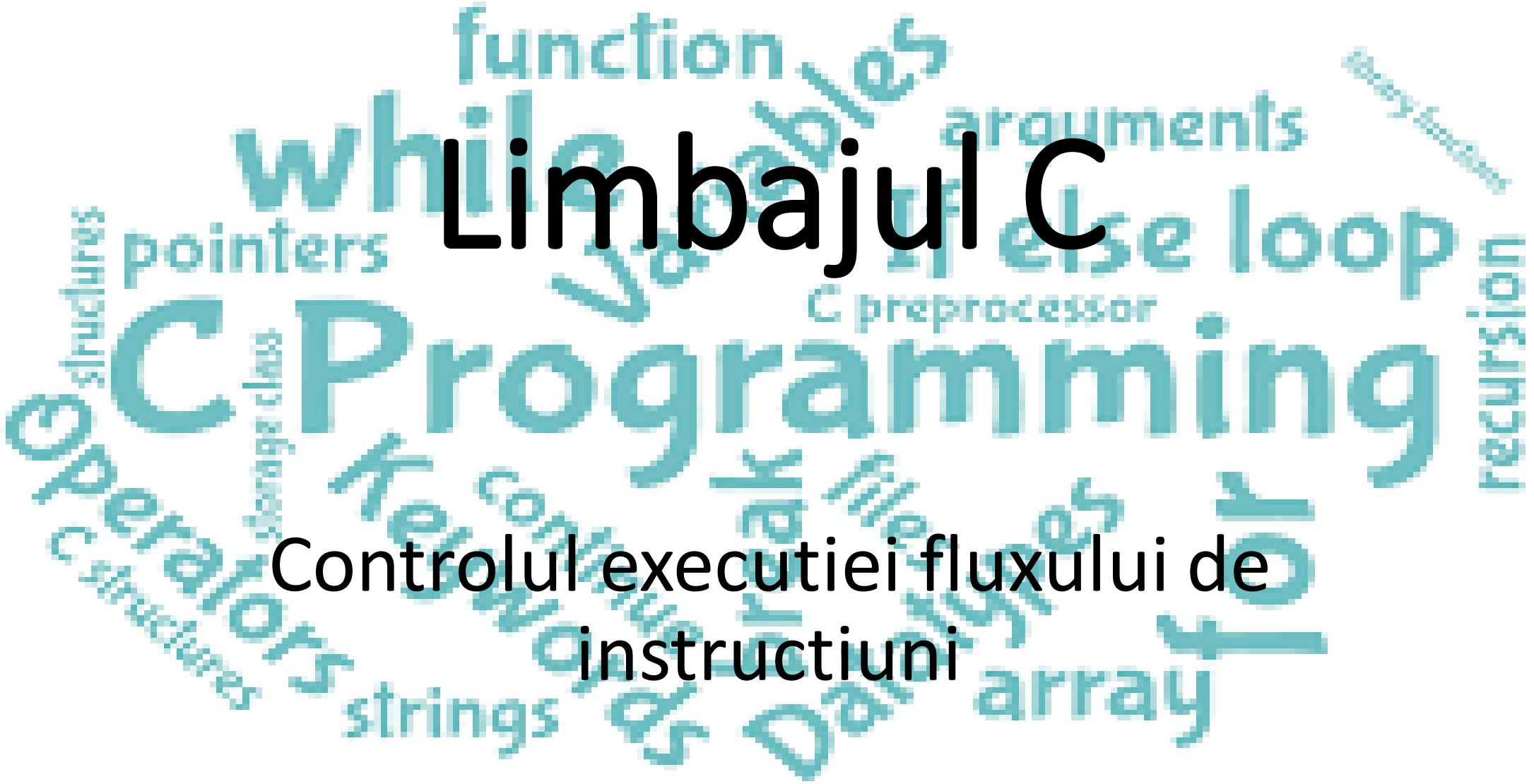


Limbajul C

Controlul executiei fluxului de instructiuni

Cuprins

- Instructiuni si blocuri
- Structuri alternative: instructiunile *if-else* si *switch*
- Structuri repetitive: instructiunile *for* si *while*
- Structuri repetitive de tip *do-while*
- Intructiunea *break*
- Instructiunea *continue*
- Instructiunea *goto*

Controlul fluxului

- Instructiunile de control al fluxului dintr-un limbaj specifica ordinea in care se fac calculele;
- Instructiuni si blocuri:

```
x = 0;
i++;
printf(...);
```

- Atentie la “;” – reprezinta terminatorul de instructiune
- { } sunt folosite pentru a grupa instructiuni si declaratii intr-un bloc
- Un bloc este sintactic echivalent cu o singura instructiune;
- Nu se pune ; dupa un bloc!!!

