Programarea Calculatoarelor

Curs 4

Iulian Năstac

Recapitulare din cursul precedent

Sisteme de operare

- Pentru ca un calculator să poată fi folosit trebuie să existe iniţial un soft de bază care să permită conversaţia între un limbaj de programare (sau un soft specializat, etc.) şi procesorul sistemului de calcul. Acesta este de fapt ***sistemul de operare (SO)*** al calculatorului.

Recapitulare din cursul precedent

Definiţie:

Un sistem de operare (S.O.) este un set de proceduri care permit unui grup de utilizatori să folosească eficient şi eventual simultan sistemul de calcul (S.C.) avut la dispoziţie.

Recap.
Clasificare

- sistemele de operare pe loturi (**batch processing**);
- sistemele de operare cu multiprogramare (**multiprogramming**);
- sistemele de operare cu divizarea timpului (**time sharing**);
- sistemele de operare cu multiprelucrare (**multiprocessing**).

Recap
Alte clasificări :

- În timp real (Real-time)
- Pentru mai mulți utilizatori simultan (Multi-user)
- Multi-tasking / single-tasking
- Distribuite
- Încapsulate (Embedded)

Recap
Înscrierea informaţiei în unitatea de memorie de masă (HDD)

- Prin partiţionarea hard-disk-ului se segmentează unitatea în mai multe regiuni, numite *partiţii*, care pot conţine sistemul de fişiere al unui anume S.O.

- Un sisteme de fișiere separat (un SO separat chiar), poate fi utilizat pe fiecare partiție

- Sistemele de operare ale calculatoarelor utilizează unul din cele trei sisteme uzuale de fişiere:
  - File Allocation Table (FAT)
  - High Performance File System (HPFS)
  - New Technology File System (NTFS)

Softul instalat în memoria de masă a unui sistem de calcul

- ***Sistemul de operare (S.O.)***

- ***Utilitare***
  - soft pentru plăcile conectate pe magistrala I/O
  - editoare de texte
  - programe de administrare şi instrumente de depanare
  - programe de editare grafică, video, simulatoare, jocuri,etc.

- ***Limbaje de programare***

Limbaje de programare

Definiție:

Un limbaj de programare este un limbaj formal bazat pe instrucțiuni, care este proiectat pentru a putea pune în aplicare o serie de sarcini specifice.

Clase de limbaje de programare

- **De nivel scăzut** (limbajele de asamblare - cu instrucţiuni procesor)

- **De nivel ridicat**
- bazate pe interpretoare: BASIC, MATLAB, JAVA, unele programe de baze de date
- bazate pe compilatoare: FORTRAN, PASCAL, ADA, C, etc.

Interpretorul

- Un interpretor traduce codul sursă într-o reprezentare intermediară eficientă și apoi o execută imediat.

Compilatorul

- Un compilator transformă codul sursă scris într-un limbaj de programare (sursa) într-un cod obiect și, în cele mai multe cazuri, într-un program executabil.

O altă clasificare pentru limbajele de programare cuprinde trei categorii:

- **Limbaje de nivel înalt** (Ada, Pascal, Fortran, etc.)
  - limbaje de programare cu un nivel înalt de abstracție care nu depinde de detaliile unui anumit calculator
- **Limbaje de nivel mediu** (C, C++, FORTH, etc.)
- **Limbaje de nivel scăzut** (Macro-assembler; Limbaje de asamblare)
  - limbaje de programare ce depind puternic de detaliile unui anumit calculator

Limbajul de programare C

- 1966 Martin Richards (University of Cambridge) a dezvoltat **BCPL** (Basic Combined Programming Language)

- 1969 **Ken Thomson** ajutat și de Dennis Ritchie – **B programming language**

- 1969-1973 **Dennis Ritchie** – **C programming language**

Dezvoltarea limbajul de programare C

- Începutul anilor ’70 – codul S.O. UNIX este rescris în C
  - De atunci există întotdeauna un compilator C (C shell) încorporat în fiecare UNIX (chiar în unele SO înrudite cu UNIX).

