Programa “ConsoleRPN”

Programa criado com o Visual Studio 2008
para efetuar operações algébricas entre números,
uma calculadora funcionando com console usando RPN

PUC-Rio – CIV 2802 - Sistemas Gráficos para Engenharia
Luiz Fernando Martha & André Maués Brabo Pereira

Objetivos da Aula:

Apresentar os conceitos de programação da linguagem C/C++ utilizando um exemplo de uma calculadora RPN (Reversed Polish Notation)

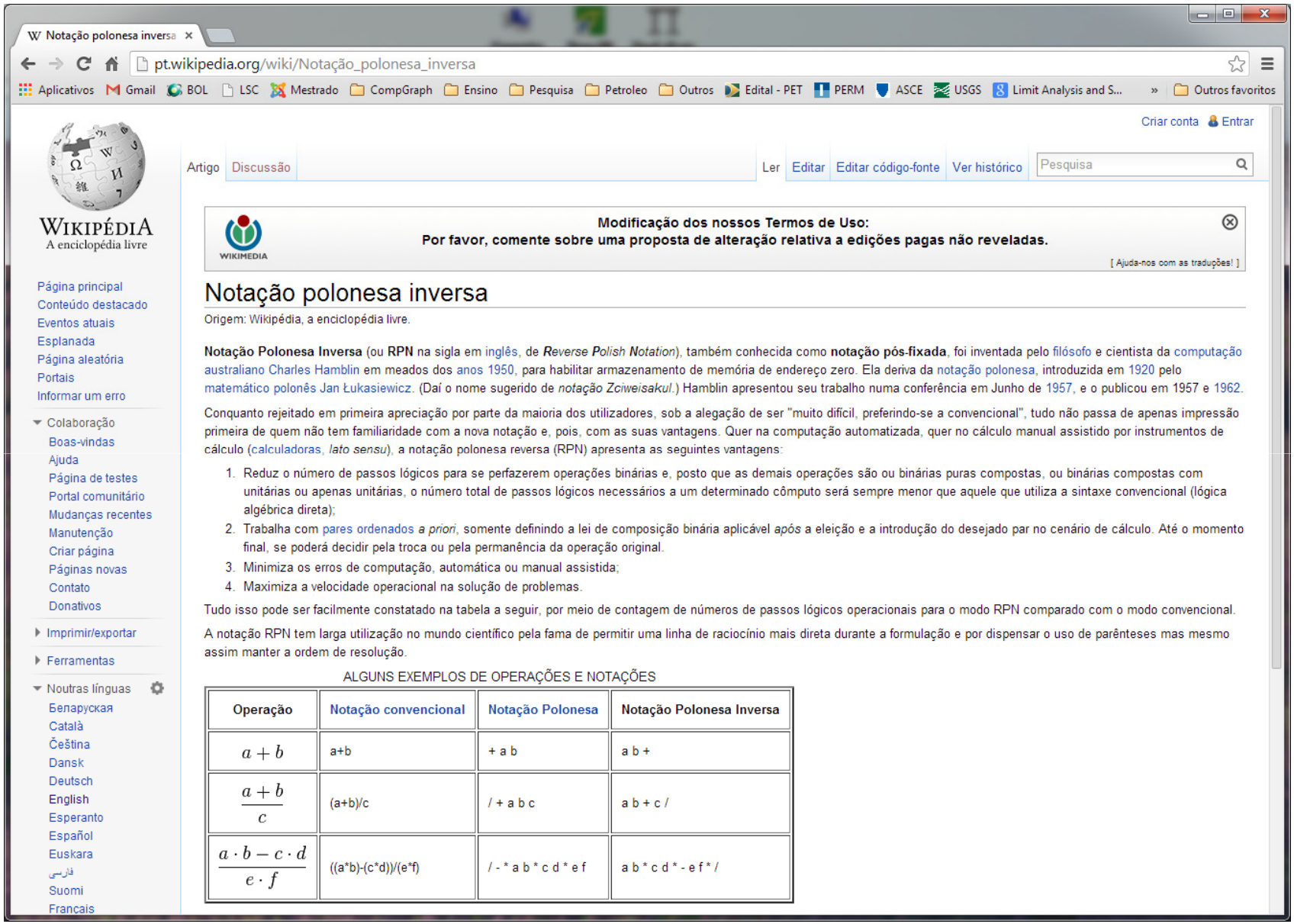
Conteúdo/Assuntos Abordados:

Introdução à linguagem de programação C/C++.
Introdução ao conceito de classes e objetos em C++.

Competências/Habilidades:

Conhecer os recursos básicos da linguagem C/C++.
Capacidade de criar um projeto console no Visual Studio 2008.
Capacidade de entendimento de algoritmos e estruturas de dados.
Entendimento do uso de funções e os mecanismos de passagem de parâmetros para funções na linguagem C/C++.
O aluno deverá ser capaz de desenvolver aplicativos práticos.

O que significa uma calculadora RPN (Reserved Polish Notation)?



Notação polonesa inversa

pt.wikipedia.org/wiki/Notação_polonesa_inversa

Aplicativos Gmail BOL LSC Mestrado CompGraph Ensino Pesquisa Petroleo Outros Edital - PET PERM ASCE USGS Limit Analysis and S... Outros favoritos

Criar conta Entrar

Artigo Discussão Ler Editar Editar código-fonte Ver histórico Pesquisa

WIKIPÉDIA
A enciclopédia livre

Página principal
Conteúdo destacado
Eventos atuais
Esplanada
Página aleatória
Portais
Informar um erro

Colaboração
Boas-vindas
Ajuda
Página de testes
Portal comunitário
Mudanças recentes
Manutenção
Criar página
Páginas novas
Contato
Donativos

Imprimir/exportar

Ferramentas

Noutras línguas
Беларуская
Català
Čeština
Dansk
Deutsch
English
Esperanto
Español
Euskara
فارسی
Suomi
Français

WIKIMEDIA

Modificação dos nossos Termos de Uso:
Por favor, comente sobre uma proposta de alteração relativa a edições pagas não reveladas.

[Ajuda-nos com as traduções!]

Notação polonesa inversa

Origem: Wikipédia, a enciclopédia livre.

Notação Polonesa Inversa (ou **RPN** na sigla em inglês, de *Reverse Polish Notation*), também conhecida como **notação pós-fixada**, foi inventada pelo filósofo e cientista da computação australiano Charles Hamblin em meados dos anos 1950, para habilitar armazenamento de memória de endereço zero. Ela deriva da notação polonesa, introduzida em 1920 pelo matemático polonês Jan Łukasiewicz. (Daí o nome sugerido de *notação Zciweisakul*.) Hamblin apresentou seu trabalho numa conferência em Junho de 1957, e o publicou em 1957 e 1962.

Conquanto rejeitado em primeira apreciação por parte da maioria dos utilizadores, sob a alegação de ser "muito difícil, preferindo-se a convencional", tudo não passa de apenas impressão primeira de quem não tem familiaridade com a nova notação e, pois, com as suas vantagens. Quer na computação automatizada, quer no cálculo manual assistido por instrumentos de cálculo (calculadoras, *lato sensu*), a notação polonesa reversa (RPN) apresenta as seguintes vantagens:

1. Reduz o número de passos lógicos para se perfazerem operações binárias e, posto que as demais operações são ou binárias puras compostas, ou binárias compostas com unitárias ou apenas unitárias, o número total de passos lógicos necessários a um determinado cômputo será sempre menor que aquele que utiliza a sintaxe convencional (lógica algébrica direta);
2. Trabalha com pares ordenados *a priori*, somente definindo a lei de composição binária aplicável *após* a eleição e a introdução do desejado par no cenário de cálculo. Até o momento final, se poderá decidir pela troca ou pela permanência da operação original.
3. Minimiza os erros de computação, automática ou manual assistida;
4. Maximiza a velocidade operacional na solução de problemas.

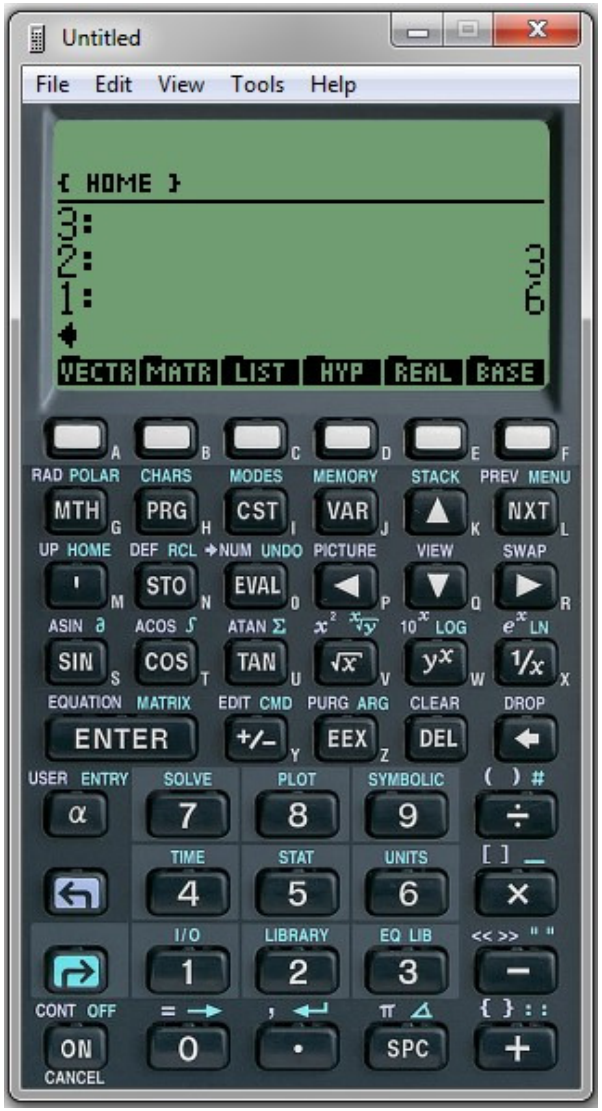
Tudo isso pode ser facilmente constatado na tabela a seguir, por meio de contagem de números de passos lógicos operacionais para o modo RPN comparado com o modo convencional.

A notação RPN tem larga utilização no mundo científico pela fama de permitir uma linha de raciocínio mais direta durante a formulação e por dispensar o uso de parênteses mas mesmo assim manter a ordem de resolução.

ALGUNS EXEMPLOS DE OPERAÇÕES E NOTAÇÕES

Operação	Notação convencional	Notação Polonesa	Notação Polonesa Inversa
$a + b$	a+b	+ a b	a b +
$\frac{a+b}{c}$	(a+b)/c	/ + a b c	a b + c /
$\frac{a \cdot b - c \cdot d}{e \cdot f}$	((a*b)-(c*d))/(e*f)	/ - * a b * c d * e f	a b * c d * - e f * /

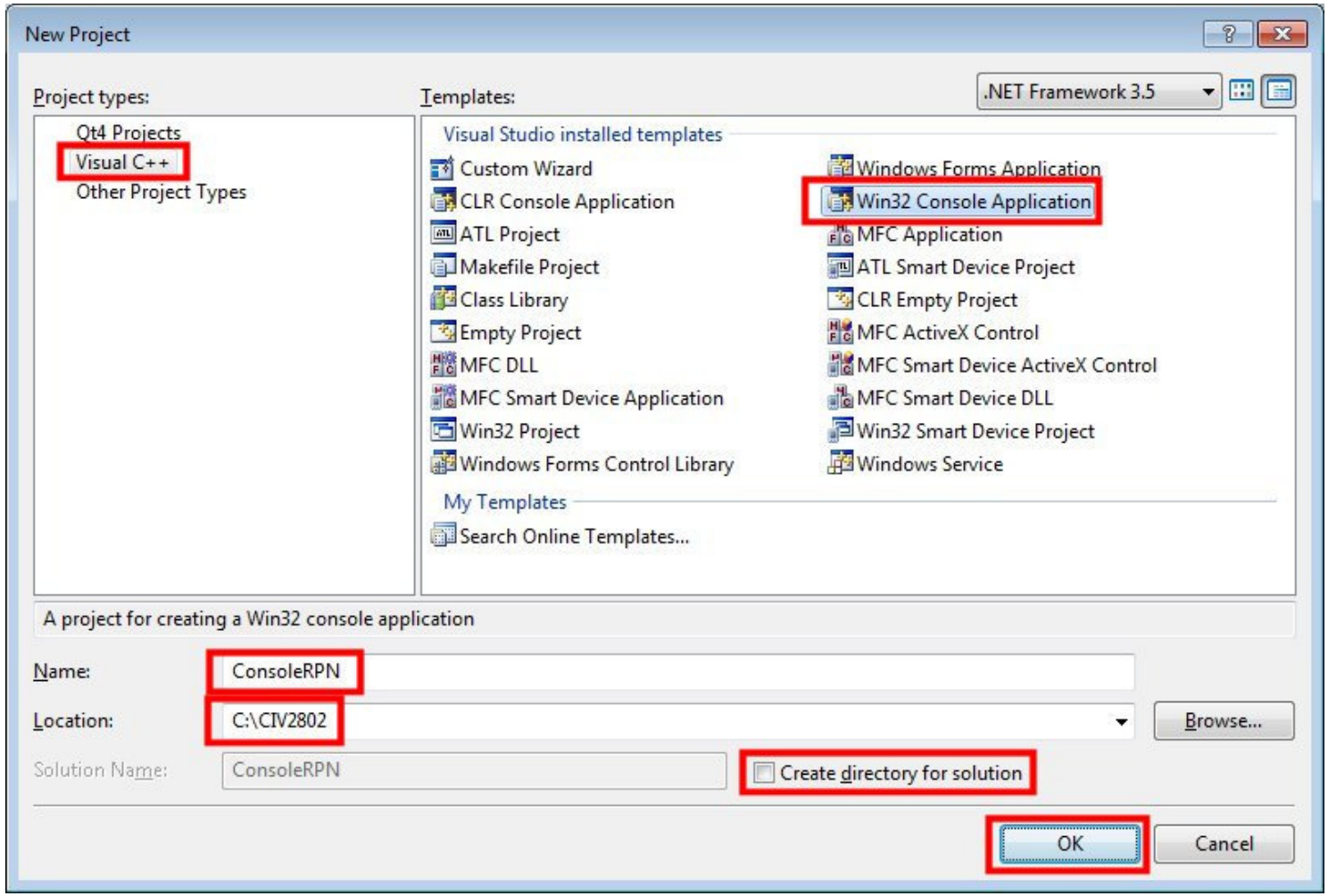
Exemplo clássico de calculadora que utiliza a Notação Polonesa Reversa (RPN)



