



**Profº Luiz Paulo Zanetti**

**E-mail: [luizpaulozanetti@hotmail.com](mailto:luizpaulozanetti@hotmail.com)**



**Curso Superior de Tecnologia em  
Análise e Desenvolvimento de Sistemas**

**Disciplina  
Linguagem de Programação**

# **Introdução**

## **Linguagem de programação C**

A linguagem C é uma linguagem estruturada em bloco simples. Uma característica distintiva de uma linguagem estruturada em bloco é a compartimentalização de seu código e de seus dados, que é a habilidade de uma linguagem tem de seccionar e esconder do resto do programa todas as instruções necessárias para a realização de uma determinada tarefa.

# **Introdução**

## **Linguagem de programação C**



**Dennis MacAlistair Ritchie (Bronxville, 9 de Setembro de 1941 — Berkeley Heights, 12 de Outubro de 2011)<sup>1 2</sup> foi um cientista da computação estadunidense, notável pela sua influência em linguagens de programação como ALTRAN, B, BCPL e C, e em sistemas operacionais como o Multics e o UNIX.**

**Nascido em Bronxville, Nova Iorque, Ritchie formou-se em física e matemática aplicada pela Universidade de Harvard. Em 1967 começou a trabalhar no Centro de Investigação de Ciências Computacionais dos laboratórios Bell. Foi chefe do Departamento de Investigação de Software de Sistemas da Lucent Technologies. Em 1983, ele e Ken Thompson receberam o Prémio Turing "pelo seu desenvolvimento de teoria de sistemas operativos genéricos e especialmente pela sua implementação do sistema operativo UNIX."**

# Introdução

## Algumas vantagens da linguagem

### Rapidez

- Consegue obter performance semelhante ao Assembly, usando instruções em alto nível.

### Simples

- Sintaxe simples, número diminuto de palavras reservadas, de tipos de dados e de operadores.

### Portável

- Padrão ANSI – código escrito em uma máquina pode ser compilado em outra máquina (com poucas ou sem alterações)

### Popular

- É a mais conhecida e utilizada no mundo.

### Modular

- Permite a programação modular, facilita a separação de projetos em módulos distintos e independentes, uso de funções.

### Alto Nível

- Linguagem de terceira geração, permite acesso a maior parte das funcionalidades de Assembly.

### Outros

- Bibliotecas adicionais, evolução (POO) C++ . Java se baseia em C/C++.

# Introdução

## Filosofia da programação em C

Modularidade:

- Separar e implementar pequenos pedaços de códigos que realizem corretamente uma única função, e a realize bem. Exemplo:



Cada módulo é implementado de maneira independente.  
Cada módulo, por sua vez, é dividido nos diversos componentes que o compõe.

# Introdução

## C versus C++

A linguagem C é um subconjunto da linguagem C++, isto é, C++ contém todas as características da linguagem C e mais um subconjunto de características próprias.

Nota:

– Para se dar um salto para C++, é imprescindível que o aluno tenha o domínio de C.

# Introdução

## Função

- Um programa em linguagem C é formado por uma ou mais funções.
- Cada função possui um nome exclusivo e corresponde à um bloco de código, delimitado por um par de chaves: { }
- Contém um conjunto de declarações, expressões, comandos de controle e chamadas à outras funções.



# Introdução

## main e void main

- A função denominada **main** é obrigatória em todos os programas, pois é o seu ponto de entrada, isto é, o programa começa a ser executado no início da função main e termina ao final desta função.
- Normalmente a declaração desta função possui a seguinte forma: **void main()** ou **main()** sua distinção se dá ao compilador utilizado.

# Introdução

`main()`

- Os parênteses sem mais nada após a função indicam que ela não recebe qualquer informação exterior.
- Nota: C é Case Sensitive. Faz diferenciação entre
- maiúsculas e minúsculas. Todas as instruções de C são escritas em letras minúsculas. Usa-se letras maiúsculas quando se deseja utilizar variáveis, mensagens ou funções.

# Comandos de Atribuição

`x = 4;`

`b = b + 2;`

`y = 2.5;`

`sexo = 'F';`

# Introdução

## Exemplo - Olá Mundo – Hello World

No exemplo temos:

### **Linha 1:**

- É uma diretiva que indica ao compilador que deverá adicionar ao processo um arquivo chamado “stdio.h” Biblioteca de entrada e saída.

### **Linha 2:**

- Função principal, entrada principal do programa.

### **Linhas 3 e 5:**

- Respectivamente início e fim do bloco de comandos.

```
1: #include <stdio.h>
2: main()
3: {
4:     printf("Hello World");
5: }
```

### **Linha 4:**

- Comando de saída – exibe a mensagem “Hello World” na console

# **Introdução**

## **Nome de Variáveis**

**Nomes de variáveis só podem conter letras do alfabeto, números e o caractere underscore “\_”.**

**Não podem começar com um número.**

**Nomes que comecem com um ou dois caracteres underscore (“\_” e “\_\_”) são reservados para a implementação interna do programa e seu uso é extremamente desaconselhado. O compilador não acusa erro quando criamos variáveis desse jeito, mas o programa criado se comportará de forma inesperada.**

# **Introdução**

## **Nome de Variáveis**

**Não é possível utilizar palavras reservadas da linguagem C ou o mesmo nome de um função, mesmo que essa função tenha sido criada pelo programador ou seja uma função de biblioteca.**

**C diferencia letras maiúsculas e minúsculas em nomes de variáveis. Ou seja, casa, Casa e CASA são três nomes de variáveis distintos.**

**C não estabelece limites para o número de caracteres em um nome de variável, e todos os caracteres são significantes.**

