divisor do número
   while ( (i < n)  &&  c )
      if (n % i == 0)  //resto da divisão for zero
         c = 0;
      else
         i++;

if (c)
      printf("O Numero e Primo" );
 else
      printf("O Numero nao e Primo");
 }
```

Corpo da função

n : parâmetro de entrada

c: saída

Programa para verificar se um número é *primo*


```
void main ()
{
int n, c, i;
   scanf("%d", &n);
   c = 1;  //true
   i = 2;  //divisor do número
   while ( (i < n)  &&  c )
      if (n % i == 0)  //resto da divisão for zero
         c = 0;
      else
         i++;

if (c)
      printf("O Numero e Primo" );
 else
      printf("O Numero nao e Primo");
 }
```

A leitura de **n** fica no **programa principal**

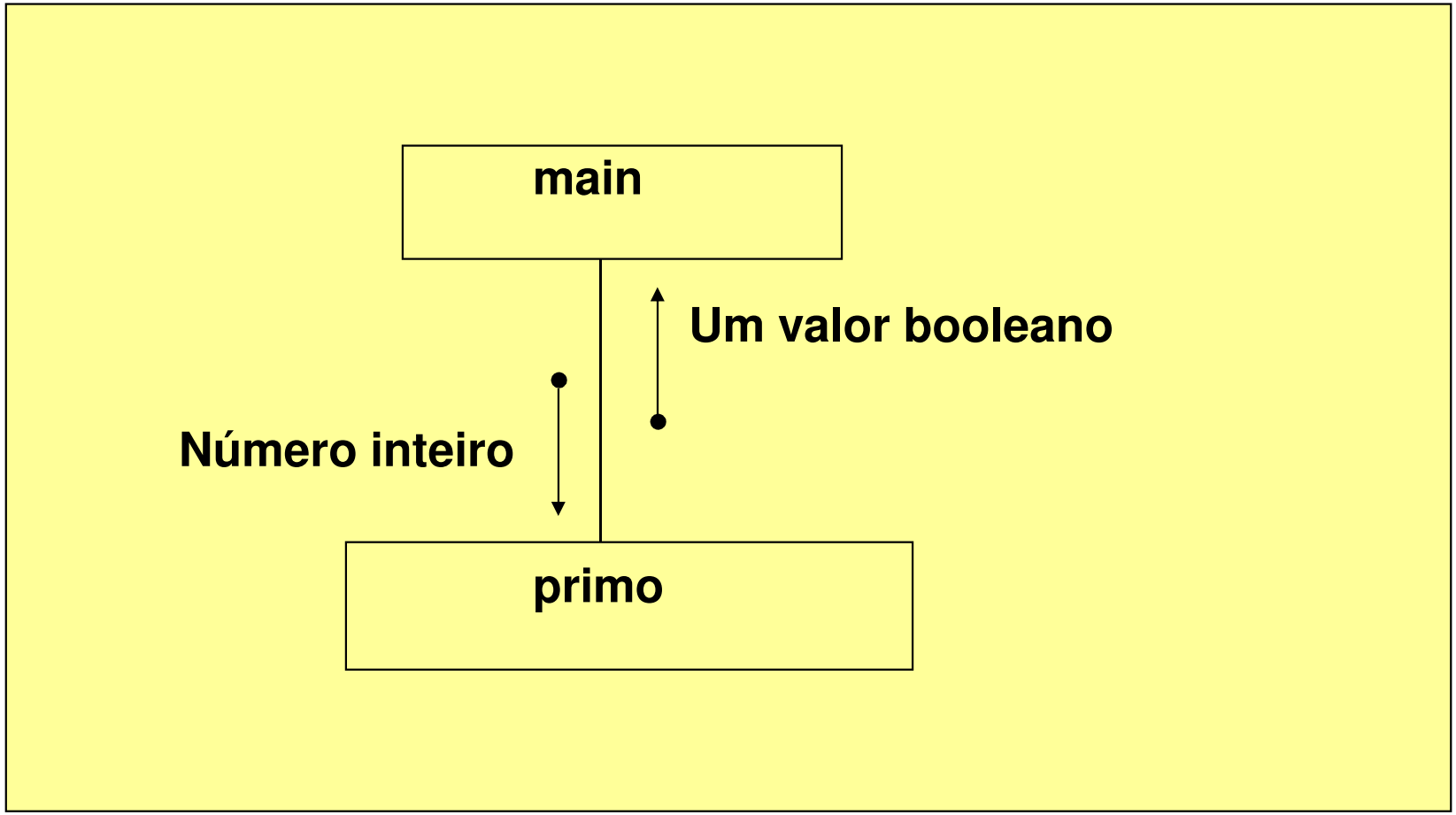
As **mensagens** de saída ficam no **programa principal**

Programa para verificar se um número é *primo*