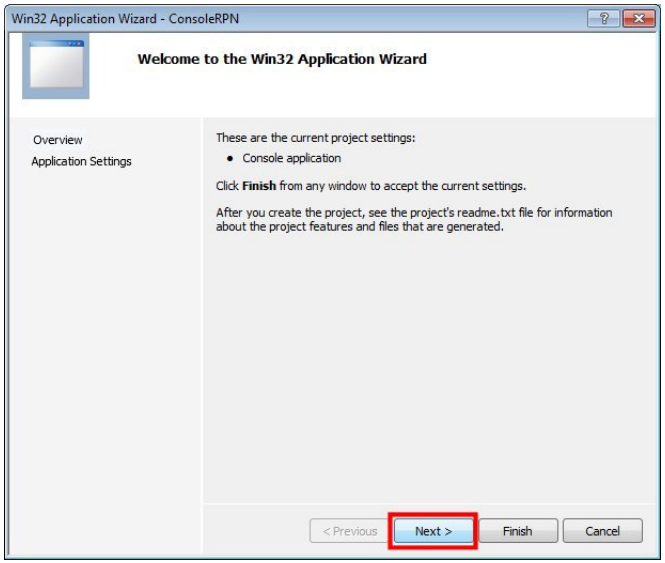
Como criar nossa própria calculadora utilizando uma linguagem de programação e uma interface gráfica?

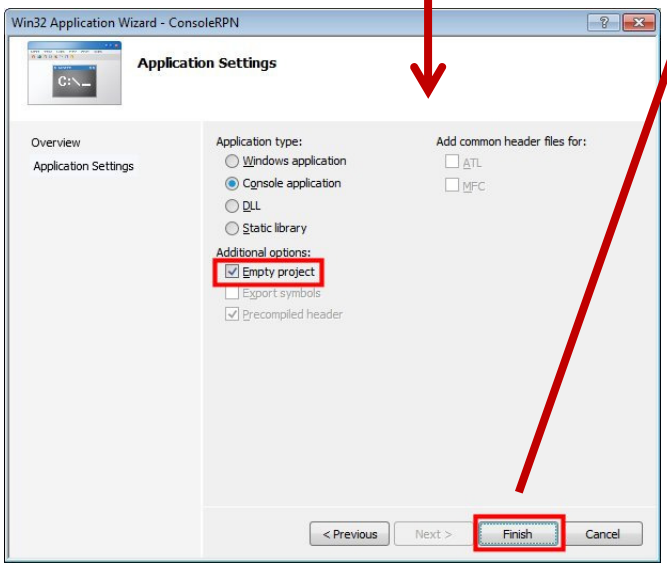
Criação de um novo projeto no Visual Studio 2008 do tipo Console

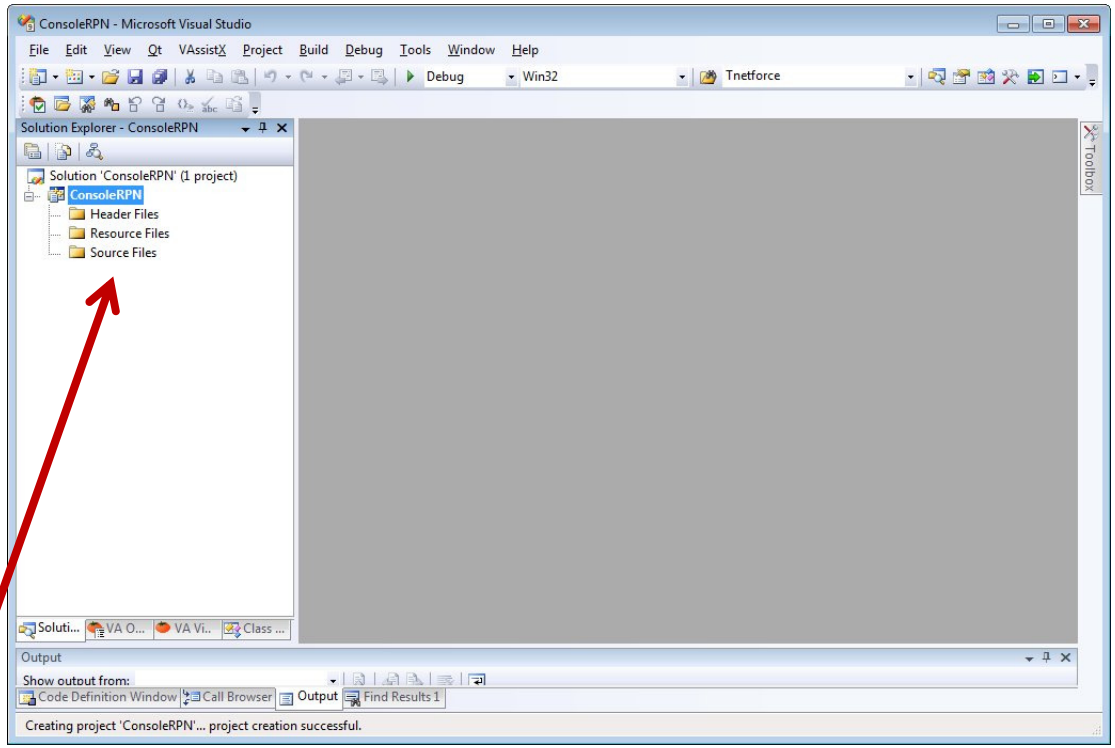
Vamos começar programando uma calculadora sem interface gráfica.



Wizard para criação de um projeto do tipo Console “Vazio”

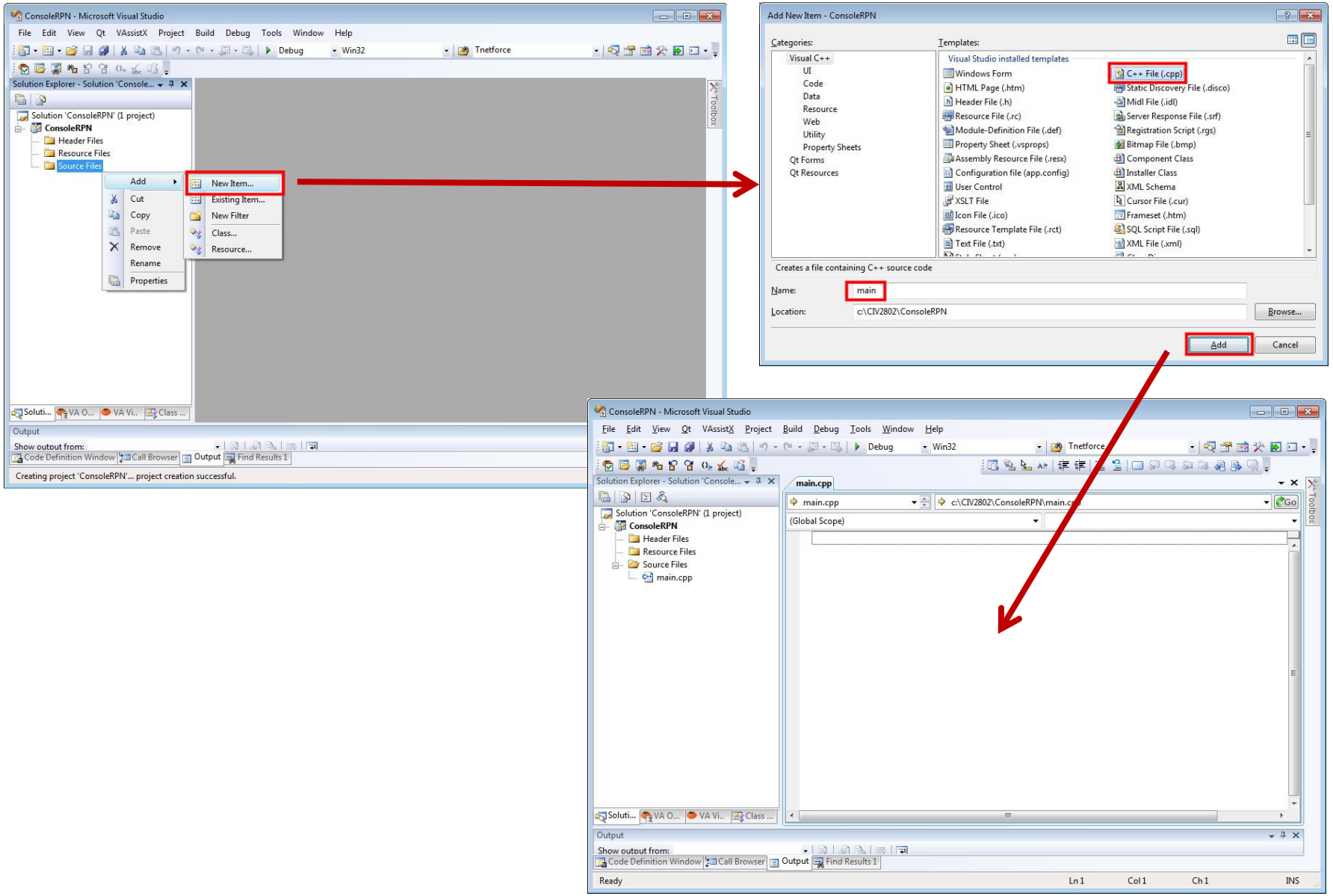






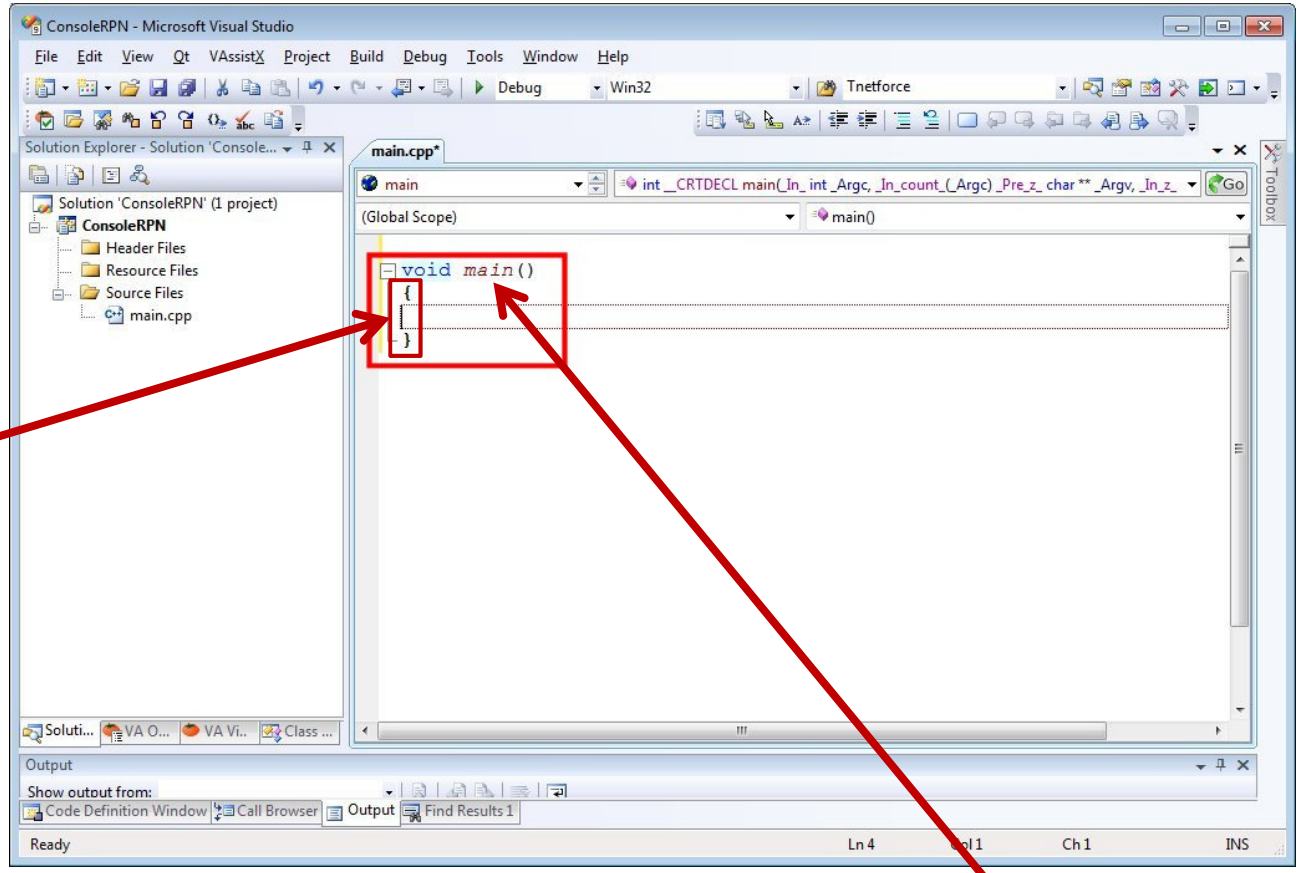
Como começar desenvolver
um programa do zero?