# **Introdução**

## **Nome de Variáveis**

**A linguagem C possui 3 tipos básicos de variáveis que são:**

**int**

**float**

**char**

# Introdução

## Nome de Variáveis

### TIPO

### FAIXA DE VALORES

### TAMANHO

char	-128 a 127	8 bits
unsigned char	0 a 255	8 bits
int	-32768 a 32767	16 bits
unsigned int	0 a 65.535	16 bits
short int	-32768 a 32767	16 bits
long	-2.147.483.648 a 2.147.483.647	32 bits
unsigned long	0 a 4.294.967.295	32 bits
float	$3.4 \times 10^{-38}$ a $3.4 \times 10^{38}$	32 bits
double	$1.7 \times 10^{-308}$ a $1.7 \times 10^{308}$	64 bits
long double	$3.4 \times 10^{-4932}$ a $1.1 \times 10^{4932}$	80 bits



# Introdução

## Declaração de Inteiros

Podemos usar os seguintes prefixos:

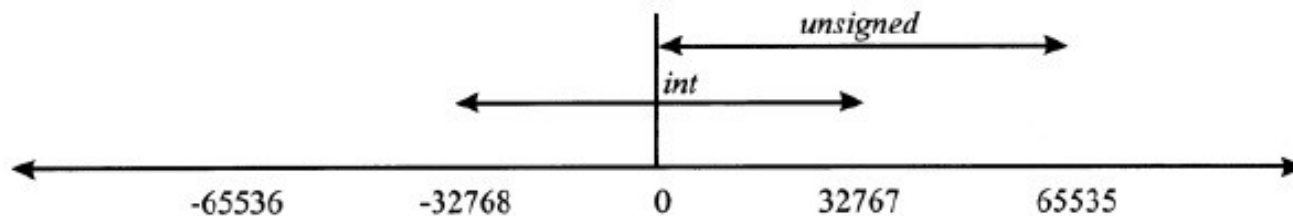
OBS: valores em um SO de 16 bits.

- **short** — Inteiro pequeno (2 *bytes*)
- **long** — Inteiro grande (4 *bytes*)
- **signed** — Inteiro com sinal (nºs negativos e positivos)
- **unsigned** — Inteiro sem sinal (apenas nºs positivos)

# Introdução

## signed e unsigned

- Uma variável do tipo inteiro admite valores positivos e negativos.
- Se um inteiro for armazenado em 2 bytes os seus valores podem variar entre -32768 e 32767.
- Se desejar valores apenas positivos use o prefixo unsigned.



**Atenção1: o formato para leitura e escrita de inteiros sem sinal é %u ao invés de %d**

**Atenção2: - 2147483648 0 2147483647 (2 ^ 32) [Sistema de 32 bits]**

# Introdução

## Variáveis

- Sempre que desejamos guardar um valor, devemos declarar variáveis.
- Uma variável é um nome que damos a uma determinada posição de memória para conter um valor de um determinado tipo de dados.
- A declaração de uma variável deve ser feita antes de sua utilização e antes de qualquer instrução.

```
main()  
{  
    Declaração de variáveis ←  
  
    Instrução1;  
    Instrução2;  
}
```

# Introdução

## Nome de Variáveis

Conjunto de regras para definição de nomes de variáveis:

- O nome de uma variável deve ser constituído por letras do alfabeto (maiúsculas e minúsculas).
- Maiúsculas e minúsculas representam caracteres diferentes, logo variáveis distintas.
- O primeiro caracteres não pode ser um dígito. Pode ser uma letra, ou o caractere underscore.
- Uma variável não pode ter por nome uma palavra reservada da linguagem C

# Introdução

## Nome de Variáveis

```
int idade;          /* Correto */
int Num_Cliente;    /* Correto */
float a1b2c3;       /* Correto */
float 7a2b3c;       /* INCORRETO: primeiro caractere é um dígito */
char float;         /* INCORRETO: utilizou-se uma palavra reservada */
double vinte%;      /* INCORRETO: utilizou-se caractere inadmissível */
char sim?não;       /* INCORRETO: utilizou-se caractere inadmissível */
int _alfa;          /* Correto, mas não aconselhável */
int _123;           /* Correto, mas não aconselhável */
                    /* Notar que o primeiro caractere não é um dígito */
                    /* mas sim o underscore */
char Num, NUM;      /* Correto, pois o C é case sensitive. */
                    /* Será aconselhável ??? */
```

# Introdução

## Atribuição

Sempre que uma variável é declarada, estamos solicitando ao compilador para reservar um espaço em memória para armazená-la.

– Esse espaço passará a ser referenciado por esse nome da variável.

Nota: Quando uma variável é declarada fica sempre com um valor, **o qual é o resultado do estado aleatório dos bits que a constituem.**

# Introdução

## Atribuição

Uma variável poderá ser iniciada com um valor através de uma operação de atribuição.

```
int num = -17;    /* num é declarada do tipo int e automaticamente */  
                  /* iniciada com o valor -17 */  
  
int n1=3, n2=5;   /* n1 e n2 são declaradas e ficam com os valores */  
                  /* 3 e 5 respectivamente */  
  
int a = 10, b, c = -123, d;  
                  /* a e c são automaticamente iniciadas com os  
                  * valores 10 e -123.  
                  * b e d ficam com um valor aleatório ("lixo")  
                  * porque não foram iniciadas.  
                  */
```

# Introdução

## Operações sobre inteiros

Operação	Descrição	Exemplo	Resultado
+	Soma	$21 + 4$	25
-	Subtração	$21 - 4$	17
*	Multiplicação	$21 * 4$	84
/	Divisão Inteira	$21 / 4$	5
%	Resto da Divisão Inteira ( <b>Módulo</b> )	$21 \% 4$	1