Controlul fluxului

- Un bloc de instructiuni:

```
{
    X = 0;
    Y = 5;
    Z = X + Y;
}
```

Controlul fluxului: if-else

- Instructiunea if-else este folosita pentru luarea de decizii;
- Ea implementeaza structura alternativa;
- Sintaxa:

```
if(expresie)
        instructiune1;
else
        instructiune2;
```

- Partea *else* este optionala;

Controlul fluxului: if-else

- Deoarece un "if" testeaza pur si simplu valoarea numerica a unei expresii, sunt posibile anumite prescurtari de cod:

  if(expresie)

  in loc de :

  if(expresie != 0)

- Deoarece partea cu "else" a unui if-else este optionala, se poate ajunge la o ambiguitate cand se omite un else dintr-o secventa imbricata de if-uri;
- Aceasta este rezolvata asa: else este asociat cu if-ul anterior cel mai apropiat, care nu face pereche cu un alt “else".

Controlul fluxului: if-else

- Exemplul 1:

```
if (n > 0)
  if (a > b)
        z = a;
  else
        z = b;
```

- In acest caz else face pereche cu if(a>b)

- Exemplul 2:

```
if (n > 0)  {
      if (a > b)
            z = a;
}
else
      z = b;
```

- In acest exemplu, else face pereche cu primul if, if(n>0);

Controlul fluxului: if-else

- Ambiguitatea este poate conduce la interpretari gresite, indeosebi in situatii ca urmatoarea (in acest caz, else face pereche cu cel mai interior if):

```
if (n > 0)
      for (i = 0; i < n; i++)
            if (s[i] > 0)  {
                  printf("...");
                  return(i);
            }
else   /* se face o presupunere gresita! */
      printf("error - n is zero \n");
```

Controlul fluxului: if-else

Structurile if pot fi dispuse „in cascada“ atunci cand testam mai multe conditii:

```
if (expresie1)
     instructiune1;
else if (expresie2)
     instructiune2;
else if (expresie3)
     instructiune3;
else
     instructiune4;
```

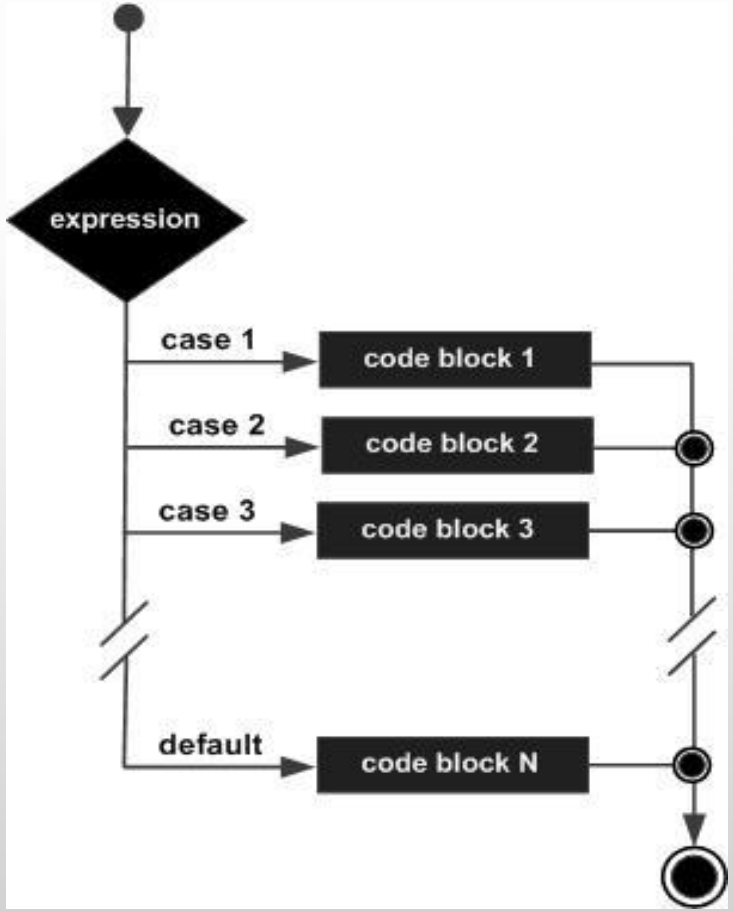
Controlul fluxului: switch

Pentru a compara valoarea unei expresii cu mai multe constante, in loc sa folosim o serie de instructiuni if imbricate, putem folosi instructiunea *switch* care are sintaxa urmatoare:

```
switch(expresie) {
   case constanta1:
      instructiuni1;
      break; /* optional */

   case constanta2:
      instructiuni2;
      break; /* optional */

 case constanta3:
      instructiuni2;
      break; /* optional */

   /* oricate case*/
   default : /* optional */
   intructiuni_n;
}
```