Criação de um arquivo novo, com o nome “main” e extensão “.cpp”



Implementação do arquivo “main.cpp”

Estrutura básica de um programa em C



Um par de chaves define um bloco de código

A execução do programa inicia pela chamada da função **`main`**.

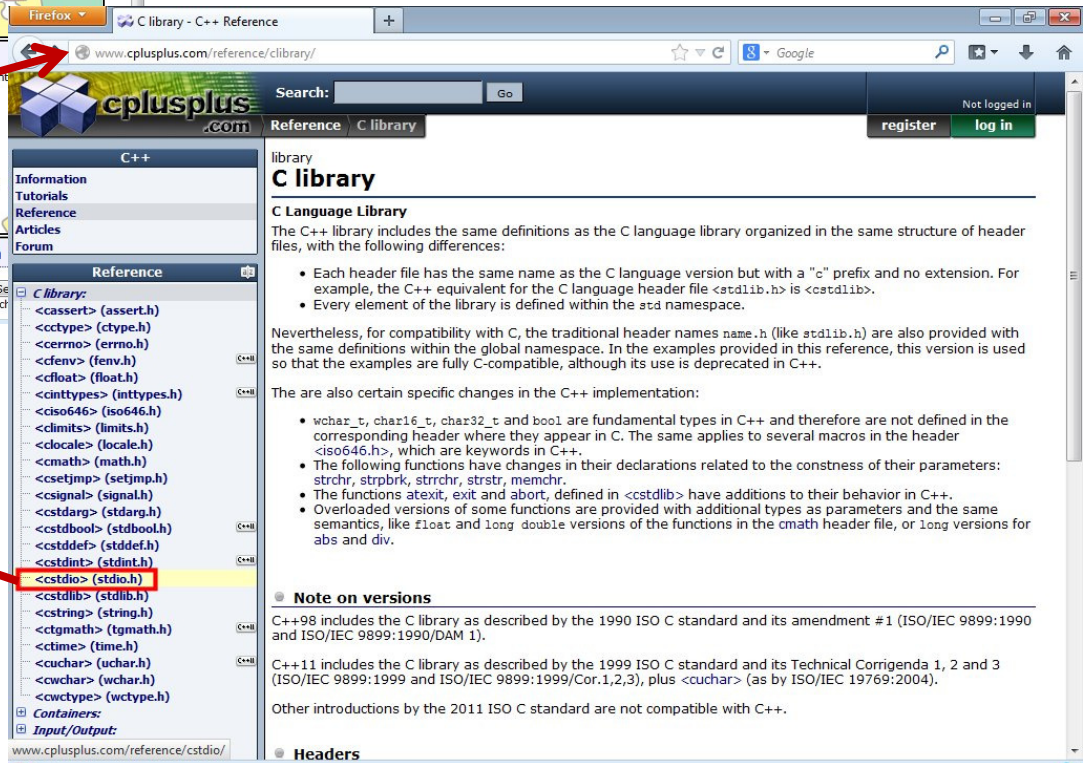
A função main pode ter diferentes assinaturas:

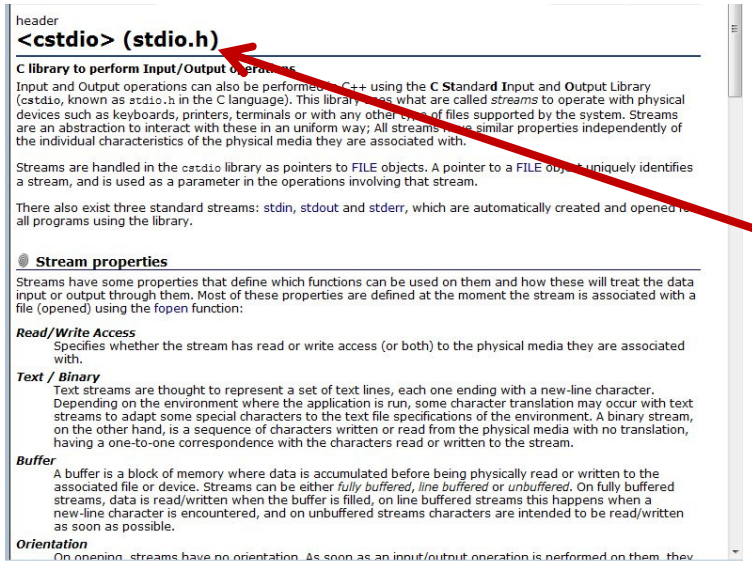
- **`main()`**
- **`int main(int argc, char **argv)`**
- **`int main(int argc, char **argv, char **env)`**

Todo programa em C contém pelo menos uma função:
`main`
Nesse caso **`void main()`** não requer parâmetros e não retorna parâmetros.

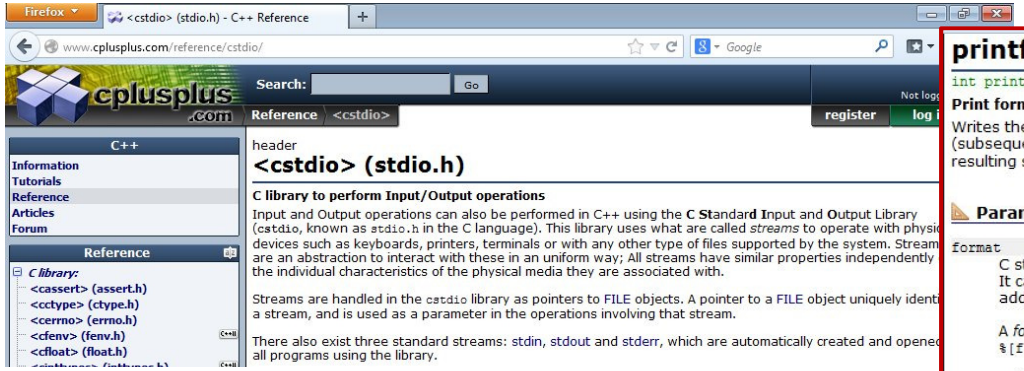
Documentação e referências para programar em C e C++

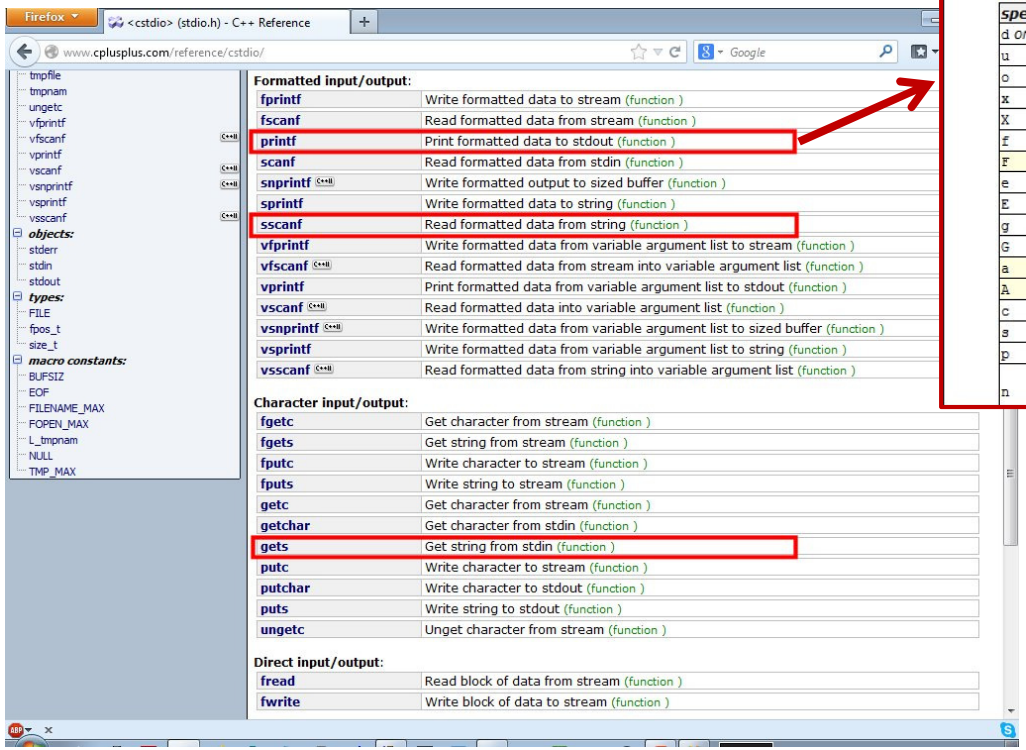
Como imprimir dados na janela de um programa console?





Tratamento de entrada e saída (io) em um programa em C





printf

<cstdio>

```
int printf ( const char * format, ... );
```

Print formatted data to stdout

Writes the C string pointed by *format* to the standard output (stdout). If *format* includes *format specifiers* (subsequences beginning with %), the additional arguments following *format* are formatted and inserted in the resulting string replacing their respective specifiers.

Parameters

`format`

C string that contains the text to be written to stdout.
It can optionally contain embedded *format specifiers* that are replaced by the values specified in subsequent additional arguments and formatted as requested.

A *format specifier* follows this prototype: [see compatibility note below]
`%[flags][width][.precision][length]specifier`

Where the *specifier character* at the end is the most significant component, since it defines the type and the interpretation of its corresponding argument:

specifier	Output	Example
d *or* i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65
a	Hexadecimal floating point, lowercase	-0xc.90fep-2
A	Hexadecimal floating point, uppercase	-0XC.90FEP-2
c	Character	a
s	String of characters	sample
p	Pointer address	b8000000
n		