**NOTA:** Qualquer operação entre inteiros retorna um inteiro.



# Introdução

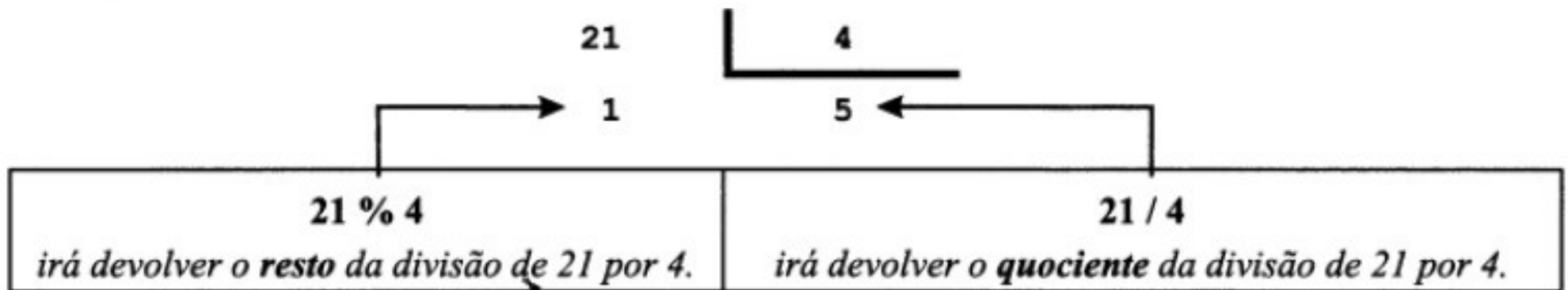
## Função para Leitura de Caracteres

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int num;
6:
7:     printf("Introduza um N°: ");
8:     scanf ("%d",&num);
9:     printf("O N° introduzido foi %d\n",num);
10: }
```

# Introdução

## Operações sobre inteiros

Assim, da divisão entre 21 e 4 não irá resultar 5,25, como se poderia pensar, uma vez que o resultado de uma operação entre dois inteiros (21 e 4) tem sempre como resultado um inteiro.



# Introdução

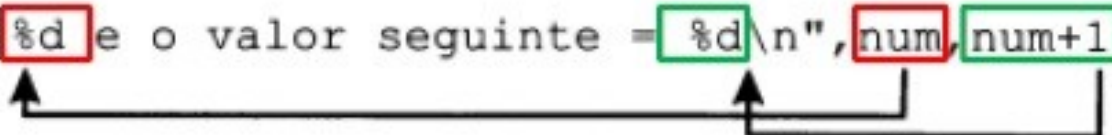
## Formato da escrita de um inteiro (%d)

Vamos então colocar o símbolo `%d` no local onde queremos escrever os inteiros:

O valor de `num` = `%d` e o valor seguinte = `%d\n`

Falta apenas indicar ao *printf* quais os valores que terá que colocar nos locais assinalados por `%d`.

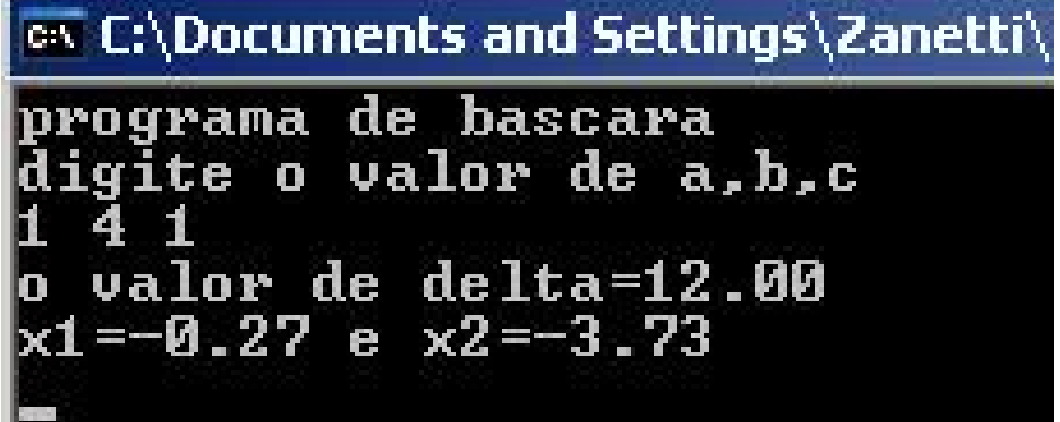
```
printf("O valor de num = %d e o valor seguinte = %d\n", num, num+1);
```



The diagram illustrates the argument passing in the `printf` function call. It shows the format string `"O valor de num = %d e o valor seguinte = %d\n"` with two placeholders `%d` highlighted in red boxes. The first `%d` is followed by the variable `num` (highlighted in a red box), and the second `%d` is followed by the expression `num+1` (highlighted in a green box). Arrows indicate the mapping: one arrow points from the first `%d` to `num`, and another arrow points from the second `%d` to `num+1`.

# Introdução

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
void main()
{
float a,b,c,x1,x2,d;
clrscr();
printf("programa de bascara\n");
printf("digite o valor de abc\n");
scanf("%f%f%f",&a,&b,&c);
d=(b*b)-(4*a*c);
x1=(-b+sqrt(d))/(2*a);
x2=(-b-sqrt(d))/(2*a);
printf("delta=%.2f\n",d);
printf("x1=%.2f e x2=%.2f\n",x1,x2);
getch();
}
```



```
C:\Documents and Settings\Zanetti\
programa de bascara
digite o valor de a,b,c
1 4 1
o valor de delta=12.00
x1=-0.27 e x2=-3.73
_
```

# Introdução

```
[■] \EXERC\1_1-BA~1.CPP
#include<conio.h> //RESPONSÁVEL PELOS COMANDOS CLRSCR E GETCH
#include<stdio.h> //RESPONSÁVEL PELOS COMANDOS PRINTF E SCANF
#include<math.h> //RESPONSÁVEL PELO COMANDO SQRT
void main() //PROGRAMA PRINCIPAL
{ //INICIO
float a,b,c,x1,x2,d; //VARIÁVEL TIPO REAL
clrscr(); //LIMPAR TELA
printf("programa de bascara\n"); //IMPRIMIR TEXTO PARA O USUARIO
printf("digite o valor de a,b,c\n"); //IMPRIMIR TEXTO PARA O USUARIO
scanf("%f%f%f",&a,&b,&c); //OBTER INFORMACAO DO USUARIO
d=(b*b)-(4*a*c); //FORMULA DE DELTA
x1=(-b+sqrt(d))/(2*a); //FORMULA X1
x2=(-b-sqrt(d))/(2*a); //FORMULA X2
printf("delta=%.2f\n",d); //IMPRIMIR TEXTO PARA O USUARIO
printf("x1=%.2f e x2=%.2f\n",x1,x2); //IMPRIMIR TEXTO PARA O USUARIO
getch(); //PAUSA
} //FIM
```