- 1978 **Dennis Ritchie** și **Brian Kernighan** au elaborat împreună celebra carte "The C Programming Language".

Alte limbaje de programare bazate pe C

- C#, C++, Objective-C
- D
- Go
- Rust
- Java, JavaScript
- Limbo,
- LPC
- Perl
- PHP
- Python
- Verilog

Se impunea necesitatea unui standard ...

- Înainte de sfârșitul anilor '80, mulți utilizatori de C se bazau doar pe specificațiile cărții lui Dennis Ritchie și Brian Kernighan
- 1989 *American National Standards Institute* – *ANSI* – a publicat prima versiune a standardului pentru C ("**ANSI C**" sau "**C89**")
- 1990 - ISO emite standardul internațional (numit "C90").
- 1995 - ISO lasează o extensie a C-ului standard
- 1999 – un standard revizuit (cunoscut ca "**C99**")
- Decembrie 2011 – alta revizuire a standardului de C ("**C11**")
- 2017-2018 – ultima versiune a standardului ("**C18**")

Standardul C++

- 1998 C++ standard a fost ratificat ca ISO/IEC 14882:1998.
- 2003 – unele modificări au fost adăugate la ISO/IEC 14882:2003.
- 2011 – Standardul curent de C++ cu noi adăugiri a fost denumit ISO/IEC 14882:2011 (uneori denumit C++11)
- 2014 - Standardul C ++ 14 înlocuiește C ++ 11 cu caracteristici noi și o bibliotecă standard extinsă.
- 2017 – Specificațiile C ++ 17 au atins stadiul Draft International Standard (DIS) în martie 2017.
- Se prefigureaza un nou standard C++20 pana la sfarsitul lui 2020.

Portabilitatea

- **calitate a unui limbaj de programare, scris pentru o anumită mașină, de a putea fi utilizat pe o altă mașină.**

Portarea

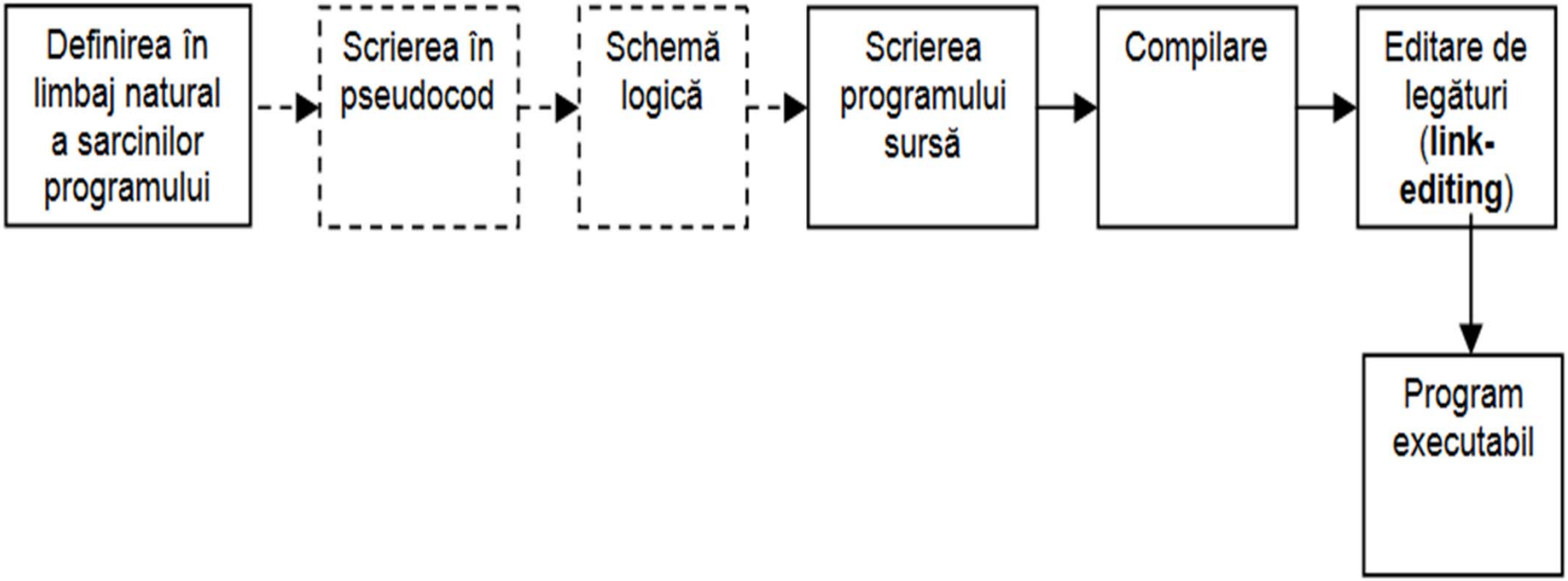
- **Portarea** este procesul de adaptare a unui software.

