Programare procedurală
– Laborator 3 –

I) Funcţii de citire/scriere pentru caractere

1) Citire/scriere de caractere

În aceste funcţii, caracterele apar ca şi unsigned char convertite la int- au fie valoare 0 .. 255, fie EOF = sfarşit de fişier, definit ca -1 (EOF introdus de la tastatură: Ctrl+D (UNIX) sau Ctrl+Z (DOS)):

a) Citirea unui caracter de la intrare

```
int getchar(void);
```

Observaţie:

- Returnează caracterul citit sau EOF.

b) Tipărirea unui caracter la ieşire

```
int putchar(int c);
```

Observaţie:

- returnează caracterul tipărit, sau EOF in caz de eroare

Observaţii generale:

- Citirea/scrierea caracter cu caracter şi cea formatată pot fi amestecate liber în program - fiecare continuă de unde s-a oprit precedenta.
- **Nu** sunt standard C: `conio.h, getch(), getche(), clrscr()`. **Nu** folosiţi pentru operaţiunile de intrare/ieşire uzuale.

2) Citire/scriere de şiruri de caractere

Observaţii generale:

- La citirea datelor de intrare: utilizatorul poate introduce orice, deci trebuie să ne protejăm de date (ne)intenţionat eronate.
- Utilizatorul poate introduce mai multe caractere decât memoria alocată - corupe memoria, termină programul, probleme de securitate.
- **Nu** folosiţi gets()! **Nu** folosiţi scanf("%s", sir)! Pentru o citire corectă şi sigură, folosiţi limitări in scanf:

-citirea unui caracter: `char c; scanf("%c", &c);`
-citirea mai multor caractere: într-un tablou (şir), în limitele acestuia:

- un număr fix de caractere: `char s[80]; scanf("%80c", s);` orice caractere, inclusiv spaţii albe; nu se adaugă automat '\0'
- un cuvânt (orice până la spaţiu alb): `char s[80]; scanf("%79s", s);` ignoră spaţii albe iniţiale; adaugă '\0' la sfarşit
- o linie de text, până la '\n': `char s[80]; fgets(s, 80, stdin);` citeşte max. 80-1 caractere, inclusiv '\n', adaugă '\0' (stdin: identificator definit in stdio.h pt. fişierul standard de intrare).

II) Directive de preprocesare

a) Includerea fișierelor cu texte sursă

Exemple:

```
#include<string.h>
#include "fisier.h"
#include "fisier.c"
```

b) Definirea de constante/simboluri

Exemple:

```
#define pi 3.14159
……
#undef pi
```

```
#define separatori {'*', '#', '%', '!', '+', '$', '=', '-'}
char sir[]=separatori;
```

c) Macroinstrucțiuni

Exemple:

```
#define produs(a,b) a*b
```

Macro-uri multiline

```
1. #define Aparitii(v,n,x) { \
2. int nr=0;\
3. printf("Dati valoarea cautata: "); \
4. scanf("%d",&x); \
5. for(i=1;i<=n;i++) if(v[i]==x) nr++; \
6. printf("Numarul de aparitii: %d",nr); \
7. }
```

Observații:

1. Folosirea macro-ului definit: `Aparitii(w,k,z)` (w,k,z declarate/citite anterior).
   Nu `Aparitii(w,20,15).`
2. Necesită atenție.
   Exemplu: `produs(a+c,b+d)` va genera în pasul de preprocesare expresia `a+c*b+d`.
   Soluție: `#define produs(a,b) (a)*(b)`
3. Utile. *Exemplu:* modificări ulterioare sunt necesare într-un singur punct în program

```
1.  #define DIM_MAX 100
2.  int main()
3.  {
4.        int v[DIM_MAX],w[DIM_MAX],…
5.        for(int i=0;i<DIM_MAX;i++) ……
```

4. Discuție macro-uri vs. constante:
   - *scoping* (nu poți defini un scope pentru un macro);
   - *debug* (în preprocesare are loc o simplă înlocuire textuală petru macro-uri – greu de urmărit);
   - *adresare* (macro-urile nu au adrese, nu ocupă memorie);
   - *siguranța tipului de date* (nu poți specifica tipul de date pentru un macro).

d) Instrucțiuni de compilare condițională

```
1. #ifdef constată/macro
2. ……
3. #endif
```

```
1. #ifndef constată/macro
2. ……
3. #endif
```

Exemplu:

```
6. #ifndef pi
7. #define pi 3.14159
8. #endif
```

```
1. #if constantă1/condiție1
2.   ……
3. #elif constantă2/condiție2
4.   ……
5. #elif constantă3/condiție3
6.   ……
7.
8. #else
9.   ……
10. #endif
```

Exemple:

```
1. //#define initializare(a,b,c) a=1; b=2; c=3;
2. ……
3. int main()
4. {  int x,y,z;
5. #ifndef initializare
6.     scanf("%d",&x);
7.     scanf("%d",&y);
8.     scanf("%d",&z);
9.     printf("%d   %d   %d",x,y,z);
10.
11. #else
12.    initializare(x,y,z);
13.    printf("%d",x+y+z);
14. #endif
15. ……
```

```
16. #ifdef TURBOC
17.     #define INT_SIZE 16
18.  #else
19.    #define INT_SIZE 32
20. #endif
```

```
1. #if SYSTEM==WIN32 || SYSTEM==_WIN32 || SYSTEM==__WIN32 || SYSTEM==__WIN32__
2. ……
3. #else  ……
4. #endif
```

```
1. #if defined(_WIN64)
2. ……
3. #elif defined(_WIN32)
4. ……
5. #endif
```

Exemple de macro-uri predefinite:
https://gcc.gnu.org/onlinedocs/cpp/Common-Predefined-Macros.html#Common-Predefined-Macros

e) Alte instrucțiuni

```
1. #pragma expresie
2. #error mesaj_de_eroare
3. #line numar_de_linie [nume_nou_fisier]
```

III) Tipuri de date structurate

În C se pot defini tipuri sructură **(struct)**, enumerare **(enum)** și uniune **(union)**.