C:\Documents and Settings\Zanetti\

```
programa de bascara
digite o valor de a,b,c
1 4 1
o valor de delta=12.00
x1=-0.27 e x2=-3.73
_
```

# Introdução

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
```

```
main()
```

```
{
```

```
float a,b,c,d,x1,x2;
```

```
printf("programa de bascara\n");
```

```
printf("digite o valor de a,b,c\n");
```

```
scanf("%f%f%f",&a,&b,&c);
```

```
d=(b*b)-(4*a*c);
```

```
x1=(-b+sqrt(d))/(2*a);
```

```
x2=(-b-sqrt(d))/(2*a);
```

```
printf("o valor de delta=%.2f\n",d);
```

```
printf("x1=%.2f e x2=%.2f\n",x1,x2);
```

```
getch();
```

```
}
```

C:\Documents and Settings\Zanetti\

programa de bascara

digite o valor de a,b,c

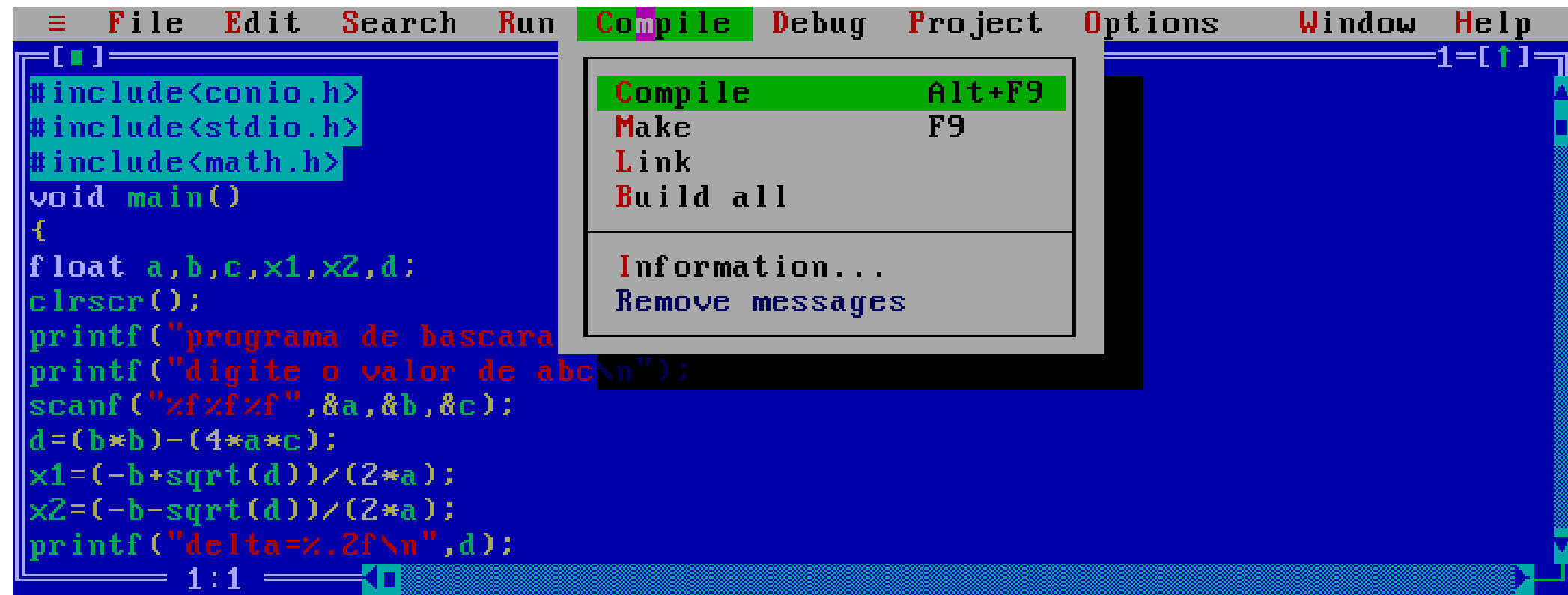
1 4 1

o valor de delta=12.00

x1=-0.27 e x2=-3.73

\_

# Compiler – ( Compile )



The image shows a screenshot of a Turbo C++ IDE. The main window displays a C program for solving a quadratic equation. The menu bar includes File, Edit, Search, Run, Compile, Debug, Project, Options, Window, and Help. The 'Compile' menu is open, showing options: Compile (Alt+F9), Make (F9), Link, Build all, Information..., and Remove messages. The program code is as follows:

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
void main()
{
float a,b,c,x1,x2,d;
clrscr();
printf("programa de bascara\n");
printf("digite o valor de abc\n");
scanf("%f%f%f",&a,&b,&c);
d=(b*b)-(4*a*c);
x1=(-b+sqrt(d))/(2*a);
x2=(-b-sqrt(d))/(2*a);
printf("delta=%.2f\n",d);
}
```

The status bar at the bottom left shows '1:1'.