```
int primo(int n)
{
int c, i;
  c = 1;  //true
   i = 2;  //divisor do número
   while ( (i < n)  &&  c )
       if (n % i == 0)  //resto da divisão for zero
            c = 0;
       else
            i++;
return(c);
}
```

```
void main ()
{
   int n, c, i;
   scanf("%d", &n);
   if (primo(n))
       printf("O Numero e Primo" );
    else
       printf("O Numero nao e Primo");
 }
```

Arquitetura



Exercício:

- Faça uma função que recebe três números inteiros e positivos como parâmetros: *a*, *b* e *c*, onde *a* é maior que 1. A função deve retornar a soma de todos os números no intervalo entre *b* e *c* que sejam divisíveis por *a* (inclusive *b* e *c*). Faça um programa que lê *a*, *b* e *c* e mostra o resultado obtido pela função descrita.

Resolução em C:

```
int soma(int a, int b, int c)
{
 int s = 0;
 while (b <= c)
 {
      if (b%a == 0)  //eh divisível por a
      {
        s = s + b;
        printf("o numero %d foi adicionado a soma\n", b);
        getch();
      }
      b++;
}
      return(s);
}
```

```
int main(int argc, char *argv[])
{
 int b, c, a;
 printf("entre com dois valores positivos para o intervalo:");
 scanf("%d%d",&b,&c);
 while(b<0 || c<0 || b>c)
 {
            printf("valores invalidos, entre novamente...\n");
            printf("entre com os valores do intervalo:");
            scanf("%d%d",&b,&c);
 }