- Practic, cu cât mai mic este costul portării, relativ la costurile de implementare, cu atât mai portabil este acel soft.

Obs.

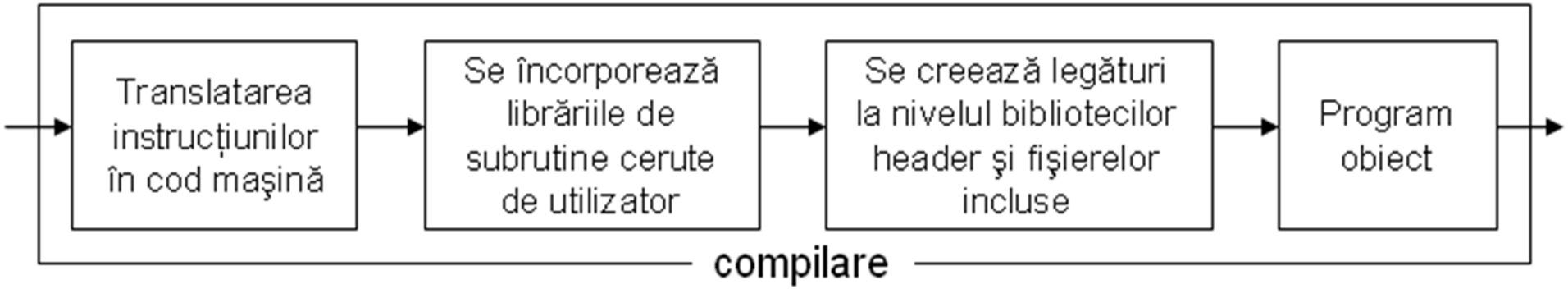
- Conceptul de portabilitate poate fi stabilit pentru diferite nivele de implementare:

  – Descrierea în pseudocod
  – Diagramă logică
  – Program sursă
  – Compilare
  – Link-editing
  – Program executabil



Un ***linke***r sau ***link editor*** este un program de calculator care preia unul sau mai multe fișiere obiect generate de un compilator și le combină într-un singur program executabil.

Compilatorul este un program complex care transforma instrucțiunile din limbajul sursă în limbaj mașină (cod de asamblare).



- Rezultatul este un program obiect.

- În cazul în care link-editorul este inclus în compilator, atunci rezultatul este un fișier executabil.

Caracteristici ale limbajului C

1. Portabilitatea
2. Tipurile de date
3. Controlul erorilor
4. Lucrează la nivelul limbajului assembler
5. Număr restrâns de cuvinte cheie (keywords)
6. Intră în categoria limbajelor structurate
7. Considerat un limbaj al programatorului

1. Portabilitatea limbajului C

- Conform ingineri software cu experiență, limbajul de programare C pare a fi cel mai portabil mediu pentru conceperea unui program.