# Compilar – ( Compile )

The screenshot shows a Turbo C++ IDE window titled "1-BASC~1.CPP". The menu bar includes File, Edit, Search, Run, Compile, Debug, Project, Options, Window, and Help. The code editor contains the following C++ code:

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
void main()
{
float a,b,c,x1,x2,
clrscr();
printf("programa d
printf("digite o va
scanf("%f%f%f",&a,
d=(b*b)-(4*a*c);
x1=(-b+sqrt(d))/(2
x2=(-b-sqrt(d))/(2
printf("delta=%.2f
printf("x1=%.2f e
getch();
}
```

A "Compiling" dialog box is displayed in the foreground, showing the compilation results:

Main file: ..\..\EXERC\1-BASC~1.CPP  
Compiling: EDITOR → 1-BASC~1.CPP

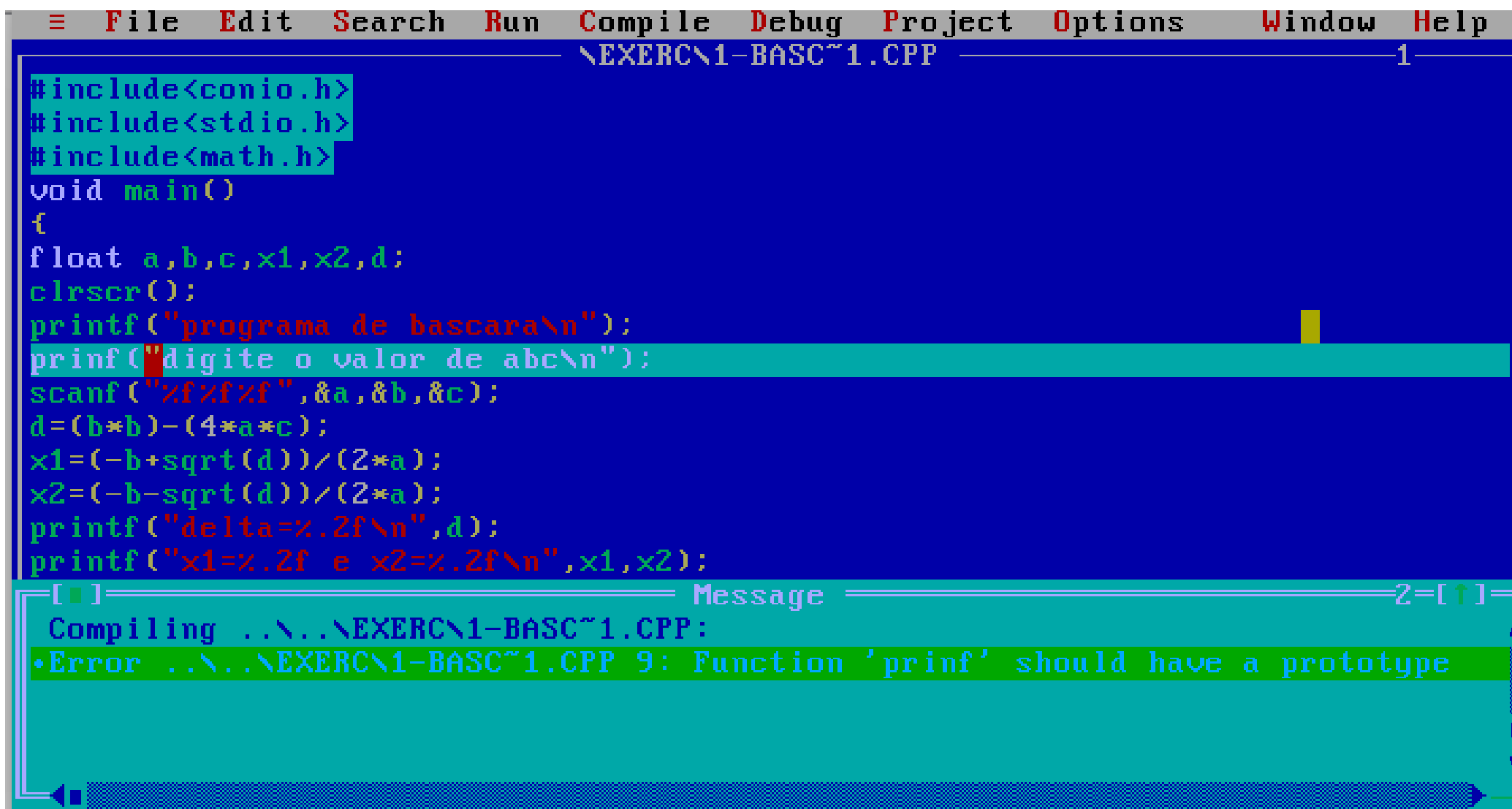
	Total	File
Lines compiled:	626	626
Warnings:	0	0
Errors:	1	1

Available memory: 1973K  
Errors : Press any key

The status bar at the bottom left shows the time as 9:5.



# Compilar – ( Compile )



The image shows a screenshot of a C++ IDE with a blue background. The menu bar at the top includes File, Edit, Search, Run, Compile, Debug, Project, Options, Window, and Help. The title bar indicates the file is \EXERC\1-BASC~1.CPP. The code in the editor is as follows:

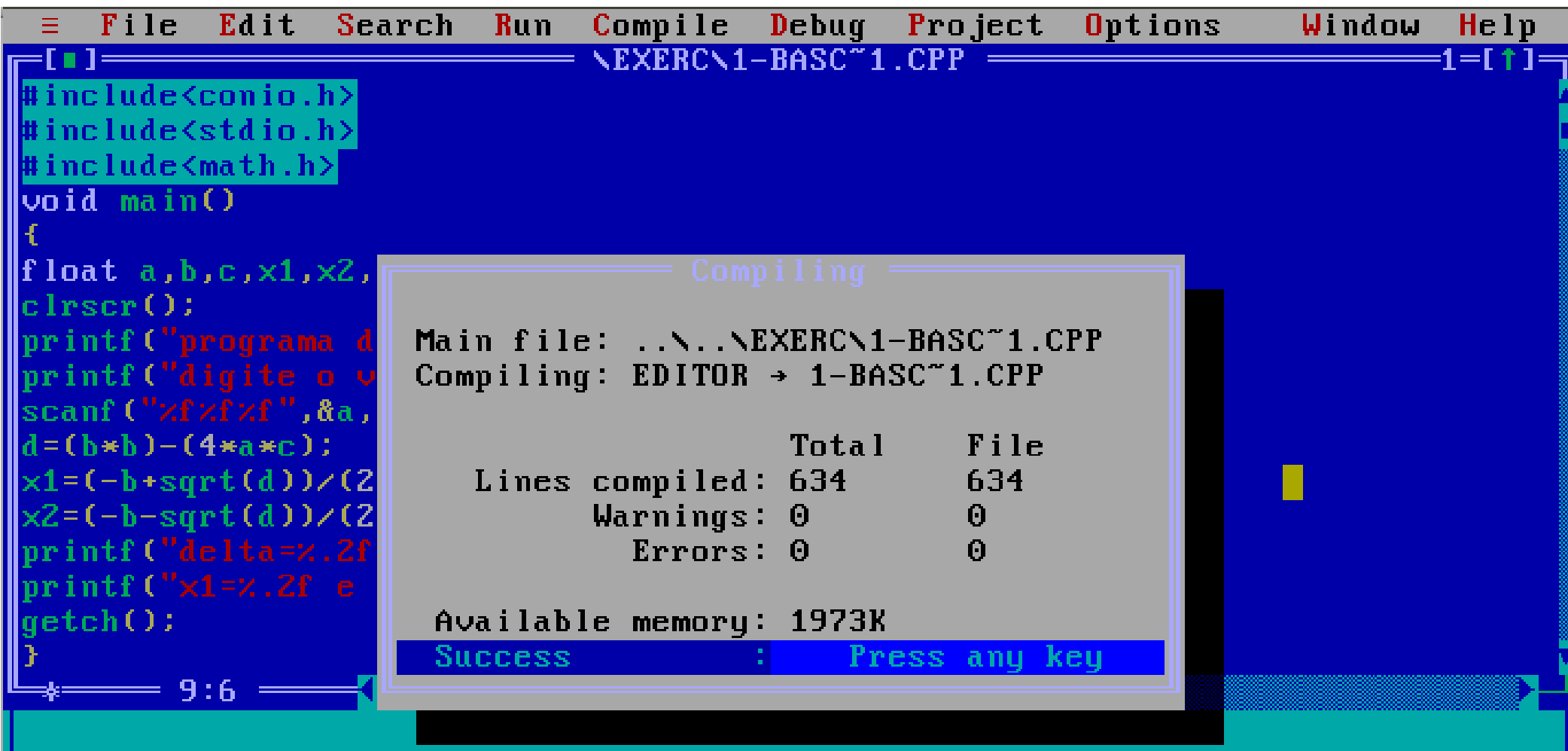
```
#include<conio.h>
