Enumerari

- o enumerare e o multime de constante de tip **intreg** care specifica toate valorile permise pe care le poate avea o variabila de acel tip
- atat numele enumerarii cat si lista de variabile sunt **optionale** – dar nu simultan
- valoarea constantei asociata unui element al enumerarii e fie implicit, fie explicit specificata

Sintaxa:

enum *nume_tip* {
lista constante (enumerare)
} lista variabile;

- implicit primului element din enumerare i se asociaza valoarea 0; iar pentru restul elementelor - valoarea elementului precedent incrementata cu 1, daca nu e specificata alta

enum E{x, y, z}; → x = 0, y = 1, z = 2

enum X {a, b, c=10, d}; → a = 0, b = 1, **c = 10, d = 11**

enum {a=4, b=6 , c=-1, d = -4} ex; → a = 4, b = 6, c = -1, d = - 4

enum boolean {fals, adevarat}; → fals=0, adevarat=1

```
enum zile_sapt {
        luni=1, marti, miercuri, joi, vineri, sambata, duminica
} azi, maine;

azi = joi;  //o variabila de tip enumerare poate lua orice valoare enumerata.
cout<<azi<<endl; //4

maine=(zile_sapt)(azi+1); // maine=(zile_sapt)5;
cout<<maine<<endl;   //5
cout<<duminica<<endl;  //7
```

OBS: Fiind de tip int, variabilele de acest tip se pot folosi ca indici in tablouri.

Aplicatie

Definiti tipul de date student.

Un Student are:

- nume
- CNP
- grupa: 311, 312, 313, 314
- note luate la materiile de pe semestrul 1: Analiza, Algebra, Fizica, Programare, Engleza.

Implementati functii pentru citirea si afisarea datelor unui student si, respectiv, calculul mediei pe semestrul 1.

```
#include <string.h>

enum Materie { Analiza, Algebra, Fizica, Programare, Engleza };

enum Grupa { Gr_311 =1 , Gr_312, Gr_313 , Gr_314};

struct stud {
        char *nume = NULL;
        char CNP[14];
        Grupa gr;
        int note[5]; // putem folosi indici de tip Materie
};

typedef struct stud Student;
```

```
// functie care citeste datele unui student
void citesc(Student *s) {
   printf("\nDati numele: ");
   char aux[100];
   scanf("%s", aux);
   free(s->nume); /*pot elibera in siguranta doar daca nume e initializat cu NULL*/
   s->nume = (char*)malloc(sizeof(char) * (strlen(aux)+1));
   strcpy(s->nume, aux);
   printf("\nDati CNP-ul: ");
   scanf("%s",s->CNP);
   int g;
   printf("\nDati grupa (intre 1 si 4): ");
   scanf("%d",&g);
   s->gr = (enum Grupa) g;
   printf("\nDati notele pt Analiza, Algebra, Fizica, Programare, Engleza: ");
   for (int i=Analiza; i<=Engleza; i++ )
      scanf("%d", &s->note[i]);
}
```

```
// functie care afiseaza datele unui student.
void afisez(Student s)  //pot sa il transmit pe s si prin referinta constanta, e chiar indicat
{
   printf("\nNumele: %s", (s.nume!=NULL)?s.nume:" ");
   printf("\nCNP-ul: %s", s.CNP);
   printf("\nGrupa: ");
   switch (s.gr)
   {
   case 1:   printf("311");   break;
   case 2:    printf("312");   break;
   case 3:    printf("313");   break;
   case 4:    printf("314");   break;
   }
   printf("\nNotele la Analiza, Algebra, Fizica, Programare, Engleza sunt: ");
   for (int i=Analiza; i<=Engleza; i++ )
      printf(" %d", s.note[i]);
}
```

```
//functie care calculeaza media unui student
double getMedia(Student s) //pot sa il transmit pe s si prin referinta constanta, e chiar indicat
{
   int media =0;
   for (int i=Analiza; i<=Engleza; i++ )
      media+=s.note[i];
   return (double)media/5;
}

int main()
{
   Student s = {NULL,"",1};
   citesc(&s);
   afisez(s);
   printf("\nMedia: %.2f", getMedia(s));
   return 0;
}
```