Controlul fluxului: switch

- Schema logica a unei instructiuni switch arata ca in figura urmatoare:



Controlul fluxului: switch

- Instructiunea *switch* este este folosita pentru decizii multiple, care testeaza daca o expresie se potriveste cu una dintr-un numar de valori constante;
- In exemplul urmator se testeaza daca valoarea variabile ***c*** de tip char este o cifra, un caracter “alb” sau altceva:

```
char c;
c = getchar();
switch(c) {
            case '0':
            case '1':
            case '2':
            case '3':
            case '4':
            case '5':
            case '6':
            case '7':
            case '8':
            case '9':
                ndigit[c-'0']++;
                break;
            case ' ':
            case '\n':
            case '\t':
                nwhite++;
                break;
            default:
                nother++;
                break;
}
```

Controlul fluxului: Bucle while si for

```
while(expresie)
      instructiune;

for(expr1; expr2; expr3)
     instructiune;
```

- Acest ***for*** este echivalent cu:

```
expr1;
while(expr2)   {
     instructiune1;
     expr3;
}
```

Controlul fluxului: Bucle while si for

- Bucla for este clar superioara atunci cand exista o initializare simpla si se cunoaste de la inceput numarul de executii ale instructiunii din corpul buclei, deoarece ea pastreaza instructiunile de control al buclei impreuna si la loc vizibil in varful buclei.
- Acest lucru este exemplificat aici:

```
for (i = 0; i < N; i++)
```

- Se foloseste in special la prelucrarea masivelor de date (tablouri/vectori/matrici);

Controlul fluxului: Bucle while si for

- Deseori, in limbajul C, in instructiunea for se foloseste un operator special: “,” – operatorul *virgula*!
- O pereche de expresii separate printr-o virgula este evaluata de la stanga spre dreapta iar tipul si valoarea rezultatului sunt tipul si valoarea operandului din dreapta;
- Exemplu de utilizare a virgulei intr-un for – o functie care inverseaza un sir de caractere:

```
reverse(char s[])   /* inverseaza pe loc sirul s */
{
      int c, i, j;
      for (i = 0, j = strlen(s) - 1; i < j; i++, j--)  {
            c = s[i];
            s[i] = s[j];
            s[j] = c;
      }
}
```

Controlul fluxului: Bucle while si for

- Atentie!
- Virgulele care separa argumentele functiilor, variabilele din declaratii, etc. nu sunt operatori "virgula" si nu garanteaza evaluarea de la stanga la dreapta;

```
int x,y,z;
```

