Fases do desenvolvimento de um programa em linguagem C

- Codificação do algoritmo;
- Compilação do programa fonte;
- Linkagem dos objetos;
- Execução do código produzido.

Codificação

Nesta fase, todo o trabalho é realizado pelo programador, que criará o programa fonte com o auxílio de um editor de texto genérico, ou específico de um ambiente de desenvolvimento. Em geral, os ficheiros deverão ter a extensão **.c**, para poderem ser reconhecidos automaticamente pelo compilador como sendo arquivos contendo código fonte em C. Exemplo: **prog1.c**

Quando usamos o compilador Dev-C++ em ambiente PC, os programas fontes tem por defeito a terminação .CPP

Compilação

A tradução em linguagem C faz-se através de um compilador.

Se existirem erros de sintaxe no programa fonte - programa na linguagem de programação escolhida o compilador detetá-los-á e indicará a sua localização junto com uma breve descrição do erro. Se não existirem erros de sintaxe o compilador produzirá o código executável, ou programa objeto, já em linguagem máquina (código binário).

O compilador pode também detetar situações que não são erro de sintaxe, mas que levantam algumas suspeitas. Neste caso emite um aviso (Warning) para cada situação suspeita.

O compilador cria um ficheiro objeto, com o nome igual ao do programa e com extensão **obj** .

Linkagem

A fase de compilação serve apenas para a verificação sintática e para a criação do ficheiro objeto. O ficheiro executável é criado a partir do ficheiro objeto e através das bibliotecas (extensão **.lib**) que contém código já compilado das funções do próprio C (como printf( )) que são necessárias ao executável final (extensões **.exe**). O responsável por esta fase é o **Linker.**

Esta fase permite juntar num único executável, ficheiros objetos que tenham como origem compiladores de linguagens diferentes.

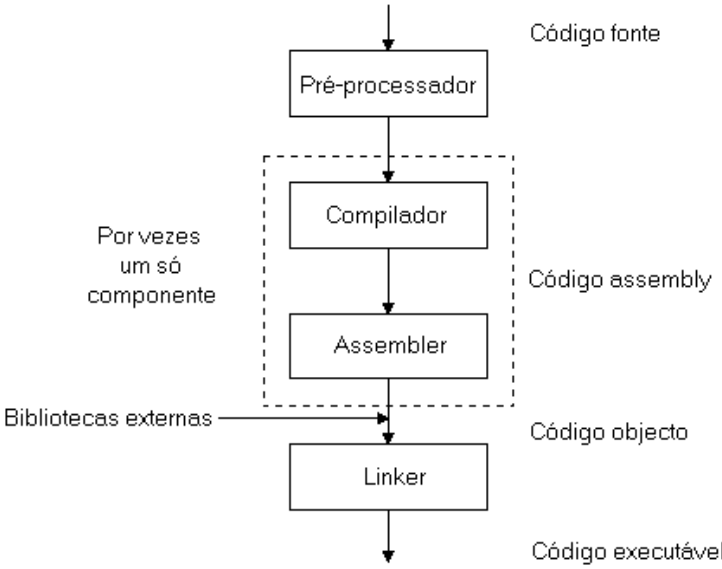
Execução

Se o programa fonte foi compilado com sucesso a execução do programa obtido faz-se invocando-o como se fosse um comando do sistema operativo.

Durante a execução podem detetar-se erros de lógica que fazem com que o programa não se comporte como esperado. Caso isso se verifique é necessário voltar à edição do programa fonte para corrigir esses erros, voltar a compilar para finalmente executar o programa.

O modelo de compilação da linguagem C

Neste esquema destacam-se apenas os seus pontos principais:



O assembler traduz código em linguagem assembly (texto) para código objeto. Pode estar integrado no compilador.

O pré-processador atua apenas ao nível do código fonte, modificando-o. Trabalha apenas com texto. Algumas das suas funções são: remover os comentários de um programa;

- Interpretar diretivas especiais a si dirigidas, que começam pelo caráter #.

Por exemplo:

- **#include** - insere o conteúdo de um ficheiro de texto no ficheiro corrente. Esses ficheiros são usualmente designados por cabeçalhos (*header files*) e têm a extensão.h, pois não têm código mas apenas os cabeçalhos das funções que representam:
    - **#include <math.h>** - Insere o conteúdo do arquivo math.h com a declaração das funções matemáticas da biblioteca standard.
    - **#include <stdio.h>** - Idem para as funções standard de entrada/saída.

    Estas linhas são diretivas que indicam ao pré-processador que deverá adicionar ao processo de compilação um ficheiro existente algures no disco do seu computador com o nome que aparece entre <>, para que o compilador tenha acesso a um conjunto de informações sobre as funções que virá a utilizar.

    - **#define** - define um nome simbólico cujas ocorrências no programa serão substituídas por outro nome ou constante:

    #define MAX 100 - substitui todas as ocorrências de MAX por 100.

Estrutura de um Programa em C

Um programa, escrito em linguagem de programação C, tem normalmente a seguinte estrutura:

```
# include <bibliotecas>
# define <funções de macros>
# define <constantes>

declaração de funções ou protótipos de funções;
definição de novos tipos de dados;
definição e declaração de variáveis globais;


main()
{

 definição e declaração de variáveis locais;
 corpo da função ou bloco operativo da função;
 }
tipo_de_dados identificador_da_funcao( argumentos)
{
   definição e declaração de variáveis locais;
    corpo da função ou bloco operativo da função;
}
```

Em resumo, a ordem será:

1. Comandos do pré-processador
2. Definições de tipos
3. Protótipos de funções - declaração dos tipos de retorno e dos tipos dos parâmetros das funções
4. Declaração de variáveis globais
5. O programa (sempre começado pela função main( )).
6. Funções

Programa exemplo

```
# include <stdio.h>                    // biblioteca de acesso
main ( )
{
printf (" Olá Turma!\n ");             // saída de dados
}
```

Algumas das bibliotecas internas

Ficheiro de cabeçalho	FINALIDADE
assert.h	Define a macro assert()
ctype.h	Manipulação de carateres
errono.h	Apresentação de erros
math.h	Possibilidade de utilização de definições matemáticas
stddef.h	Definir algumas constantes normalmente usadas
stdio.h	Suporte para entrada e saída de dados (Standard Input Output)
stdlib.h	Suporte para biblioteca padrão
string.h	Suporte para manipulação e tratamento de strings
time.h	Suporta as funções data e hora do sistema

As bibliotecas correspondem portanto a ficheiros e são constituídos por um conjunto de funções que se adicionam ao compilador.

Exercícios:

1. Qual a estrutura normal de um programa em linguagem C?
2. O que são bibliotecas?
3. Indica quais os ficheiros de cabeçalho que é necessário invocar para executar num programa operações de entrada/saída de dados e para manipular cadeias de carateres?
4. Qual a extensão de um programa fonte gerado em linguagem de programação C?
5. Qual a função que indica o início de um programa em C?
6. Qual a extensão de um ficheiro compilado?
7. Distingue entre programa fonte e programa objeto.