2. Tipurile de date

- Există patru specificatori de bază pentru date:
  - char
  - int
  - float
  - double

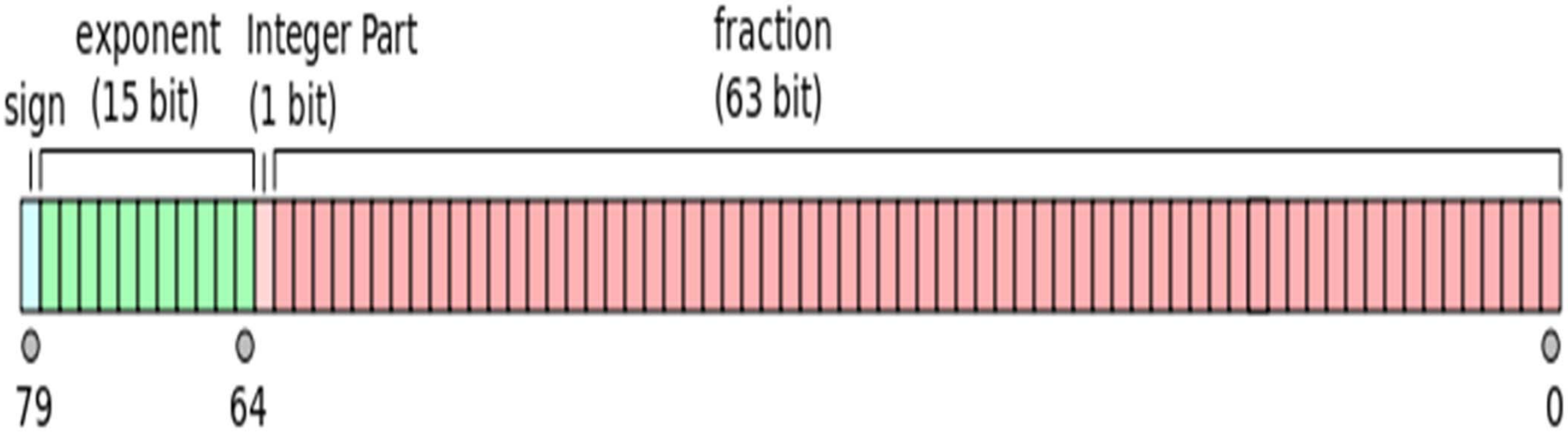
- Specificatori opționali/ suplimentari :
  - signed,
  - unsigned
  - short
  - long

Tip	Explicații
char	Cea mai mică unitate adresabila (8 biți), care poate conține setul de caractere de bază. Este un tip întreg. Poate fi cu sau fără semn.
signed char	La fel ca char, dar cu semn obligatoriu
unsigned char	La fel ca char, dar sigur fără semn.
short **short int** **signed short** **signed short int**	*Întreg scurt*. Lungime de cel puțin 16 biți
unsigned short **unsigned short int**	La fel ca short, dar fără semn
int **signed int**	Tipul întreg de bază. Lungime de cel puțin 16 biți (dar deseori dublu față de short)
unsigned **unsigned int**	La fel ca int, dar fără semn

Tip	Explicații
long **long int** **signed long** **signed long int**	*Întreg lung*. Lungime de cel puțin 32 biți
unsigned long **unsigned long int**	La fel ca long, dar fără semn.
long long **long long int** **signed long long** **signed long long int**	*long long* este un întreg cu semn de mare dimensiune. Lungime de cel puțin 64 biți (specificat odată cu versiunea standard C99).
unsigned long long **unsigned long long int**	La fel ca long long, dar fără semn. (specificat odată cu versiunea standard C99).

Tip	Explicații
float	Format în virgulă mobilă cu precizie simplă. Ocupă 4 octeți în memorie (32 biți).
double	Format în virgulă mobilă cu precizie dublă. Ocupă 8 octeți în memorie (64 biți).
long double	Format în virgulă mobilă cu precizie extinsă. Extended precision floating-point type. Ocupă 80-biți (IEEE 754 quadruple-precision floating-point format)

long double



Formatul virgulă mobilă de 80 de biți a fost disponibil pe scară largă după 1984.

Observații:

- Dimensiunea tipului întreg variază în funcție de implementarea compilatorului utilizat.
- Standardul prevede doar relații de mărime între tipurile de date și dimensiunile minime pentru fiecare tip de date.
- **long long** este mai cuprinzător decât **long**, care este mai cuprinzător decât **int**, care este mai cuprinzător decât **short**.