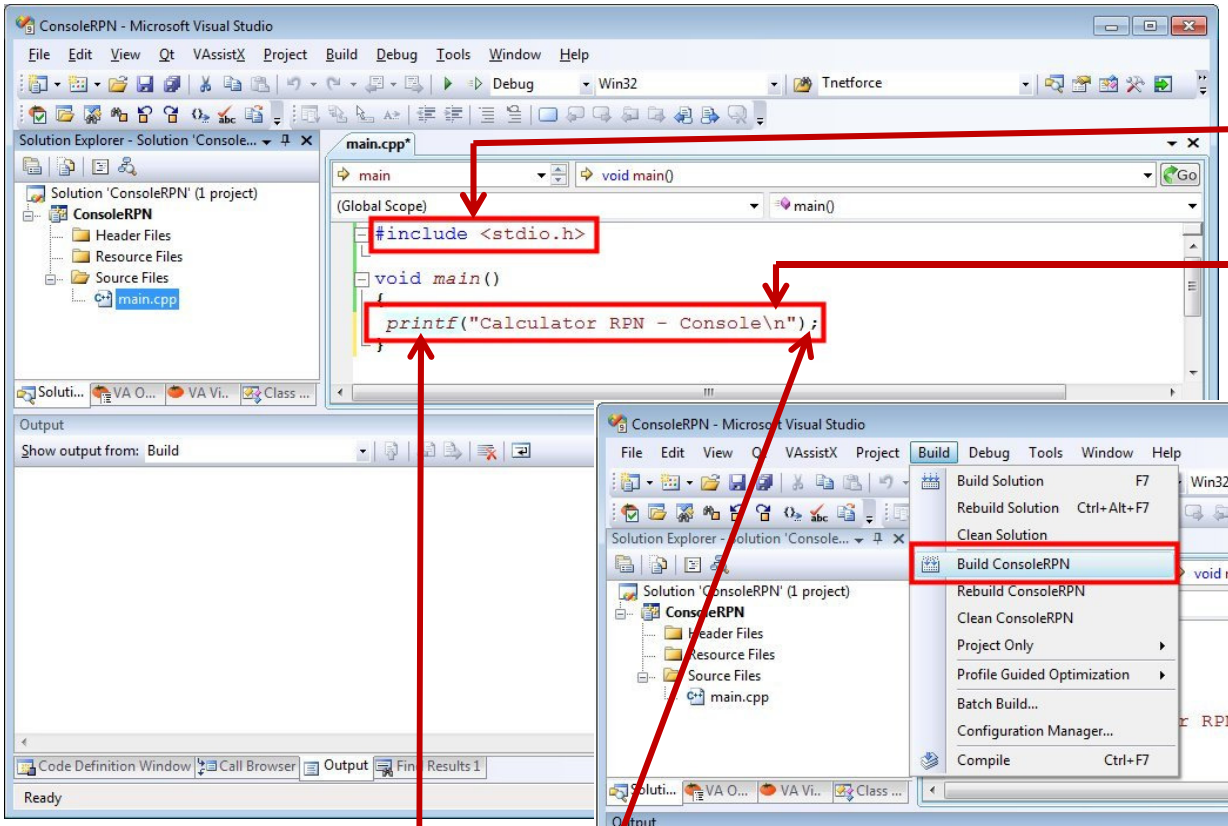
Example

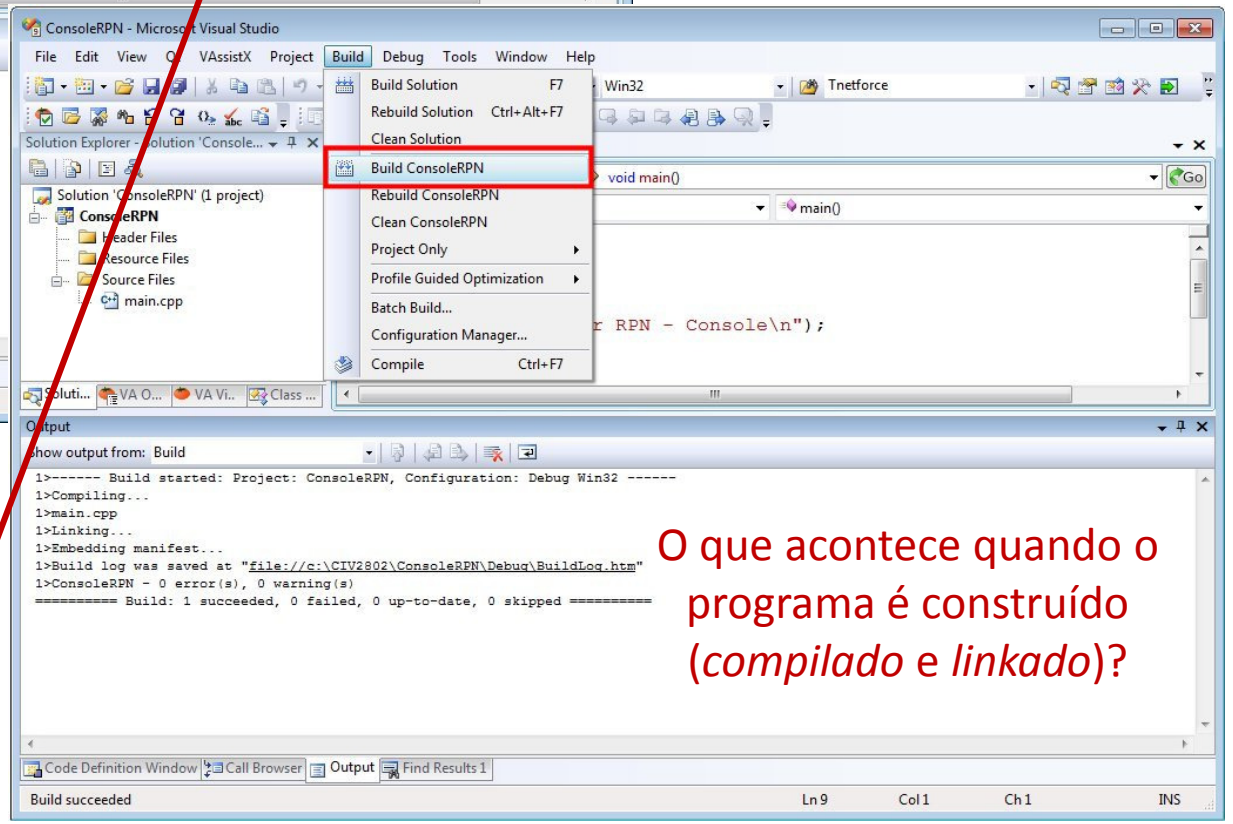
```
/* printf example */
#include <stdio.h>

int main()
{
   printf ("Characters: %c %c \n", 'a', 65);
   printf ("Decimals: %d %ld\n", 1977, 650000L);
   printf ("Preceding with blanks: %10d \n", 1977);
   printf ("Preceding with zeros: %010d \n", 1977);
   printf ("Some different radices: %d %x %o %#x %#o \n", 100, 100, 100, 100, 100);
   printf ("floats: %4.2f %+.0e %E \n", 3.1416, 3.1416, 3.1416);
   printf ("Width trick: %*d \n", 5, 10);
   printf ("%s \n", "A string");
   return 0;
}
```

Output:

```
Characters: a A
Decimals: 1977 650000
Preceding with blanks:       1977
Preceding with zeros: 0000001977
Some different radices: 100 64 144 0x64 0144
floats: 3.14 +3e+000 3.141600E+000
Width trick:    10
A string
```

Imprimindo mensagem no console usando a biblioteca C padrão I/O





Informa ao compilador que as funções de I/O padrão serão usadas.

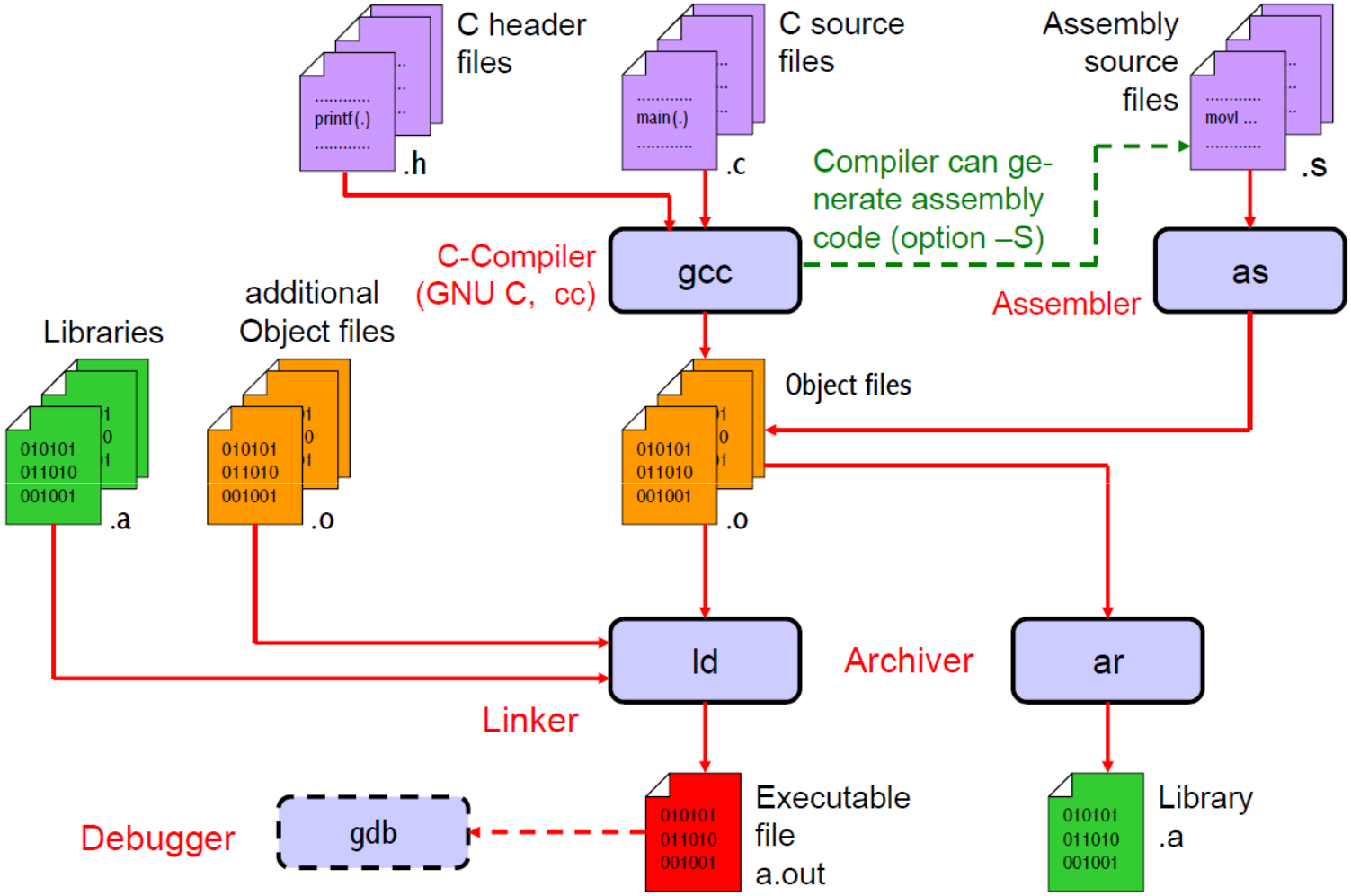
`\n` produz uma nova linha no console.

Escreve uma *string* na saída padrão do console. **`printf`** é uma função de C que é parte da biblioteca C padrão.

Toda declaração termina com um ponto e vírgula, exceto para a chave que fecha um bloco (**}**).

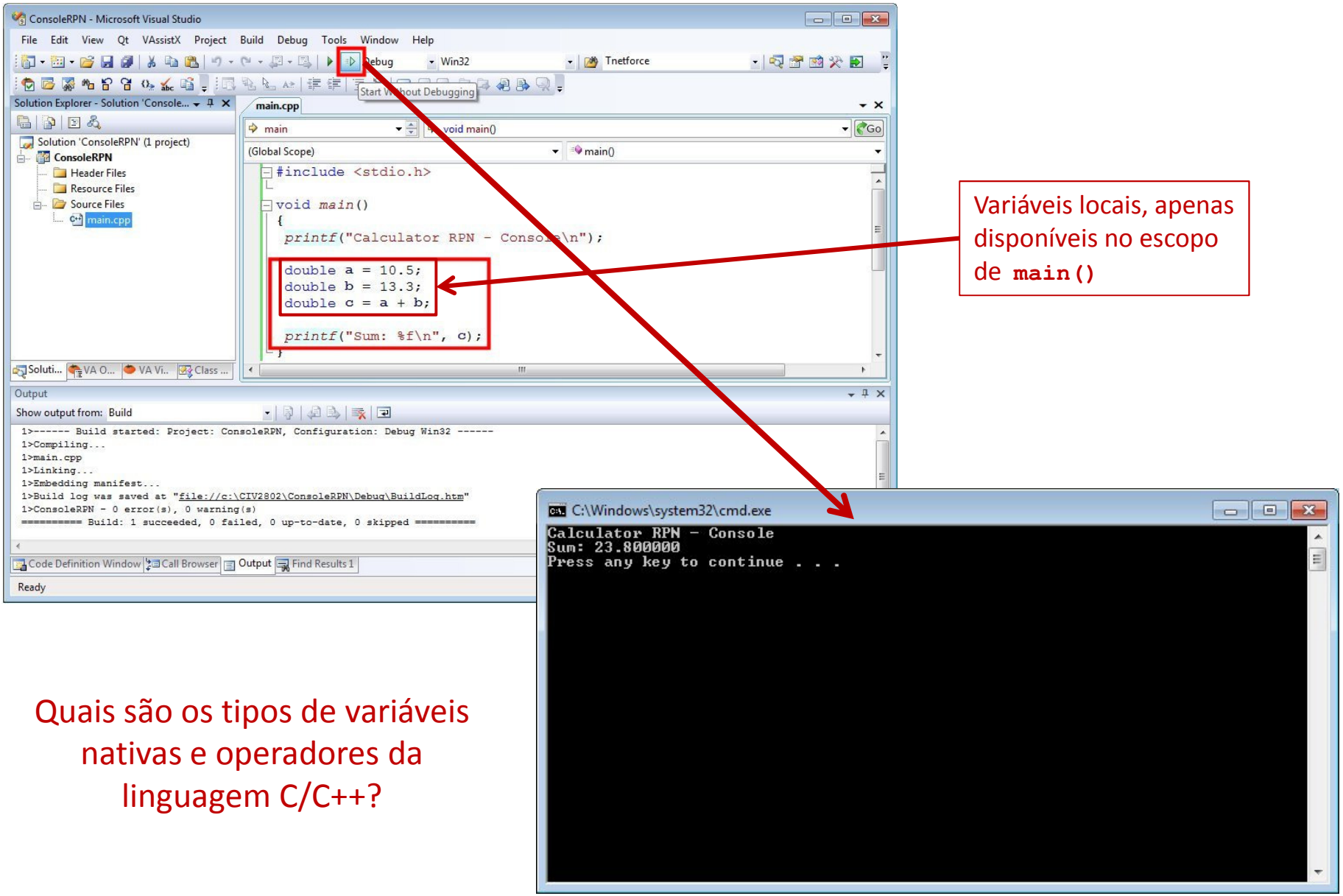
O que acontece quando o programa é construído (*compilado* e *linkado*)?

De códigos em C para um binário executável



Fonte: René Müller, Introduction to the C-Language and Programming Environment, Winter Semester 2005/06

Imprimindo no console (saída) o resultado da soma de dois números



Quais são os tipos de variáveis nativas e operadores da linguagem C/C++?