1. Struct

```
1. [typedef] struct [nume_tip_nou] {
2.       tip_de_date  camp_1;
3.       tip_de_date  camp_2;
4.       ……….
5.       tip_de_date  camp_n;
6. } [lista_identificatori];
```

Exemple:

```
1. struct angajat
2.       {
3.           char cnp[14], nume[50], data_ang[11],post_ocupat[30];
4.           float salariu;
5.           int zile_concediu, nr_copii;
6.       };
7.  int main()
8.  {
9.       struct angajat a1;
10.      scanf("%d",&a1.zile_concediu);
11.      ……
```

```
1. typedef struct
2.       {
3.            char cnp[14], nume[50], data_ang[11],post_ocupat[30];
4.            float salariu;
5.            int zile_concediu, nr_copii;
6.       } angajat;
7.  int main()
8.  { angajat a1;
9.   angajat a2={"1841211305600","Radu M.","11.12.2013","Analist",5300.45,20,2};
10.  printf("%d",a1.zile_concediu);
11.      ……
```

```
1. typedef struct
2.       {
3.            char nume[50];
4.            int semigrupa;
5.       } student;
6.  int main()
7.  {
8. student s[3]={{"Ion A.",1331},{"Vlad R.",1341},{"Dinu E.",1432}};
9.   printf("%d",s[2].semigrupa);
10.      ……
```

Observații

1. Câmpurile pot fi de orice tip de date, dar nu de tip structură (recursiv);
2. Structuri diferite pot conține câmpuri identice (fără conflict);

3. Structurile pot fi transmise/returnate în funcții;
4. Nu se pot compara folosind operatori logici.

2. **Va urma: enum, union.**

Probleme

Creați și folosiți fișierul antet ***cod.h,*** în care să includeți funcțiile definite pe parcursul rezolvării problemelor.

1. Se citește un caracter de la tastatură. Să se verifice dacă este literă mare. Dacă da, să se transforme în literă mică și să se afișeze. Altfel, să se rescrie caracterul tastat.

2. Se citesc de la tastatură construcţii de forma „a operator b”, unde a și b sunt numere întregi, iar operatorul poate fi „+”, „-”, „*”, „/”, „%”. Să se afișeze valoarea expresiei citite. Să se folosească instrucţiunea decizională switch în rezolvarea problemei. Pentru cazul în care operatorul este „/”, să se verifice dacă împărţitorul este diferit de 0 (în cazul care este 0, se va afișa un mesaj corespunzător). De asemenea, să se afișeze un mesaj corespunzător în cazul în care operatorul nu este unul din cei enumeraţi.

3. a) Să se construiască o structură ce conține următoarele date despre candidații la admitere: nr_legitimatie, nume, nota_mate, nota_info, nota_bac, medie, admis (Y/N), buget (Y/N).
b) Sa se defineasca o macroinstrucțiune ce calculează media de admitere dupa regula: 80% media la examen, 20% media de bac.
c) Sa se defineasca o constanta pentru pragul minim de promovabilitate egala cu 5.
d) Sa se scrie o funcție care citește datele unui candidat, in afara de medie, admis si buget, si le adauga unui vector al tuturor candidatilor, pastrand ordinea alfabetica. Media si promovabilitatea vor fi calculate folosind definitiile de la punctele b) si c). Numarul de candidati este citit de la tastatura.

- *se va folosi functia **strcmp**(s1,s2) ce returneaza un numar:*

  o negativ, daca s1 este mai mic decat s2 dpdv al continutului;
  o zero, daca s1 este identic cu s2;
  o pozitiv, daca s1 este mai mare decat s2 dpdv al continutului.

e). Să se scrie o funcție care completeaza campul „buget” cu Y sau N dupa regula: primii 75% (rotunjit in jos) dintre candidatii admisi, in ordinea mediilor, sunt la buget (Y), restul la taxa (N) sau nu au promovat examenul de admitere (lasati campul gol).
f) Sa se scrie o functie care afiseaza datele candidatilor in functie de optiunea aleasa: toti candidatii (alfabetic), cei admisi la buget, cei admisi la taxa, cei respinsi (ordonati descrescator dupa medie). (meniu cu switch)

4. Să se construiască o structură de date potrivită pentru a memora o matrice rară (matice de dimensiune *nxm, 1<=n,m<=50000,* numărul elementelor nenule *1<=k<=100*). Să se scrie câte o funcție pentru adunarea și înmulțirea a două matrici rare. Elementele matricei se vor citi ca triplete (l,c,x), unde l=linie, c=coloană, x=elementul nenul, în ordine crescătoare după linie și apoi după coloană. Afișarea se va face ca matrice (pe linii și coloane).

5. Să se construiască o structură de date potrivită pentru a memora un polinom (*1<=coeficient<=1.000.000, 0<=putere<=50*). Datele se vor introduce crescător după puterile lui X. Să se termine produsul a două polinoame.

Observații:

- Nu se vor folosi alte functii de lucru cu siruri de caractere în afara de strcmp și strcpy;
- Nu se vor folosi pointeri;
- Toate afisarile trebuie sa contina mesaje corespunzatoare.