 printf("Entre com um valor para o divisor:");
 scanf("%d", &a);
 while (a <= 1 || a > c)
{
       printf("valor invalido, entre novamente...\n");
       printf("Entre com um valor para o divisor:");
       scanf("%d%d",&a);
}
printf ("a soma dos numeros no intervalo eh: %d\n", soma(a,b,c));
  system("PAUSE");
  return 0;
}
```

Declaração de Funções (prototipação)

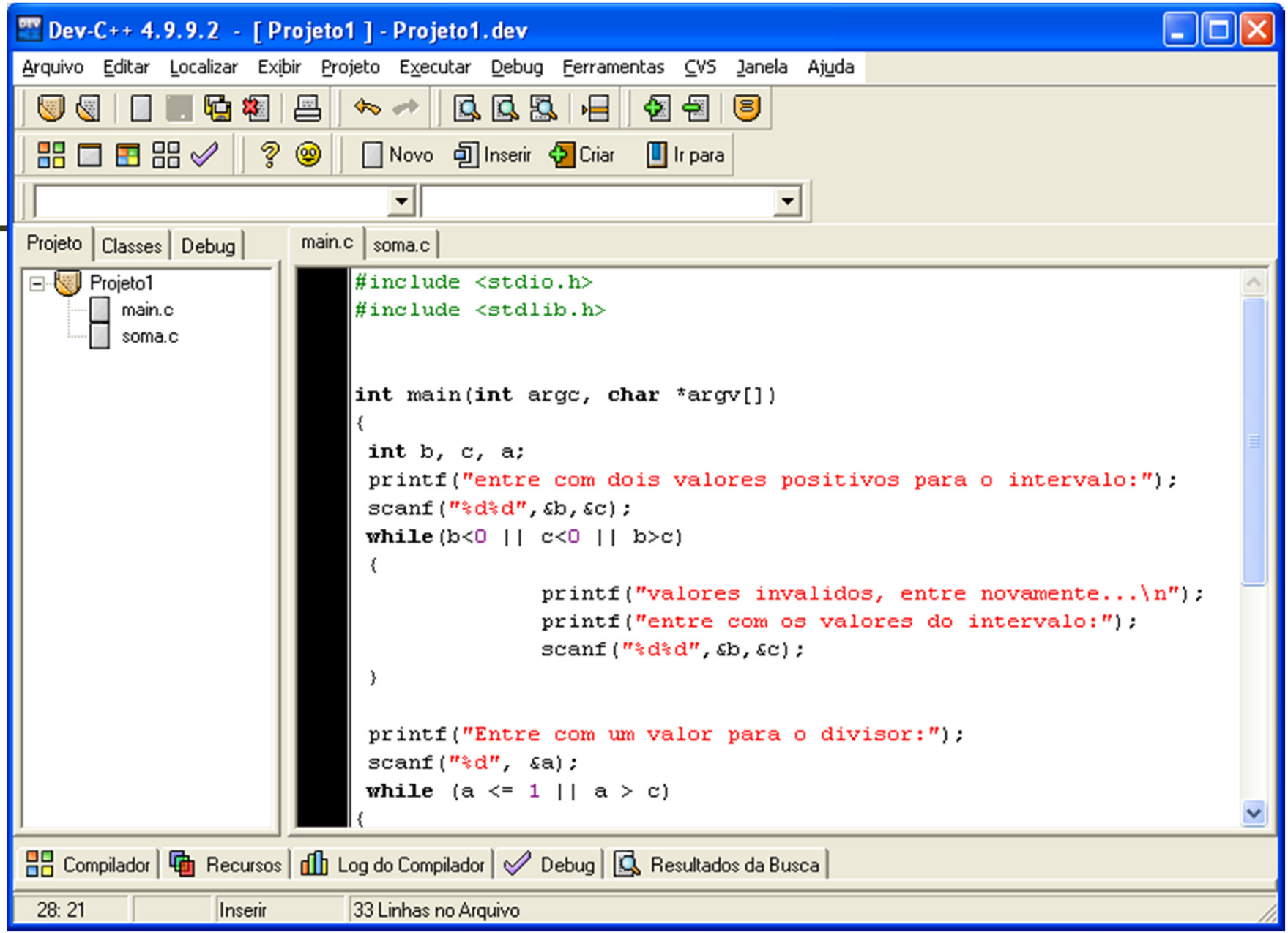
- Recurso para melhorar a legibilidade dos programas
- Flexibiliza a localização da descrição das funções no programa

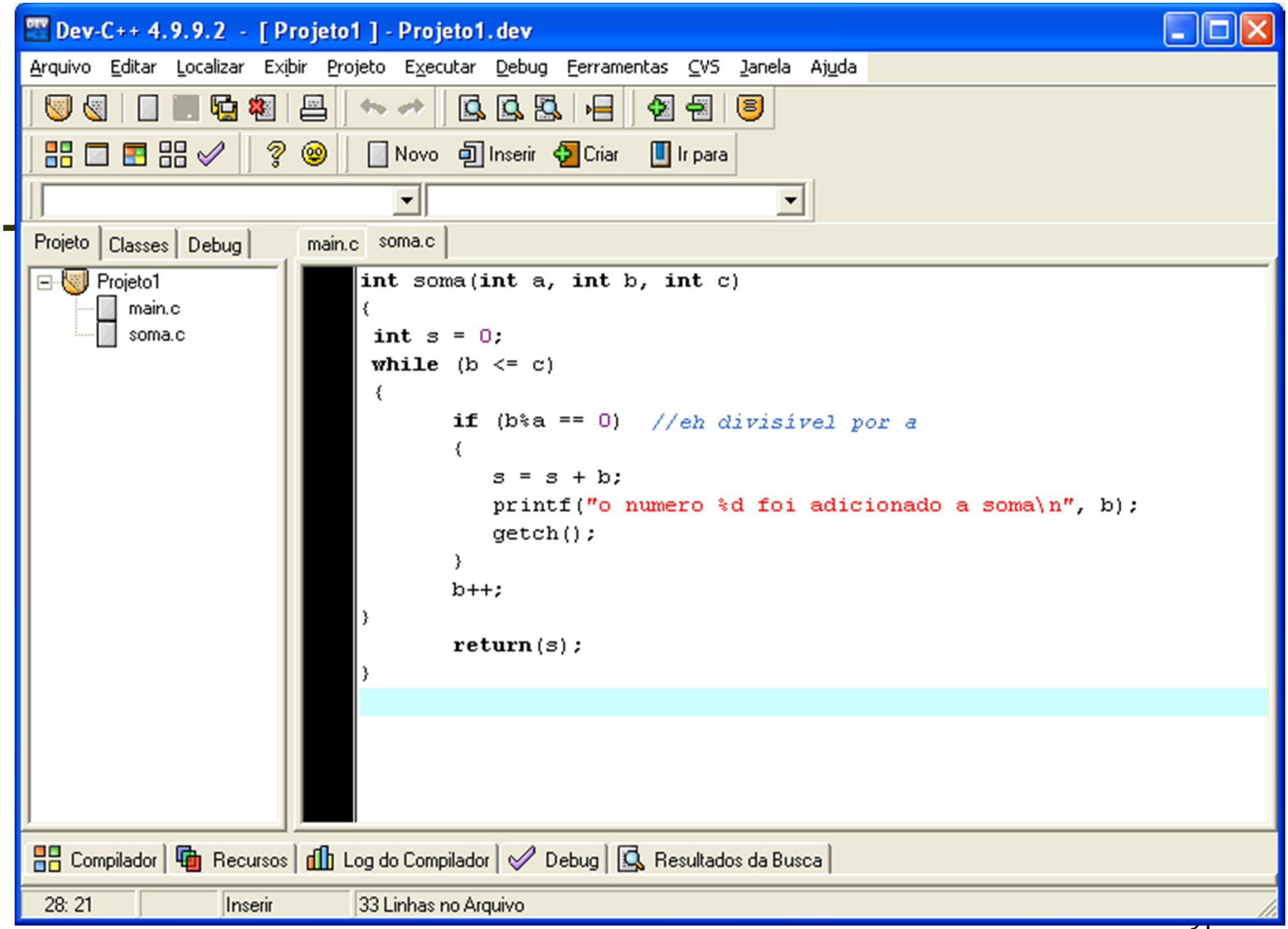
Exemplo:

```
#include <stdio.h>
#include <stdlib.h>

int soma(int , int , int ); //declaração da função (prototipação)

int main(int argc, char *argv[])
{
 int b, c, a;
....

}
int soma(int a, int b, int c)
{
 int s = 0;
 while (b <= c)
 {
      if (b%a == 0)  //eh divisível por a
              s = s + b;
      b++;
}
      return(s);
}
```

Funções

- Divisão das funções em arquivos separados


Dev-C++ 4.9.9.2 - [Projeto1] - Projeto1.dev
Arquivo Editar Localizar Exibir Projeto Executar Debug Ferramentas CVS Janela Ajuda
Novo Inserir Criar Ir para
Projeto Classes Debug
Projeto1
main.c
soma.c
main.c soma.c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
int b, c, a;
printf("entre com dois valores positivos para o intervalo:");
scanf("%d%d",&b,&c);
while(b<0 || c<0 || b>c)
{
printf("valores invalidos, entre novamente...\n");
printf("entre com os valores do intervalo:");
scanf("%d%d",&b,&c);
}
printf("Entre com um valor para o divisor:");
scanf("%d", &a);
while (a <= 1 || a > c)
{
Compilador Recursos Log do Compilador Debug Resultados da Busca
28: 21 Inserir 33 Linhas no Arquivo


Dev-C++ 4.9.9.2 - [Projeto1] - Projeto1.dev
Arquivo Editar Localizar Exibir Projeto Executar Debug Ferramentas CVS Janela Ajuda
Novo Inserir Criar Ir para
Projeto Classes Debug
main.c soma.c
Projeto1
main.c
soma.c
int soma(int a, int b, int c)
{
int s = 0;
while (b <= c)
{
if (b%a == 0) //eh divisível por a
{
s = s + b;
printf("o numero %d foi adicionado a soma\n", b);
getch();
}
b++;
}
return(s);
}
Compilador Recursos Log do Compilador Debug Resultados da Busca
28: 21 Inserir 33 Linhas no Arquivo