C data types

- **Four basic data types**
  - **`char`**: character
  - **`int`**: integer
  - **`float`**: real or floating point
  - **`double`**: double precision float
- **Four modifiers**
  - **`signed`**
  - **`unsigned`**
  - **`long`**
  - **`short`**
- **Four storage classes**
  - **`auto`**: variable is not required outside its block (the default)
  - **`register`**: the variable will be allocated on a CPU register
  - **`static`**: allows a local variable to retain its previous value upon reentry
  - **`extern`**: global variable declared in another file
- **Additionally, C supports**
  - the null data type: **`void`**
  - Any user-defined types

Type	Width (bits)	Minimum range
`char`	8	-127 to 127
`unsigned char`	8	0 to 255
`signed char`	8	-127 to 127
`int`	16	-32,767 to 32,767
`unsigned int`	16	0 to 65,535
`signed int`	16	Same as int
`short int`	16	Same as int
`unsigned short int`	8	0 to 65,535
`signed short int`	8	Same as short int
`long int`	32	-2,147,483,647 to 2,147,483,647
`signed long int`	32	--2,147,483,647 to 2,147,483,647
`unsigned long int`	32	0 to 4,294,967,295
`float`	32	Six-digit precision
`double`	64	Ten-digit precision
`long double`	128	Ten-digit precision

Fonte: Ricardo Gutierrez-Osuna, *Microprocessor-based System Design*, Wright State University.

C operators

Type	Operator	Action
Arithmetic	-	Subtraction
	+	Adittion
	*	Multiplication
	/	Division
	%	Modulus
	--	Decrement (by 1)
	++	Increment (by 1)
	+=	Increment (a+=b means a=a+b)
	-=	Decrement (a-=b means a=a-b)
Relational	>	Greater than
	>=	Greater than or equal to
	<	Less than
	<=	Less than or equal to
	==	Equal to
	!=	Different from
Logic	&&	AND
	||	OR
	!	NOT
Bit-wise	&	AND
	|	OR
	^	XOR
	~	NOT
	>>	Right shift
	<<	Left shitf
Miscellaneus	?	Ternary (y=x>9?100:200)
	& and *	Pointer operators
	sizeof	Width of a datatype (in bytes)
	. and ->	Acceess to structures
	[]	Access to arrays

Precedence	Operator
Most	( ) [ ] -> .
	! ~ ++ -- - (cast) * & sizeof
	/ %
	<< >>
	< <= > >=
	== !=
	&
	|
	&&
	||
	?
	= += -= *= /=
Least	,

Fonte: Ricardo Gutierrez-Osuna, *Microprocessor-based System Design*, Wright State University.

Variable declaration and scope

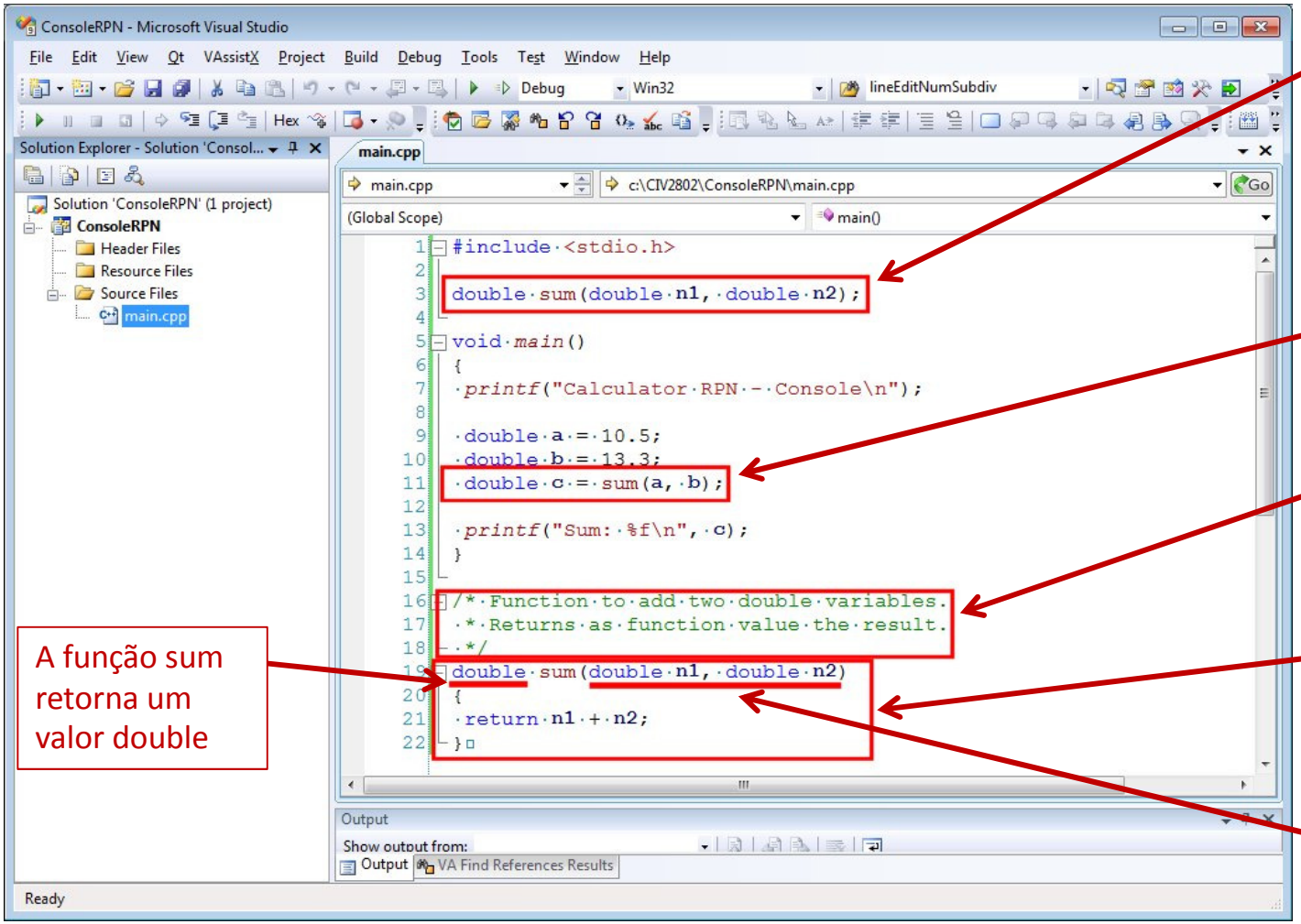
- **Variables MUST be declared before they are used**
  - Any declaration MUST precede the first statement in a block
- **Variables declared inside a block are local to that block**
  - They cannot be accessed from outside the block
- **Variables can be initialized when they are declared of afterwards**

```
int i;                  /* Integer i is global to the entire program
                           and is visible to everything from this point */
void function_1(void)   /* A function with no parameters */
   {
     int k;             /* Integer k is local to function_1 */
       {
         int q;         /* Integer q exists only in this block */
         int j;         /* Integer j is local and not the same as j in main */

       }
   }
void main(void)
  {
     int j;             /* Integer j is local to this block within function main */

  }                     /* This is the point at which integer j ceases to exist */
```

Função (*function*)? O que é isso?
Como fazer a operação de soma utilizando uma função?

Fonte: Ricardo Gutierrez-Osuna, *Microprocessor-based System Design*, Wright State University.

Implementando uma função para executar a soma de dois números



Declaração (*prototype*) da função sum

Chamada da função sum

Um par de /* */ define um comentário que é ignorado pelo compilador

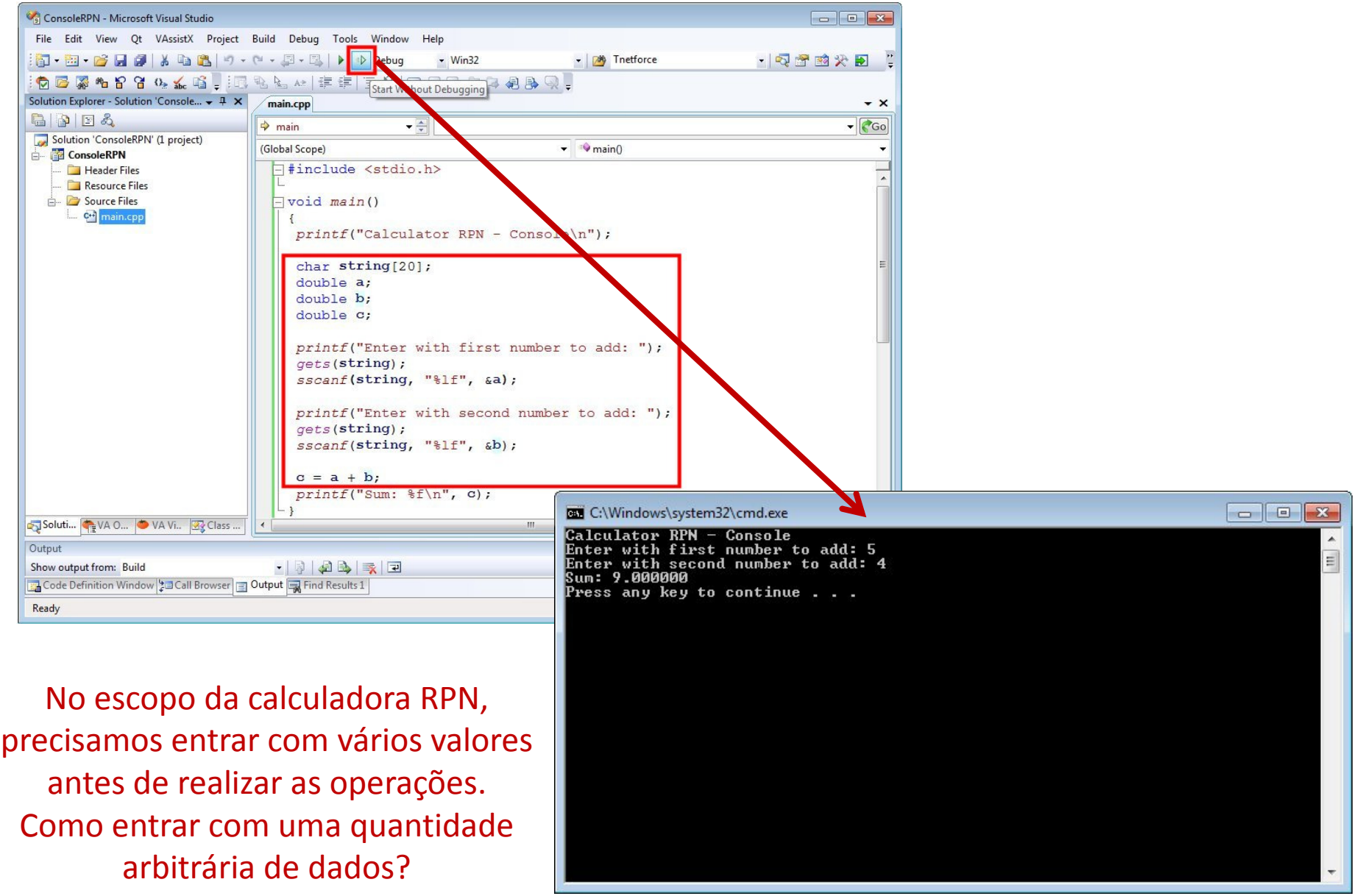
A função sum retorna um valor double

Implementação da função sum

A função sum recebe dois valores double na lista de parâmetros

Como realizar entrada de dados na linguagem C/C++?

Inserindo pelo console (entrada) os dois números a serem somados



No escopo da calculadora RPN, precisamos entrar com vários valores antes de realizar as operações. Como entrar com uma quantidade arbitrária de dados?