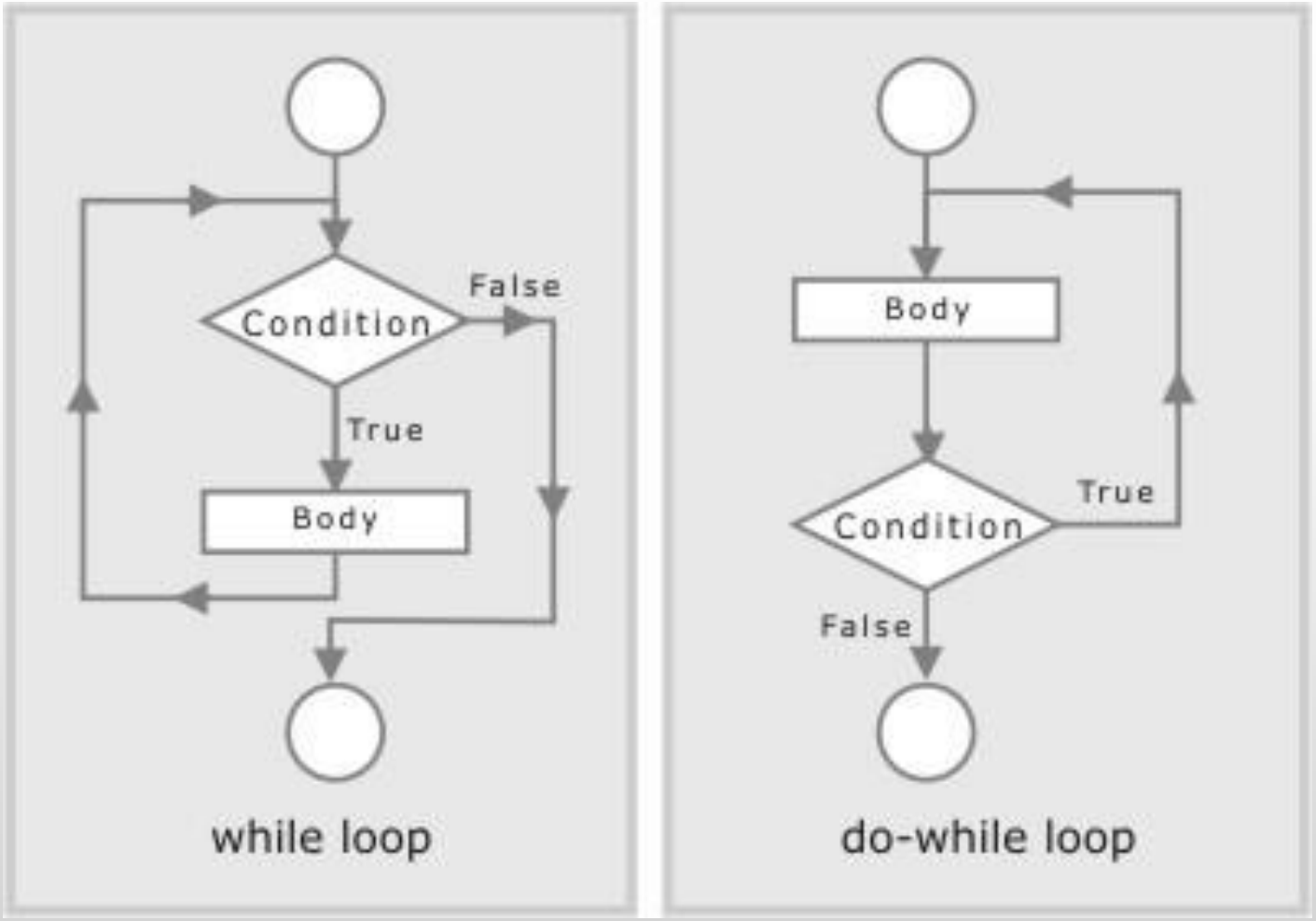
```
func(a, b, c);
```

Controlul fluxului: Bucle do while

- Bucla do-while testeaza conditia la sfarsit, dupa ce a executat intreg corpul buclei;
- Spre deosebire de while, corpul buclei este executat cel putin o data!
- Sintaxa:

```
do
   instructiune;
while(expresie);
```

Controlul fluxului: comparatie while : do-while



Controlul fluxului: break

- Adesea este convenabil sa controlam iesirile din bucle altfel decat testand conditia la inceputul sau sfarsitul buclei;
- Instructiunea ***break*** ofera o iesire mai devreme din for, while, do si switch.
- O instructiune ***break*** face ca bucla (sau switch-ul) cea mai din interior sa se termine imediat;

Controlul fluxului: continue

- ***continue*** face sa inceapa urmatoarea iteratie a buclei (while, for, do);
- In cazul lui ***while*** si ***do-while*** aceasta inseamna ca partea de test se executa imediat;
- In cazul lui for, controlul se trece la faza de reinitializare;

```
for (i = 0; i < N; i++)  {
     if (a[i] < 0)     /* sari elementele negative */
          continue;
      .../*  prelucreaza elementele pozitive */
}
```

Controlul fluxului: goto

- Limbajul C ofera instructiunea ***goto*** si etichete pentru ramificare;
- Formal, ***goto*** nu este necesara niciodata si in practica este aproape intodeauna usor sa scriem cod fara ea;

```
for (...)
     for (...)  {
          ...
          if (dezastru)
              goto error;
     }
     ...

error:
     descurca beleaua
```

Controlul fluxului: Goto

- Exemplu:

```
for (i = 0; i < N; i++)
    for (j = 0; j < M; j++)
        if (v[i][j] < 0)
            goto found;
 /* nu s-a gasit */
 ...
found:
/* s-a gasit un element negativ la pozitia i,j */
...
```