Observații:

- **char** este întotdeauna cel mai mic tip de date standard.
- Dimensiunea minimă pentru **char** este de 8 biți, dimensiunea minimă pentru **short** și **int** este de 16 biți, pentru **long** este de 32 biți, iar **long long** trebuie să conțină cel puțin 64 biți.
- Există o multitudine de conversii posibile în C.

3. Controlul erorilor

- Exceptând erorile de sintaxă, în C nu avem un alt tip de control.

- Nu există control pentru verificarea dimensiunii variabilelor utilizate.

4. Lucru la nivelul limbajului de asamblare

- Există posibilitatea de a lucra direct biți, octeți, cuvinte calculator și pointeri.

- Instrucțiunile în C necesită un număr minim, prin translatare, de instrucțiuni mașină în cadrul procesului de compilare.

5. Cuvintele cheie în limbajul **C**

auto	**double**	**int**	**struct**
break	**else**	**long**	**switch**
case	**enum**	**register**	**typedef**
char	**extern**	**return**	**union**
const	**float**	**short**	**unsigned**
continue	**for**	**signed**	**void**
default	**goto**	**sizeof**	**volatile**
do	**if**	**static**	**while**

Standardul C89

- Există doar 32 cuvinte cheie în primul standard ANSI C:
  - Din care 27 introduși deja în cartea Kernighan & Ritchie

- Multe alte limbaje de programare au cel puțin un număr dublu de cuvinte cheie.

Standardul C99 adaugă cinci cuvinte cheie noi:

_Bool

_Complex

_Imaginary

inline

restrict

Standardul C11 adaugă alte 7 cuvinte cheie :

_Alignas

_Alignof

_Atomic

_Generic

_Noreturn

_Static_assert

_Thread_local

6. Limbaj structurat

- Programare structurată este o paradigmă de programare care vizează îmbunătățirea (claritate, calitatea și timpul de dezvoltare) unui program de calculator prin utilizarea pe scară largă de subrutine, structuri bloc și bucle.

- Acest lucru este în contrast cu folosirea de teste și salturi, cum ar fi instrucțiunea **goto** care este atât de dificil de urmărit și de întreținut.

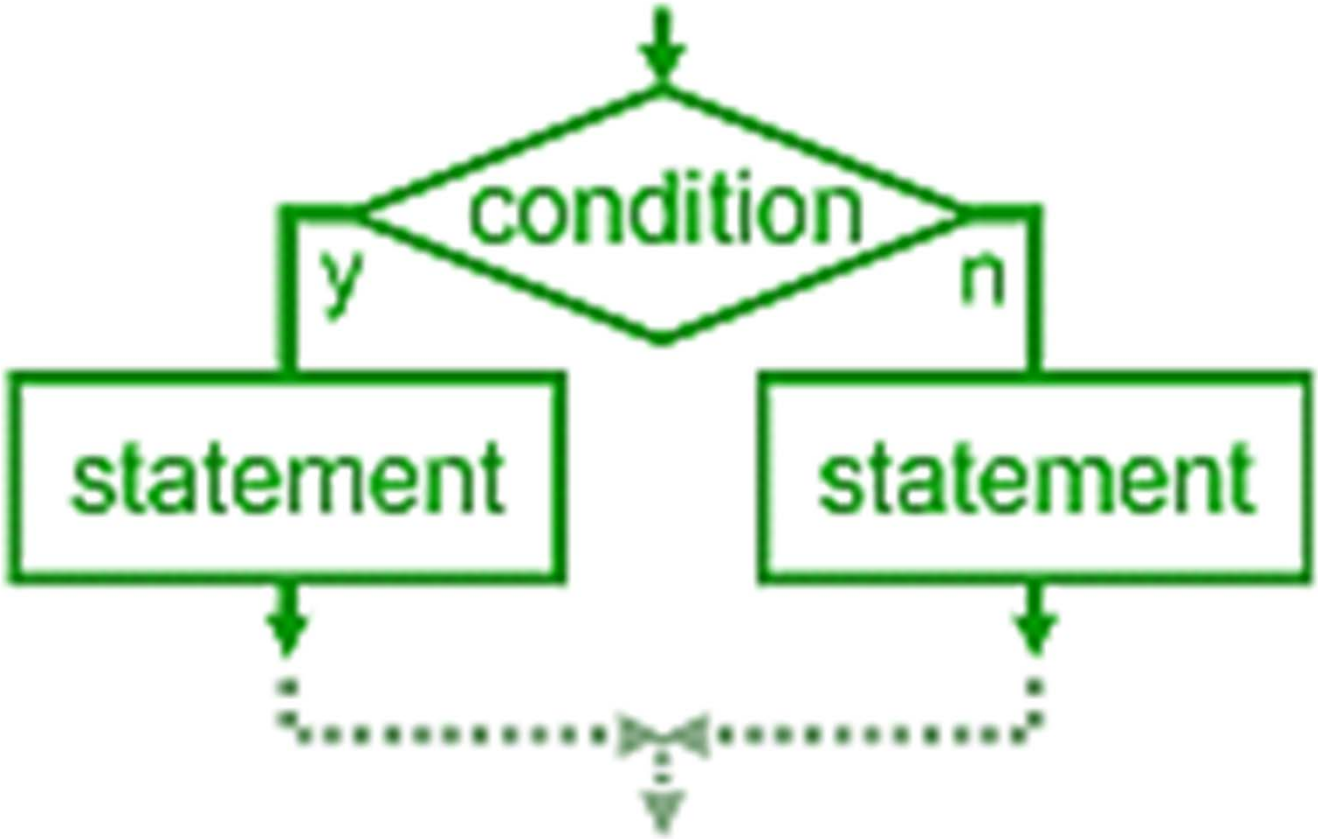
Observații:

- Programe structurate sunt adesea compuse din structuri înlănțuite ierarhic.

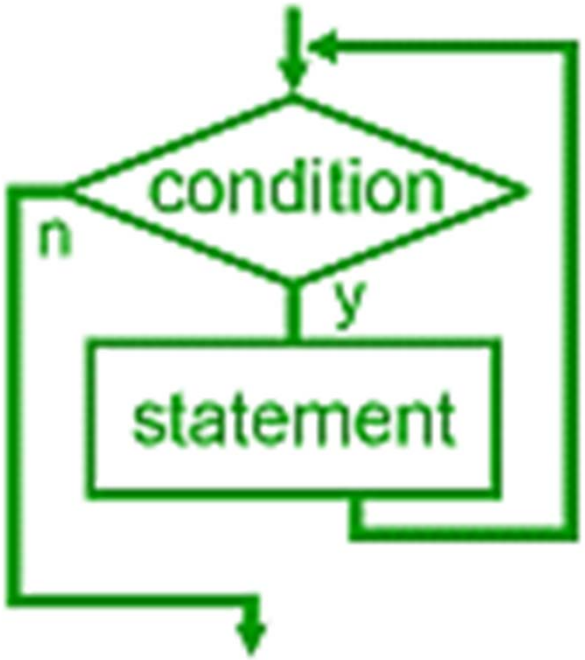
- Acestea sunt: secvența, selecția, și repetarea (iterația).

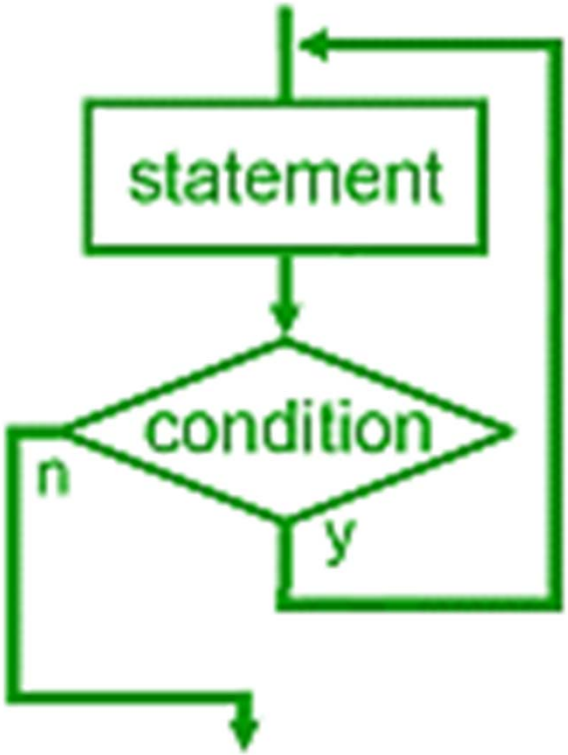
Secvență

Selecție



Iterare (bucle)