Loops and iterations

- **In C any expression different than ZERO is `TRUE`, including negative numbers, strings, …**
- **C provides the following constructs**

if-else

```
if (expr2) {
  block2;
} else if (expr3) {
  block3;
} else {
  default_block;
}
```

while, do-while

```
while (expression) {
  block;
}

do {
block;
} while (expression);
```

goto

```
goto label;
block1;
label:
block2;
```

for

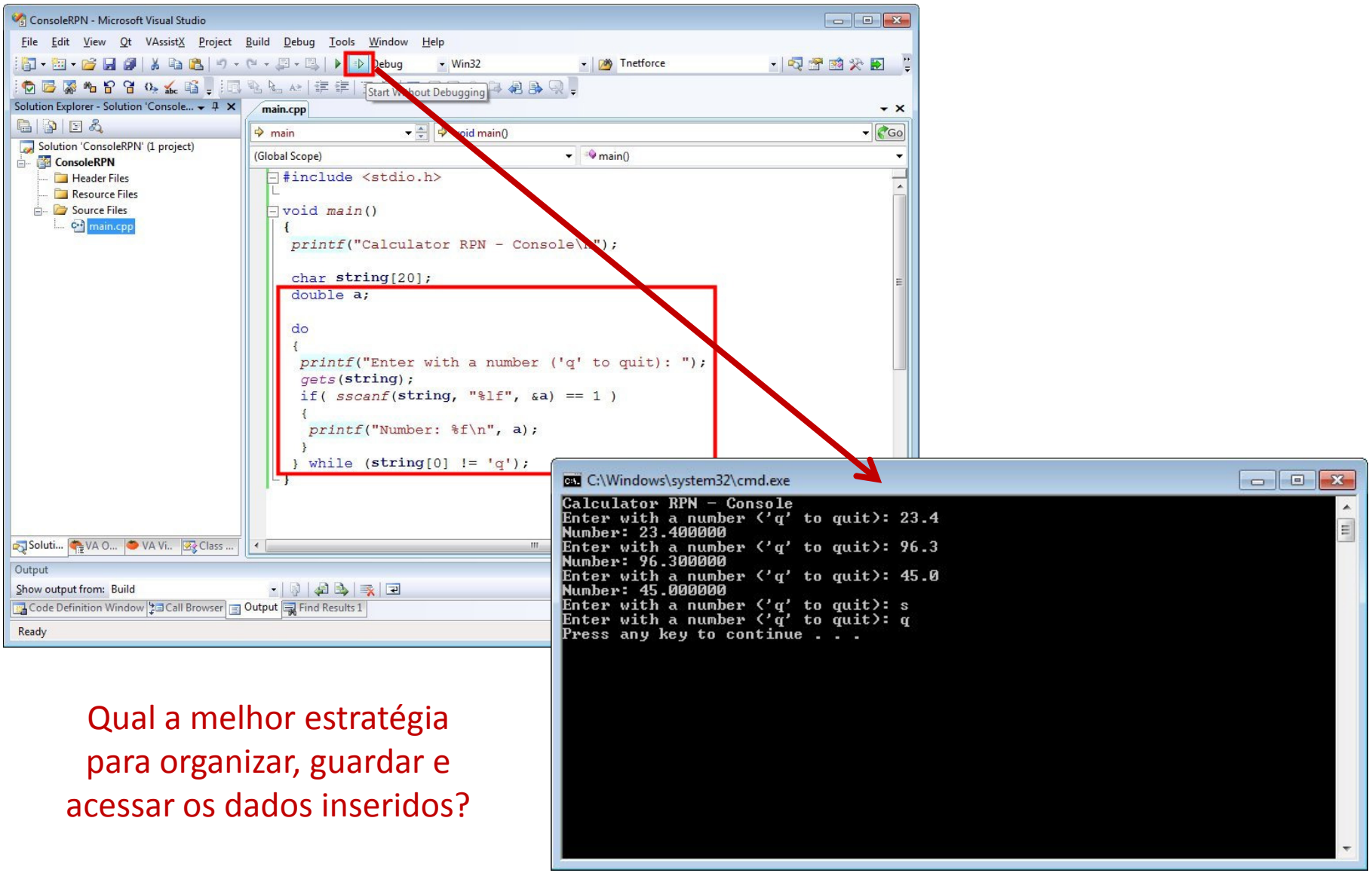
```
for (initialization;condition;increment) {
  block;
}

for (;;;) {
  block;
  if (expr)
    break;
}
```

switch-case

```
switch (expression) {
  case constant1:
    block1;
    break;
  case constant2:
    block2;
    break;
  default:
    block_default;
}
```

Fonte: Ricardo Gutierrez-Osuna, *Microprocessor-based System Design*, Wright State University.

Preparando o programa para inserir uma quantidade qualquer de números, e criação de comando para encerrar o programa



Qual a melhor estratégia para organizar, guardar e acessar os dados inseridos?

Estruturas de Dados

Qual a estrutura de dados mais adequada para o problema da calculadora RPN?

Firefox — W Estrutura de dados – Wikipédia, a enciclo... — pt.wikipedia.org/wiki/Estrutura_de_dados — pilha

Criar conta | Entrar

Artigo | Discussão | Ler | Editar | Editar código-fonte | Ver histórico | Pesquisa

WIKIPÉDIA
A enciclopédia livre

- Página principal
- Conteúdo destacado
- Eventos atuais
- Esplanada
- Página aleatória
- Portais
- Informar um erro

Colaboração
- Boas-vindas
- Ajuda
- Página de testes
- Portal comunitário
- Mudanças recentes
- Manutenção
- Criar página
- Páginas novas
- Contato
- Donativos

Imprimir/exportar

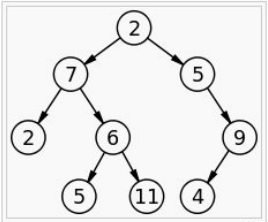
Ferramentas

Noutras línguas
- العربية

Estrutura de dados

Origem: Wikipédia, a enciclopédia livre.

Na Ciência da computação, uma **estrutura de dados** é um modo particular de armazenamento e organização de dados em um computador de modo que possam ser usados eficientemente.[1] [2]



Uma árvore binária é uma estrutura de dados.

Diferentes tipos de estrutura de dados são adequadas a diferentes tipos de aplicação e algumas são altamente especializadas, destinando-se a algumas tarefas específicas. Por exemplo, as B-trees são particularmente indicadas para a implementação de bases de dados , enquanto que a implementação de compiladores geralmente requer o uso de tabela de dispersão para a busca de identificadores.

Estruturas de dados e algoritmos são temas fundamentais da ciência da computação, sendo utilizados nas mais diversas áreas do conhecimento e com os mais diferentes propósitos de aplicação. Sabe-se que algoritmos manipulam dados. Quando estes dados estão organizados (dispostos) de forma coerente, caracterizam uma forma, uma **estrutura de dados**. A organização e os métodos para manipular essa estrutura é que lhe conferem singularidade e diminuição do espaço ocupado pela memória RAM, além de tornar o código-fonte do prog simplificado.

As estruturas de dados são chamadas tipos de dados compostos que dividem-se em homogêneos (veto (registros). As estruturas homogêneas são conjuntos de dados formados pelo mesmo tipo de dado primi heterogêneas são conjuntos de dados formados por tipos de dados primitivos diferentes (campos do regi estrutura. A escolha de uma estrutura de dados apropriada pode tornar um problema complicado em um simples. O estudo das estruturas de dados está em constante desenvolvimento (assim como o de algori existem certas estruturas clássicas que se comportam como padrões.

Índice [esconder]
1 Estruturas de dados clássicas
- 1.1 Vetores ou *arrays*
- 1.2 Lista
- 1.3 Fila
- 1.4 Pilha
- 1.5 Árvores
  - 1.5.1 Árvores binárias
- 1.6 Grafo
- 1.7 Deque
- 1.8 Tabela de hashing

2 Referências
3 Ver também

Estruturas de dados clássicas [editar | editar código-fonte]

Vetores ou *arrays* [editar | editar código-fonte]

Ver artigo principal: Array

Vetores[PB], ou **vectores**[PE] ou *arrays* são estruturas de dados lineares e estáticas, isto é, são compostas por um número fixo (finito) de elementos de um determinado tipo de dados. O tempo de acesso aos elementos de um vetor é muito rápido, sendo considerado constante: o acesso aos elementos é feito pelo seu índice no vetor. Porém, a remoção de elementos pode ser custosa se não for desejável que haja espaços "vazios" no meio do vetor, pois nesse caso é necessário "arrastar" de uma posição todos os elementos depois do elemento removido.

Essa é uma estrutura muito recomendada para casos em que os dados armazenados não mudarão, ou pouco mudarão, através do tempo.

Lista [editar | editar código-fonte]

Ver artigo principal: Lista

Uma Lista é uma estrutura de dados linear. Uma lista ligada, também chamada de encadeada, é linear e dinâmica, é composta por nós que apontam para o próximo elemento da lista, o ultimo elemento apontará para nulo. Para compor uma lista encadeada, basta guardar seu primeiro elemento.

Fila [editar | editar código-fonte]

Ver artigo principal: FIFO

As filas são estruturas baseadas no princípio **FIFO** (*first in, first out*), em que os elementos que foram inseridos no início são os primeiros a serem removidos. Uma fila possui duas funções básicas: **ENQUEUE**, que adiciona um elemento ao final da fila, e **DEQUEUE**, que remove o elemento no início da fila. A operação **DEQUEUE** só pode ser aplicada se a fila não estiver vazia, causando um erro de **underflow** ou fila vazia se esta operação for realizada nesta situação.

Pilha [editar | editar código-fonte]

Ver artigo principal: LIFO

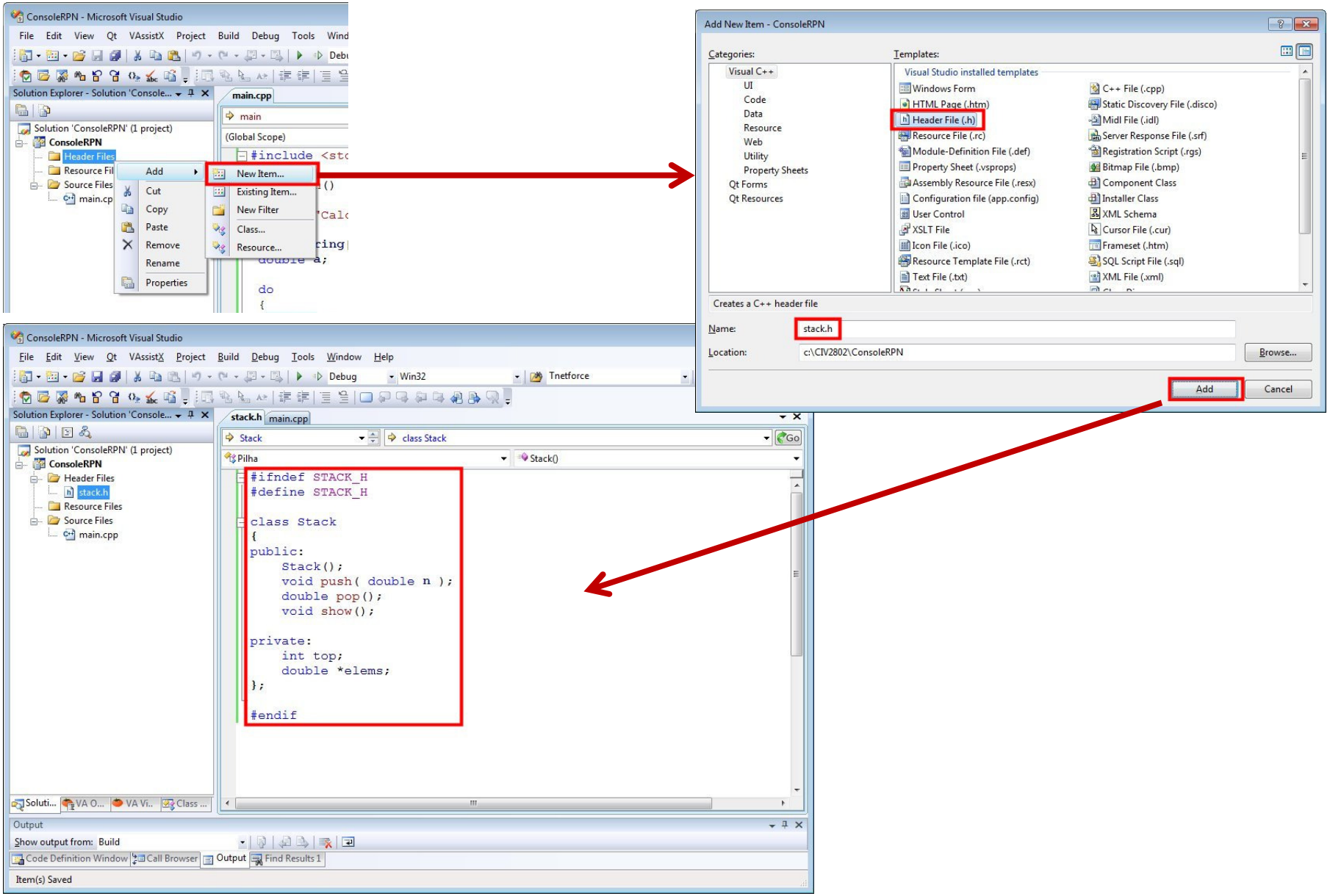
A pilha é uma estrutura de dados baseada no princípio LIFO (*last in, first out*), na qual os dados que foram inseridos primeiros na pilha serão os últimos a serem removidos. Existem duas funções que se aplicam a todas as pilhas: **PUSH**, que insere um dado no topo da pilha, e **POP**, que remove o item no topo da pilha.