#include<stdio.h>
#include<math.h>
void main()
{
float a,b,c,x1,x2,d;
clrscr();
printf("programa de bascara\n");
printf("digite o valor de abc\n");
scanf("%f%f%f",&a,&b,&c);
d=(b*b)-(4*a*c);
x1=(-b+sqrt(d))/(2*a);
x2=(-b-sqrt(d))/(2*a);
printf("delta=%.2f\n",d);
printf("x1=%.2f e x2=%.2f\n",x1,x2);
}
```

A yellow cursor is positioned at the end of the line `printf("digite o valor de abc\n");`. Below the code editor, a message box is open with the title "Message". It contains the following text:

Compiling ..\..\EXERC\1-BASC~1.CPP:  
•Error ..\..\EXERC\1-BASC~1.CPP 9: Function 'printf' should have a prototype

The error message is highlighted in green. The message box has a close button (X) in the top right corner.

# Compilar – ( Compile )



The screenshot shows a Turbo C++ IDE window titled "\EXERC\1-BASC~1.CPP". The menu bar includes File, Edit, Search, Run, Compile, Debug, Project, Options, Window, and Help. The code editor contains the following C++ code:

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
void main()
{
float a,b,c,x1,x2;
clrscr();
printf("programa d");
printf("digite o v");
scanf("%f%f%f",&a,
d=(b*b)-(4*a*c);
x1=(-b+sqrt(d))/(2
x2=(-b-sqrt(d))/(2
printf("delta=%.2f");
printf("x1=%.2f e");
getch();
}
```

