10. Alocarea dinamică a memoriei.

10.1. Funcţii pentru gestiunea dinamică a memoriei.

Utilizatorul poate solicita în timpul execuţiei programului alocarea unei zone de memorie. Această zonă de memorie poate fi eliberată, în momentul în care nu mai este necesară. Alocarea şi eliberarea de memorie la execuţie permite gestionarea optimă a memoriei.

Biblioteca standard oferă 4 funcţii, având prototipurile în **<alloc.h>** şi **<stdlib.h>.** Acestea sunt:

```
void *malloc(unsigned n);
```

Funcţia alocă un bloc de memorie de **n** octeţi. Funcţia întoarce un pointer la începutul zonei alocate. În caz că cererea de alocare nu poate fi satisfăcută, funcţia returnează **NULL**.

```
void *calloc(unsigned nelem, unsigned dim);
```

Alocă **nelem*dim** octeţi de memorie (**nelem** blocuri formate din **dim** octeţi fiecare). Întoarce un pointer la începutul zonei alocate sau **NULL**.Memoria alocată este iniţializată cu zerouri.

```
void free(void *p);
```

Funcţia eliberează o zonă de memorie indicată de **p**, alocată în prealabil prin **malloc()** sau **calloc()**.

```
void  *realloc(void *p, unsigned dim);
```

Modifică dimensiunea spaţiului alocat prin pointerul **p**, la **dim**. Funcţia întoarce adresa zonei de memorie realocate, iar pointerul **p** va adresa zona realocată.

- dacă dimensiunea blocului realocat este mai mică decât a blocului iniţial, **p** nu se modifică, iar funcţia va întoarce valoarea lui **p**.
- dacă **dim==0** zona adresată de **p** va fi eliberată şi funcţia întoarce **NULL**.
- dacă **p==NULL**, funcţia alocă o zonă de **dim** octeţi (echivalent cu **malloc()**).

Funcţiile de alocare întorc pointeri generici **(void*)** la zone de memorie, în timp ce utilizatorul alocă memorie ce păstrează informaţii de un anumit tip. Pentru a putea accesa memoria alocată, indirect, prin intermediul pointerului, acesta va trebui să fie un pointer cu tip, ceea ce impune conversia explicită (prin cast) a pointerului întors de funcţia de alocare într-un pointer cu tip.

De exemplu, pentru a aloca un vector de întregi, având **n** elemente vom folosi:

```
int *p;
if (p=(int*)malloc(n*sizeof(int))==NULL) {
  printf("Memorie insuficienta\n");
  exit(1);
}
```

Funcţiile care întorc pointeri sunt utile în alocarea de spaţiu pentru *variabile dinamice*. Variabilele dinamice sunt alocate în momentul execuţiei, nu au nume şi sunt accesate prin pointeri.

Un *constructor* este o funcţie care alocă spaţiu în mod dinamic pentru o variabilă şi întoarce un pointer la spaţiul rezervat.

Un *destructor* este o funcţie care primeşte un pointer la o zonă alocată dinamic şi eliberează această zonă.

O funcţie utilă, **strdup()** – salvează un şir de caractere într-o zonă alocată dinamic şi întoarce un pointer la acea zonă sau **NULL**.

```
char *strdup(char *s){
   char *p;
   p=(char*) malloc(strlen(s)+1);
   if (p!=NULL)
      strcpy(p,s);
   return p;
}
```

Exemplul 29: ***Citiţi de la intrarea standard un şir de caractere de lungime necunoscută într-un vector alocat dinamic. Alocarea de memorie se va face progresiv, în incremente de lungime*** **INC**, ***după citirea a*** **INC** ***caractere se face o realocare.***

```
char *citire()
{ char *p, *q;
  int n;
  unsigned dim=INC;
  p=q=(char*)malloc(dim);
  for(n=1; (*p=getchar())!='\n' &&*p!=EOF; n++) {
    if(n%INC==0) {
      dim+=INC;
      p=q=realloc(q,dim);
      p+=n;
      continue;
    }
    p++;
  }
  *p='\0';
  return realloc(q,n);
}
```

Fişierul <string.h> ne pune la dispoziţie o serie de funcţii care ne permit să lucrăm cu zone de memorie care conţin date de tip nespecificat. Acestea vor fi considerate octeţi (caractere):

Tabel 10.1. Funcţii pentru lucru cu zone de memorie cu conţinut nespecificat

void* memcpy(void* d,const void* s, int n)	copiaza **n** octeţi din **s** în **d**; întoarce **d**
void* memmove(void* d,const void* s,int n)	ca şi **memcopy**, folosită daca **s** şi **d** se întrepătrund
void* memset(void* d,const int c, int n)	copiază caracterul **c** în primele **n** poziţii din **d**
int memcmp(const void* d,const void* s, int n)	compară zonele adresate de **s** şi **d**

Probleme rezolvate.

1. O şcoală are **nc** clase (**nc >12**, datorită existenţei claselor paralele). Clasele paralele sunt identificate suplimentar cu o majusculă (de exemplu 8A, 8B, etc). Clasele au număr diferit şi cunoscut de elevi . Numele elevilor din fiecare clasă sunt de asemenea cunoscute. Dându-se un nume, să se stabilească în ce clasă este acesta (de exemplu 9H). Datele problemei sunt:

- un şir de 12 caractere, în care caracterul **i** specifică ultima dintre clasele paralele **i+1** (de exemplu dacă **s[4]** este **C** atunci avem 3 clase a 5-a: 5A, 5B şi 5C).

- mai mulţi întregi specificând numărul elevilor din fiecare clasă

- numele elevului căutat

- numele celorlalţi elevi, introduse, câte unul pe linie, în ordinea claselor.

Menţionăm că nu se dă explicit **nc** – numărul total de clase. Alocarea de memorie pentru tablouri se face dinamic.

1. citire şir ultime clase
2. determinarea numărului de clase **nc**
3. citire nume elev căutat
4. citirea numărului de elevi din fiecare clasă
5. alocare memorie pentru păstrare nume elevi
6. citire nume elevi
7. căutare elev şi determinare număr clasă
8. formare nume clasă pe baza numărului
9. afişare nume elev şi nume clasă

2. Să se întocmească şi să se afişeze un index de cuvinte, pe baza liniilor citite de la tastatură. Acesta este un tablou (alocat dinamic, întrucât nu îi cunoaştem dimensiune de la început), în care sunt puse toate cuvintele distincte din text de lungime > **lgmin**. Fiecare cuvânt va fi însoţit de numărul său de apariţii, şi de numerele liniilor în care a apărut. Dacă un cuvânt apare de mai multe ori într-o linie, numărul de apariţii va fi actualizat corespunzător, dar numărul liniei va apare o singură dată.

Valoarea **lgmin** se citeşte înaintea textului. Între cuvintele din text pot exista ca separatori spaţii albe, virgulă, punct două puncte şi punct-virgulă **.**

1. citire lgmin, iniţializare contor linii
2. buclă citire linii
    buclă separare cuvinte din linie
        verificare lungime cuvânt separat
        căutare cuvânt în index
            dacă e găsit
                actualizare nr.apariţii
                eventuala actualizare nr.linii
            dacă nu e găsit
                creerea unei noi intrări în index
                copiere cuvânt
                actualizare nr. apariţii
                actualizare nr. linii
3. afişare tablou index

10.2. Probleme propuse.