Notă:

- **compartimentarea** – facilitatea de separare și ascundere (față de restul programului) a întregii **informații** (serii de instrucțiuni) necesare pentru a îndeplini o anumită sarcină.

- Aceasta este o caracteristică a C-ului.

Observație:

- Principala componentă structurală în C este conceptul de funcție.

- Posibilitatea de a obține compartimentarea este de a utiliza un bloc de instrucțiuni grupate între acolade.

```
{
.....
}
```

7. Un limbaj al programatorului

- C-ul este adesea folosit pentru "programare de sistem", inclusiv punerea în aplicare a sistemelor de operare și a aplicațiilor de sisteme tip embedded.

- Un programator are nevoie de:
  - cod portabil și eficient
  - capacitatea de a accesa adrese RAM specifice
  - capacitatea de a se mula peste datele impuse din exterior pentru cerințele de acces
  - cererea scăzută asupra resurselor de sistem
- C este uneori folosit uneori ca un limbaj intermediar de implementare pentru alte programe scrise în diferite alte limbaje.

Structura unui program scris în C

- Declarații globale:
  - Includerea de fișiere header
  - declararea de constante și variabile globale
  - declararea de funcții locale

- Funcția **main()**

- Alte funcții

Observații:

- **Cuvintele cheie (keywords)** sunt scrise cu litere mici

- Un program scris în C trebuie să conțină o singură funcție **main** (și numai una).

- Bibliotecile standard ale C și C++, ce conțin diverse funcții standard se află în fișierele **header**.

Fișierele Header (antet)

- Fiecare fișier **header** conține declarații pentru una sau mai multe funcții, definiții de tipuri de date și macro-uri.

Notă: Unele fișiere **header** noi s-au adăugat de fiecare dată la apariția unui nou standard imbunatățit.

Câteva fișiere **header** de bază

<stdio.h>	Definește funcțiile de intrare/ieșire
<stdlib.h>	Definește funcții de conversie numerică, funcții de generare numere pseudo-aleatoare, de alocare de memorie, funcții de control de proces, etc.
<string.h>	Definește funcțiile de prelucrare/tratare pentru șiruri de caracter.
<math.h>	Definește funcții matematice comune.

Preprocesare în C

- Un program sursă C poate fi prelucrat înainte de a fi supus compilării. Această prelucrare poartă numele de preprocesare.

- Preprocesorul asigură:
  - includeri de fişiere cu texte sursă;
  - definiţii şi apeluri de macrouri;
  - compilare condiţionată.

Includeri de fişiere

```
#include <stdio.h>
int main(void)
{       printf(„Salut!\n");
        return 0;
 }
```

- Preprocesorul înlocuiește linia
- #include <stdio.h> cu conținutul fișierului 'stdio.h', care conține și funcția printf() printre multe alte funcții

Definiţii şi apeluri de macrouri

- Definirea unei constante:

```
#define PI 3.14159
```

- Definirea unei funcții macro:

```
#define ABS(a)   (a<0) ? –a : a
```

Compilarea condiţionată

- Compilarea condiționată permite compilatorului să producă diferențe în executabilul obținut în conform cu unii parametri.

- Această tehnică este frecvent utilizată atunci când este nevoie de aceste diferențe pentru a rula software-ul de pe platforme diferite, sau cu diferite versiuni de biblioteci necesare (eventual pe un hardware diferit).

Directiva if-else

- Variante ale directivei:

  #if

  #ifdef

  #ifndef

  #else

  #elif

  #endif

poate fi folosită pentru compilare condiționată.