A Estrutura de Dados Pilha (*Stack*)

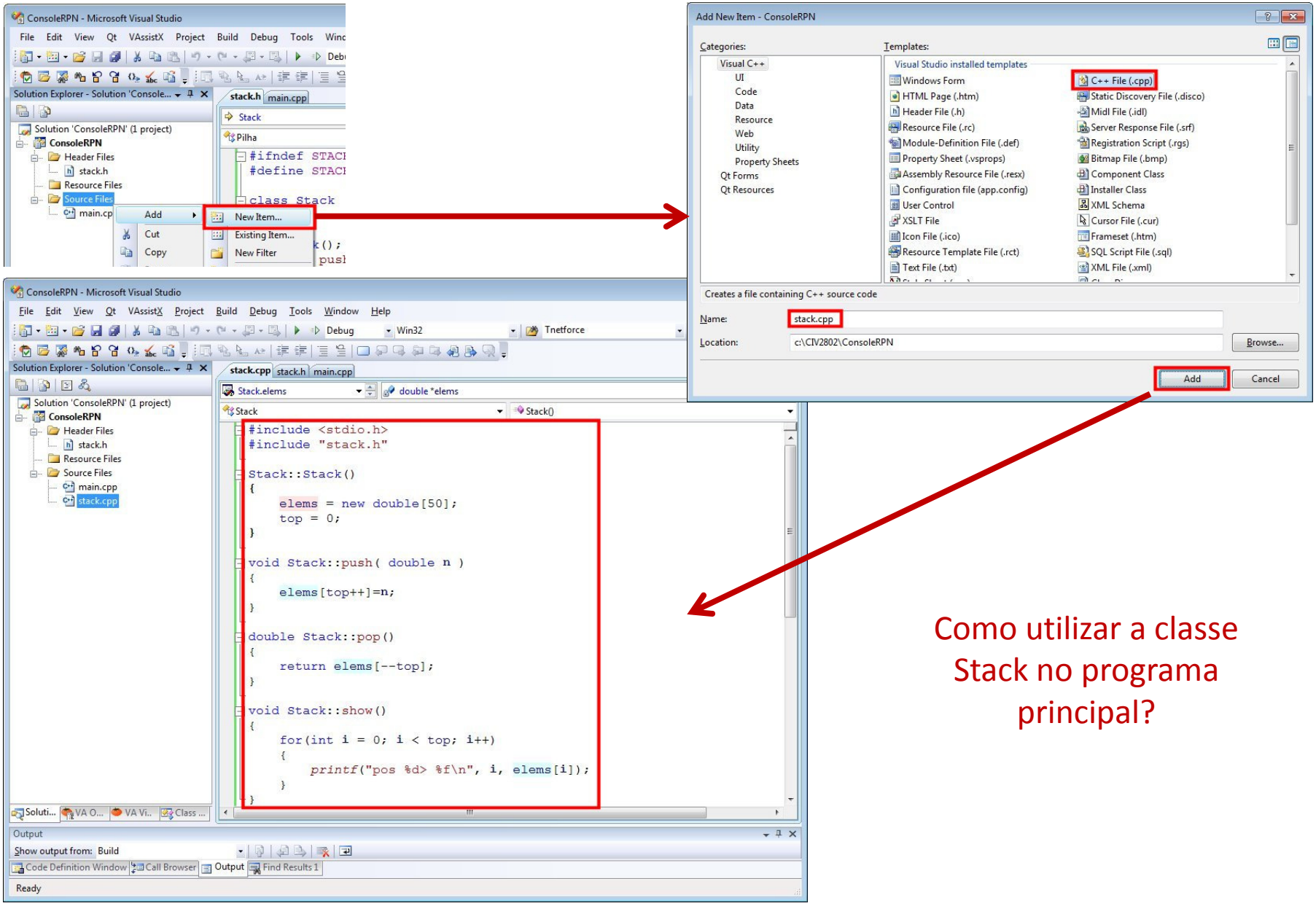
Como implementar uma estrutura de dados de pilha na linguagem C/C++?

Programação Orientada a Objetos

Implementando uma Pilha (*Stack*): criação do arquivo “stack.h”

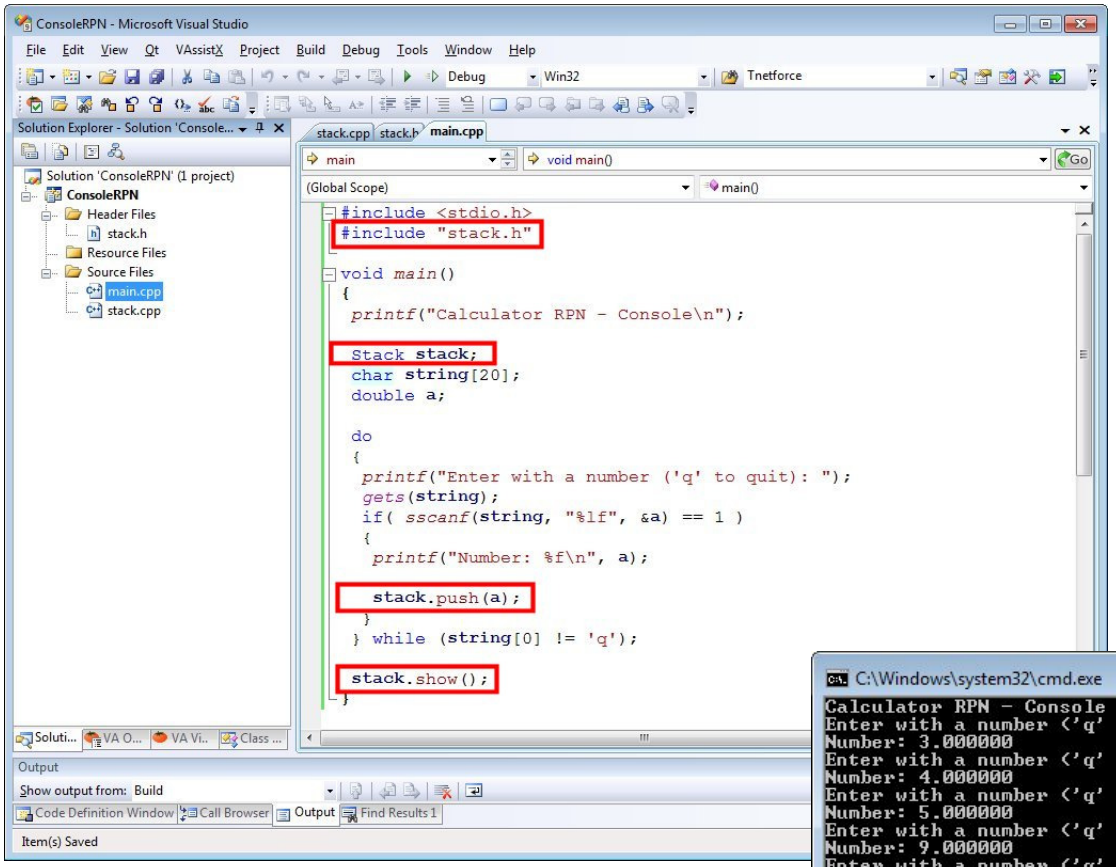


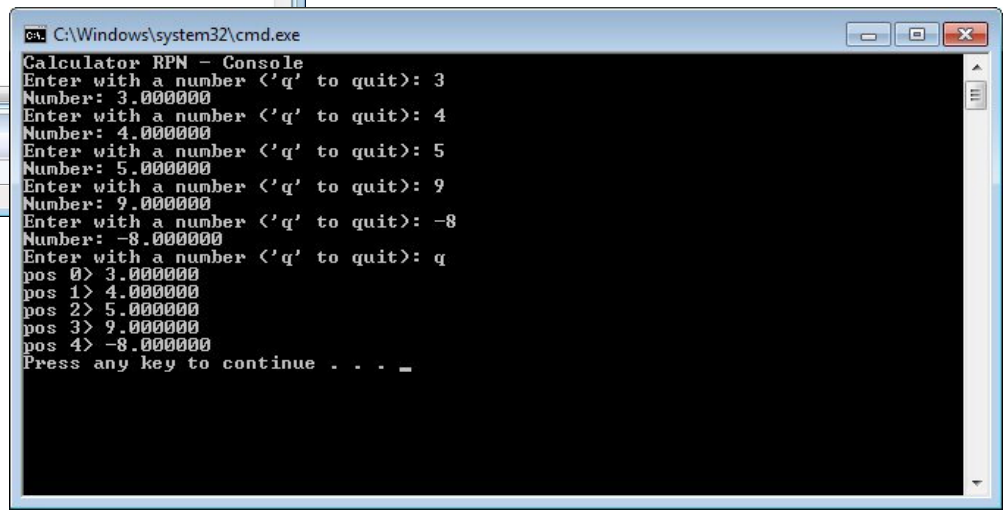
Implementando uma Pilha (*Stack*): criação do arquivo “stack.cpp”



Como utilizar a classe Stack no programa principal?

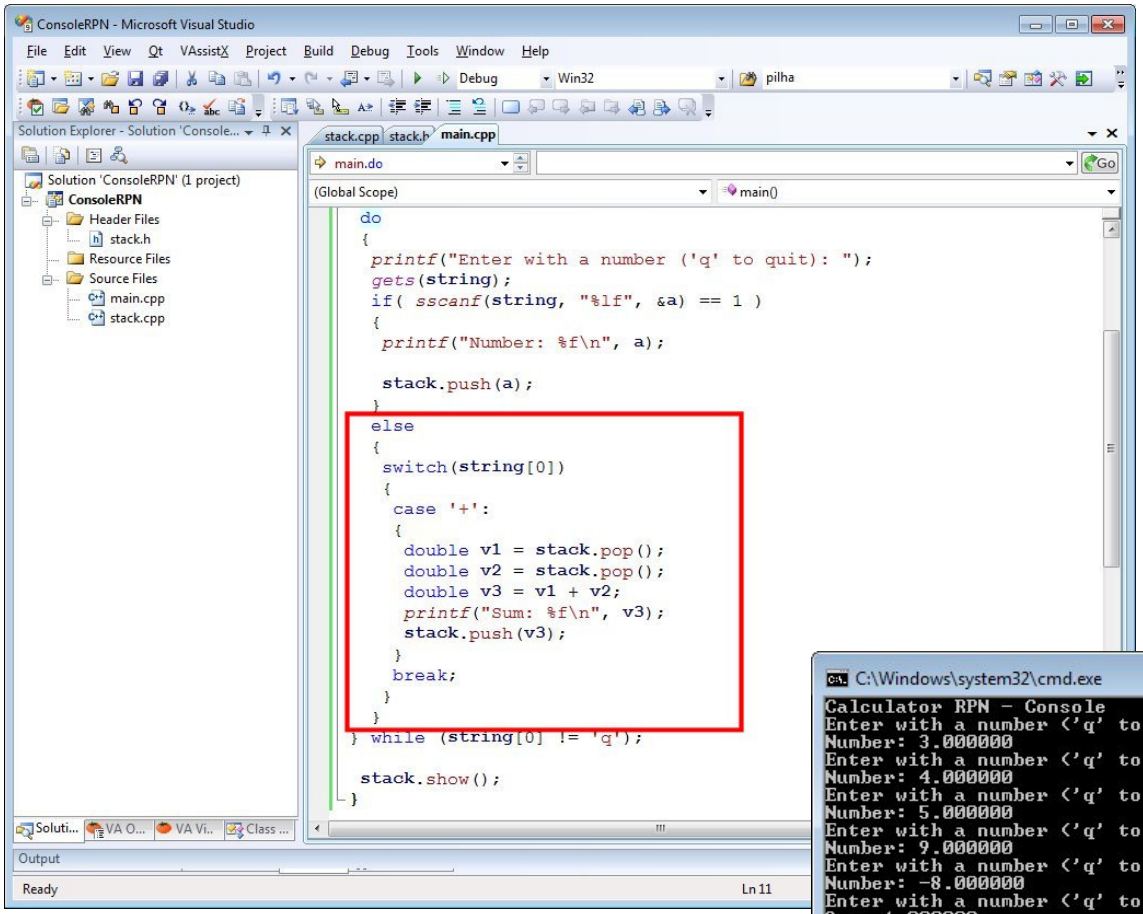
Usando a classe Stack no programa principal para armazenar os dados

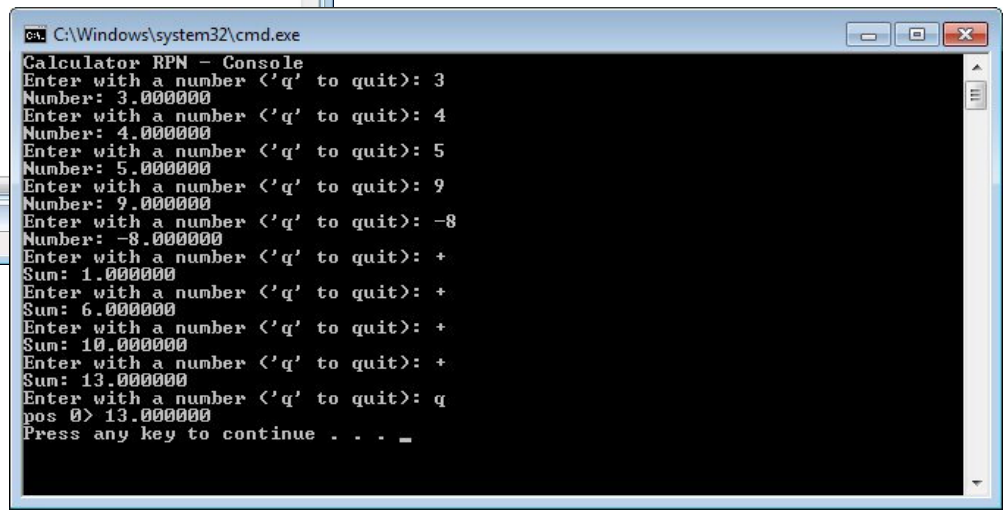




Como utilizar o objeto da classe Stack para realizar as operações da calculadora?