A "Compiling" dialog box is displayed in the foreground, showing the compilation details:

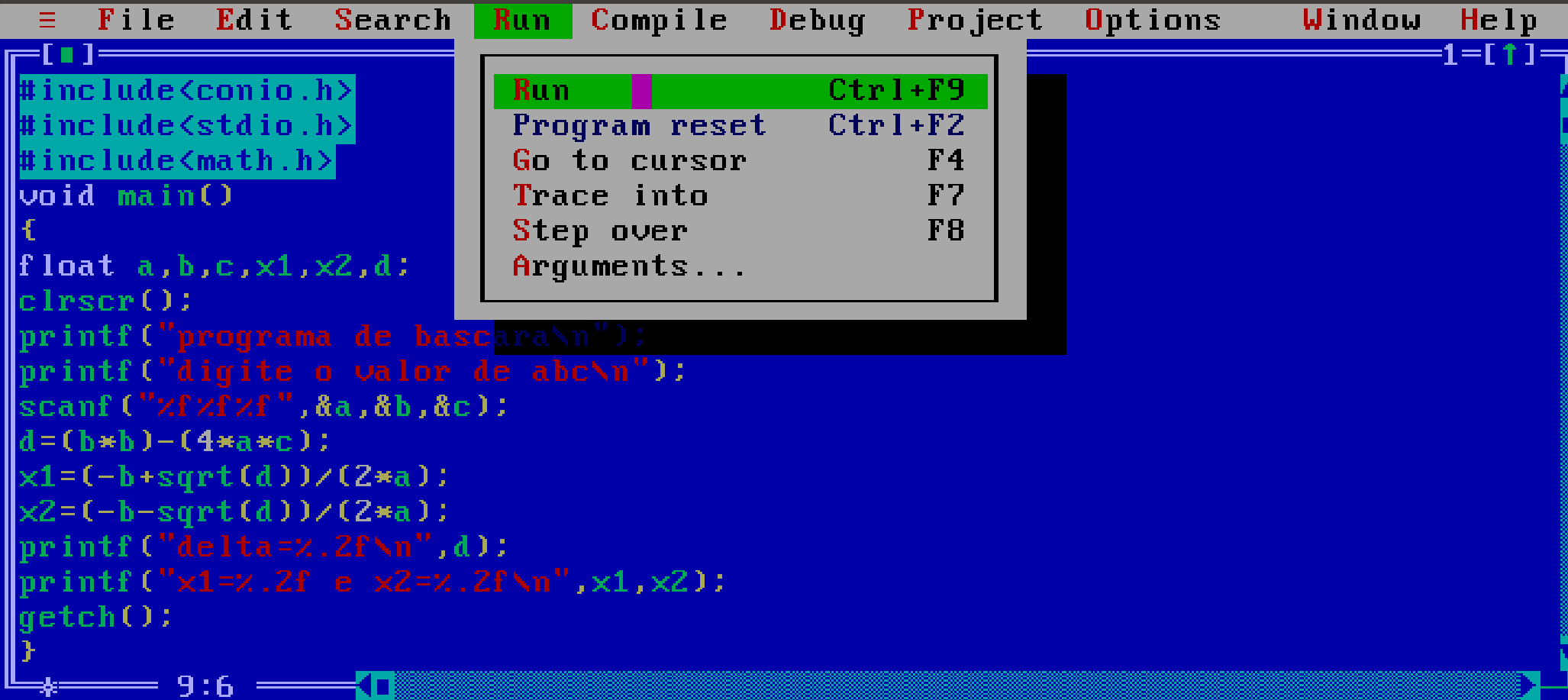
Main file: ..\..\EXERC\1-BASC~1.CPP  
Compiling: EDITOR → 1-BASC~1.CPP

	Total	File
Lines compiled:	634	634
Warnings:	0	0
Errors:	0	0

Available memory: 1973K  
Success : Press any key

The status bar at the bottom left shows the cursor position: 9:6.

# Rodar o Código – ( Run )



The image shows a screenshot of a Turbo C++ IDE. The main window displays a C program that calculates the roots of a quadratic equation. The code is as follows:

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
void main()
{
float a,b,c,x1,x2,d;
clrscr();
printf("programa de bascara\n");
printf("digite o valor de abc\n");
scanf("%f%f%f",&a,&b,&c);
d=(b*b)-(4*a*c);
x1=(-b+sqrt(d))/(2*a);
x2=(-b-sqrt(d))/(2*a);
printf("delta=%.2f\n",d);
printf("x1=%.2f e x2=%.2f\n",x1,x2);
getch();
}
```

Overlaid on the IDE is the 'Run' menu, which lists the following options and their corresponding keyboard shortcuts:

Run	Ctrl+F9
Program reset	Ctrl+F2
Go to cursor	F4
Trace into	F7
Step over	F8
Arguments...	

The status bar at the bottom of the IDE shows the file name '9:6' and a cursor icon.

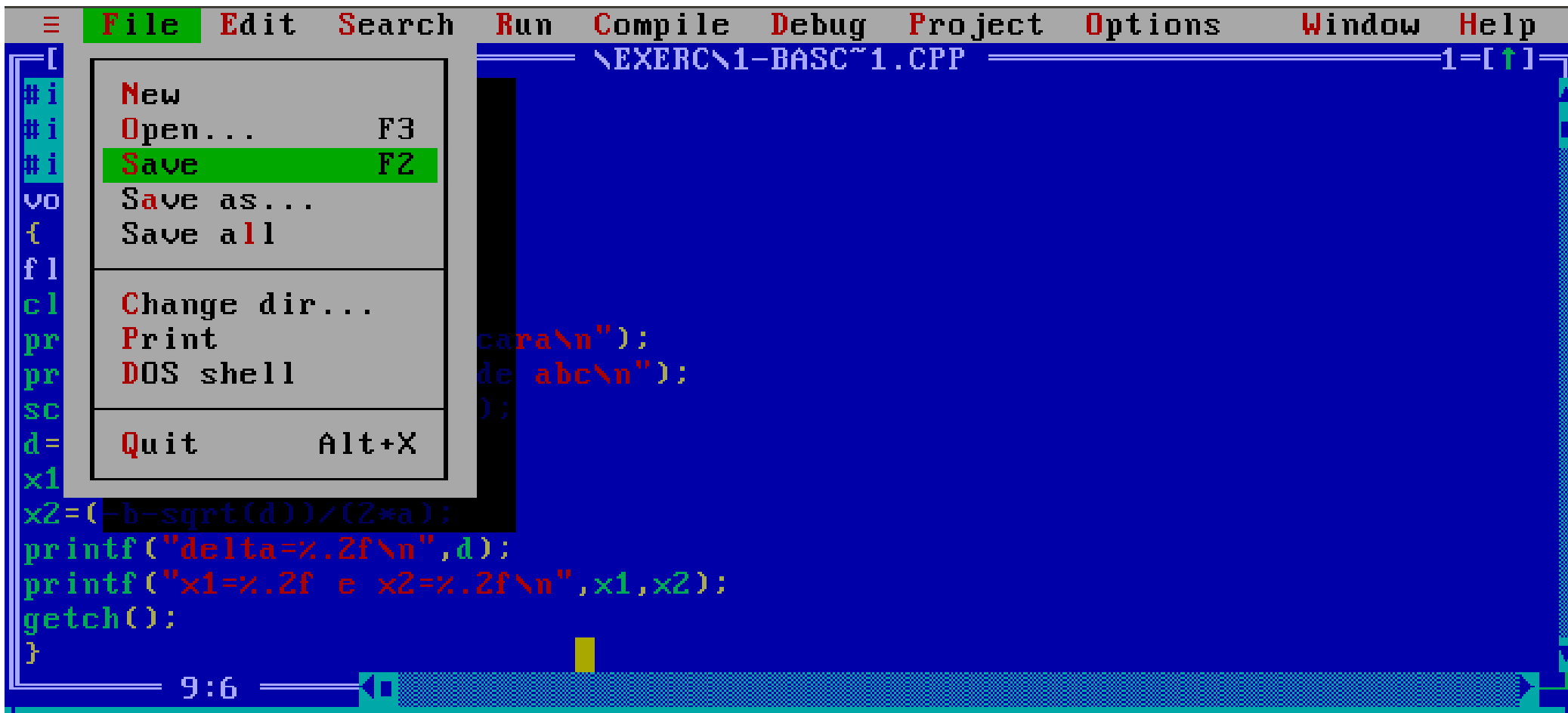
# Rodar o Código – ( Run )