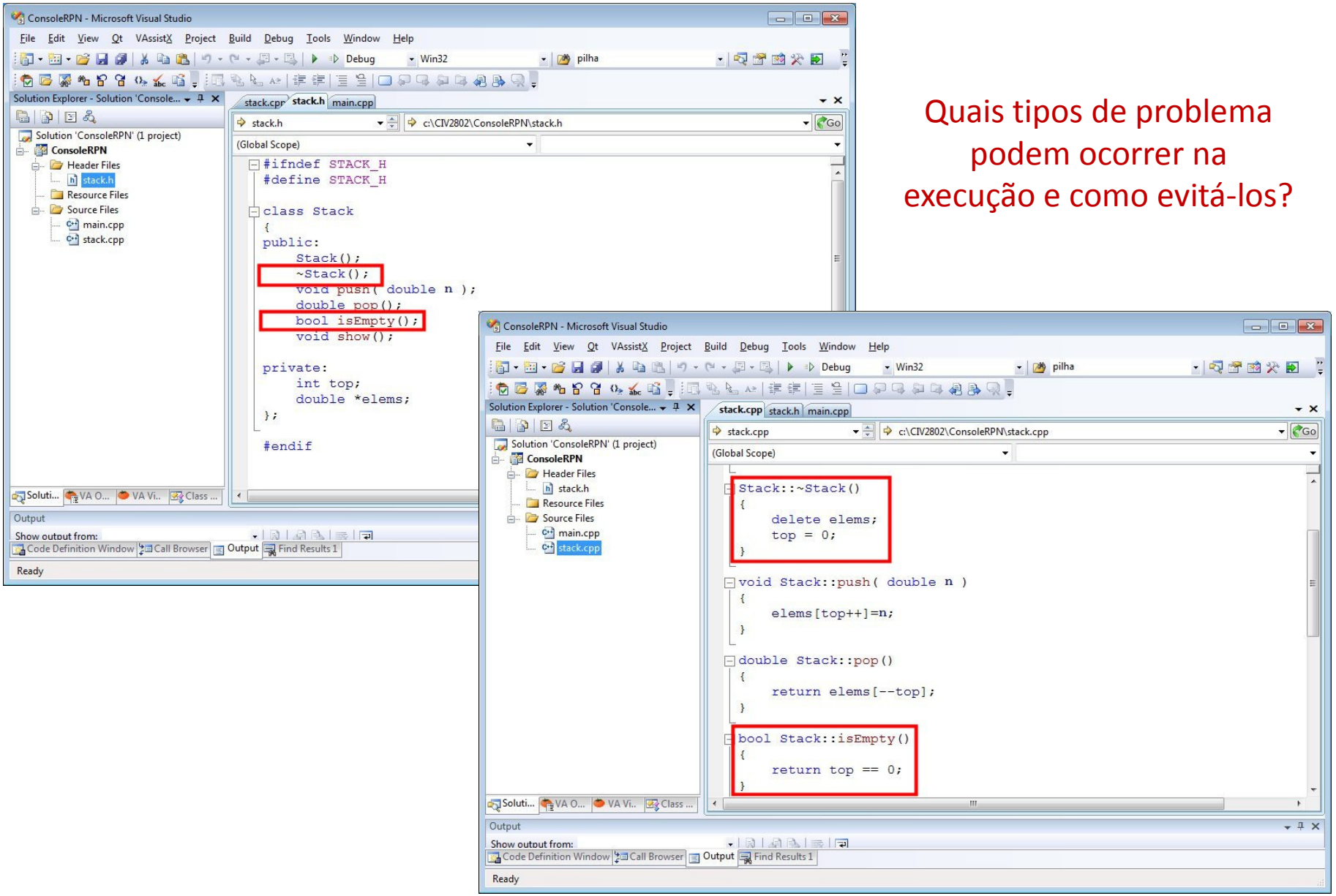
Implementação da operação de adição usando os dados da pilha





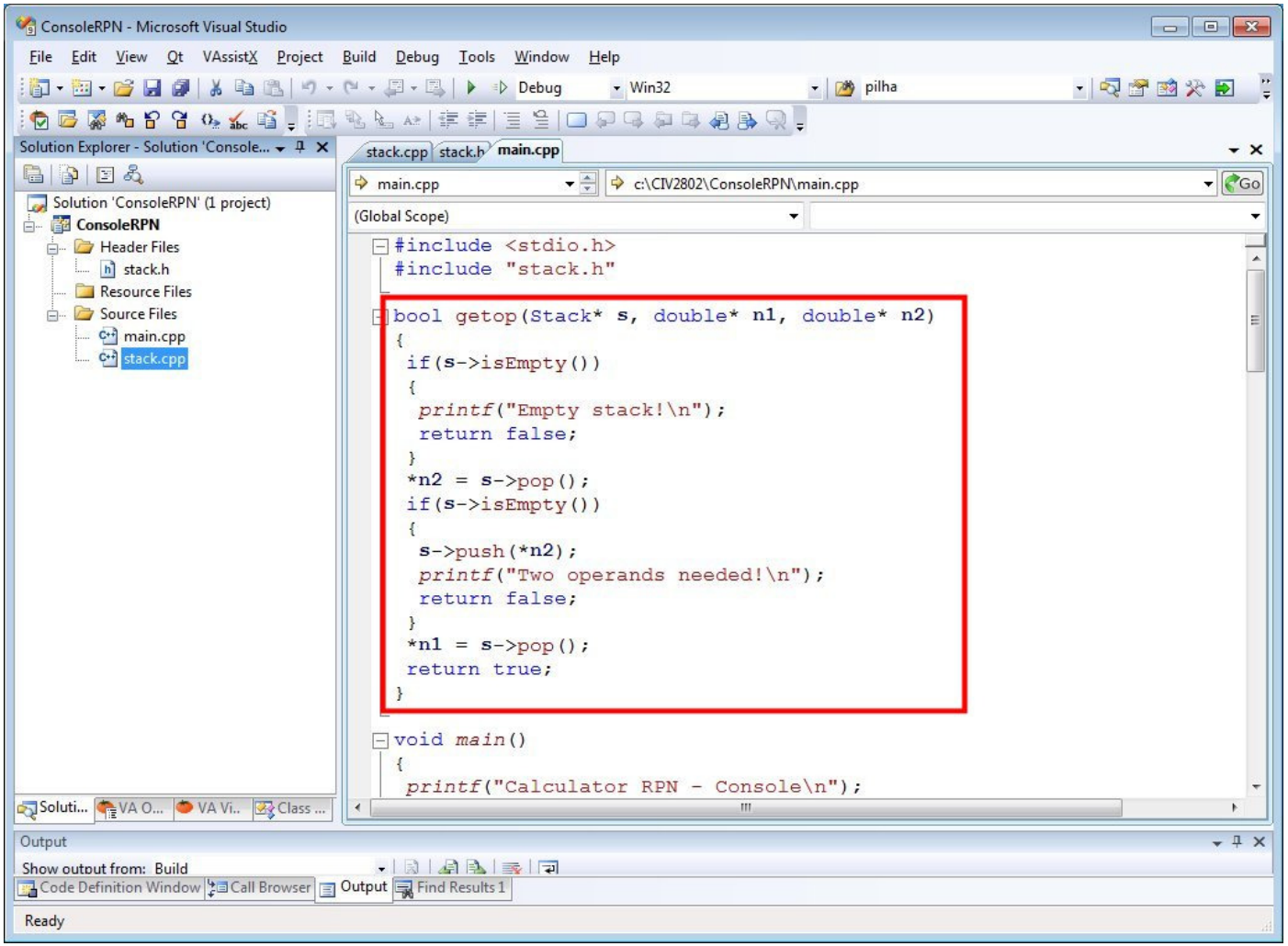
Como garantir robustez no funcionamento da calculadora?

Verificação de pilha vazia e eliminação de memória utilizada



Quais tipos de problema podem ocorrer na execução e como evitá-los?

Função auxiliar para obter os dois operandos de uma operação com o tratamento de possíveis erros



Quais são os mecanismos de passagem de parâmetros para função na linguagem C/C++?

Mecanismos de passagem de parâmetros para função em C/C++

Considere o programa abaixo com uma função “swap” cujo objetivo é a troca de valores de dois números inteiros.

```
#include <stdio.h>

void swap( int x, int y )
{
 int temp;

 temp = x;
 x = y;
 y = temp;
}

 void main( void )
 {
  int a = 2, b = 5;

  swap( a, b );

  printf( "a: %d     b: %d\n", a, b );
 }
```

Escreva o resultado do programa, isto é, o que é impresso pelo programa? Justifique.

Mecanismos de passagem de parâmetros para função em C/C++

Considere o programa abaixo com uma função “swap” cujo objetivo é a troca de valores de dois números inteiros.

```
#include <stdio.h>

void swap( int x, int y )
{
 int temp;

 temp = x;
 x = y;
 y = temp;
}

 void main( void )
 {
  int a = 2, b = 5;

  swap( a, b );

  printf( "a: %d     b: %d\n", a, b );
 }
```

Resultado do programa:

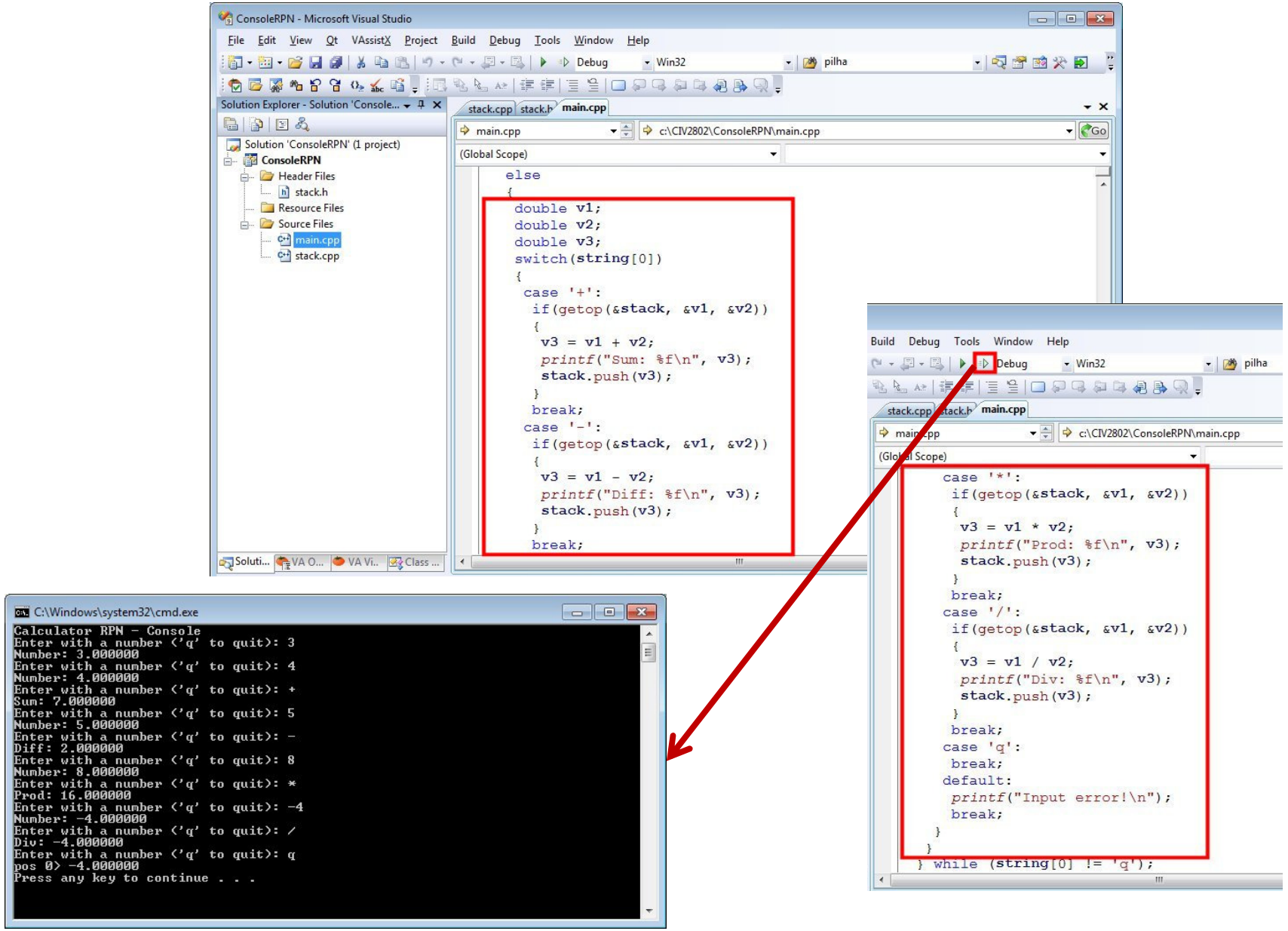
a: 2 b: 5

Programa corrigido:

```
#include <stdio.h>

void swap( int *px, int *py )
{
 int temp;

 temp = *px;
 *px = *py;
 *py = temp;
}

void main( void )
{
 int a = 2, b = 5;

 swap( &a, &b );

 printf( "a: %d     b: %d\n", a, b );
}
```

Observa-se que não houve troca dos valores dos dois números.
O motivo é que o único mecanismo de passagem de parâmetro para função em C/C++ é por valor.
Neste mecanismo, é passada uma cópia do valor da variável para o parâmetro.
A função “swap” está trocando apenas os valores das cópias e não os valores das variáveis “a” e “b”.
A solução é simular uma passagem de parâmetro por referência. Isso é feito, passando (por valor) o endereço das variáveis. Os parâmetros da função “swap” passam a ser ponteiros para inteiros.

Implementação das outras operações na calculadora via console



Testando o tratamento de possíveis erros na execução do programa