```
programa de bascara  
digite o valor de abc  
1 4 1
```

# Rodar o Código – ( Run )

```
programa de bascara  
digite o valor de abc  
1 4 1  
delta=12.00  
x1=-0.27 e x2=-3.73
```

# Salvar arquivo – ( Save )

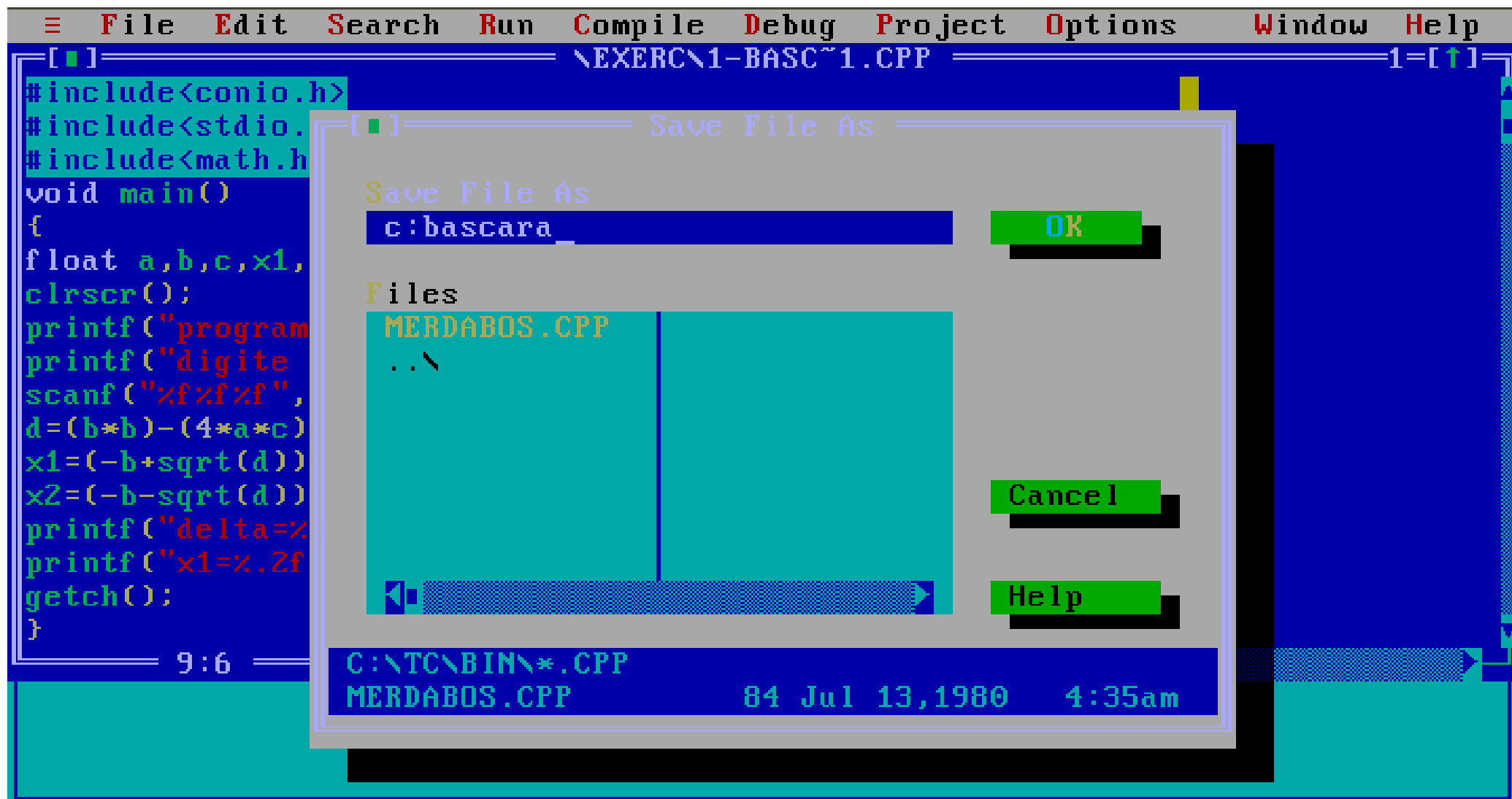


The screenshot shows the Turbo C++ IDE interface. The 'File' menu is open, and the 'Save' option is highlighted in green. The menu options are: New, Open... (F3), Save (F2), Save as..., Save all, Change dir..., Print, DOS shell, and Quit (Alt+X). The background code editor shows a C++ program for solving a quadratic equation. The status bar at the bottom indicates line 9, column 6.

```
#include <iostream>
using namespace std;

int main()
{
    float a, b, c, d, x1, x2;
    cout << "Digite os coeficientes a, b e c: ";
    cin >> a >> b >> c;
    d = b*b - 4*a*c;
    x1 = (-b + sqrt(d)) / (2*a);
    x2 = (-b - sqrt(d)) / (2*a);
    printf("delta=%.2f\n", d);
    printf("x1=%.2f e x2=%.2f\n", x1, x2);
    getch();
}
```

# Salvar arquivo – ( Save )



# Salvar arquivo – ( Save )

