# TABLOURI CU TIP DE DATE CHAR ÎN C

# String

# DECLARAREA


```
#include <stdio.h>
int main(){
        char ch1[] = “Probleme cu String”;
        printf(“%s\n”, ch1);
        ch1[20] = “Probleme cu String”;
        printf(“%s\n”, ch1);
        ch1[10] = “Probleme cu String”;
        printf(“%s\n”, ch1);
        ch1[] = {‘P’,’r’,’o’,’b’,’l’,’e’,’m’,’e’,’ ‘,’c’,’u’,’ ‘,’S’,’t’,’r’,’i’,’n’,’g’};
        printf(“%s\n”, ch1);

return 0;
}
```

# CITIREA & AFIȘAREA

```
#include <stdio.h>
int main(){
        char ch[] = “Probleme cu String”;
        printf(“%s\n”, ch);
        scanf(“%s”, ch);   // input: Probleme cu String
        printf(“%s\n”, ch);
return 0;
}
```

# CITIREA & AFIȘAREA

```
#include <stdio.h>
int main(){
        char str[20], ch;
        int i = 0;
        do {
        ch = getchar();
        str[i] = ch;
        ++i;
        } while (ch != '\n');
        str[i-1] = '\0';
        printf("%s\n", str);
return 0;
}
```


// input: Probleme cu String

# CITIREA & AFIȘAREA

- scanf("%s",str);
- printf("%s", str);
- gets(str); - citeşte de la terminalul standard un şir de caractere terminat cu linie noua (enter, \n); are ca parametru adresa zonei de memorie în care se introduc caracterele citite și va returna adresa de început a zonei de memorie. //va citi toate caracterele introduse, indiferent de memoria alocată

- fgets(str, len_max, stdin); - citeşte de la terminalul standard un şir de caractere terminat cu linie noua (enter, \n), dacă lungimea lui este mai mică decât lungimea maximă adimisă șirului, în caz contrar, va citi doar primele n-1 caractere, unde n este lungimea maximă admisă. În cazul în care șirul citit va avea o lungime mai mică decât cea maximă, înaintea terminatorului de șir(\0) în zona de memorie va fi reținut caracterul new-line(\n).
- puts(str); - afişeaza la terminalul standard şirul de caractere din zona data ca parametru, până la caracterul terminator de şir (\0), în locul căruia va afișa caracterul sfârșit de linie; are ca parametru adresa zonei de memorie de unde începe afişarea caracterelor și va returna codul ultimului caracter (diferit de \0) din şirul de caractere afişat sau -1 dacă a aparut o eroare.

# CITIREA & AFIȘAREA

```
#include <stdio.h>
int main(){
        char str[10];
        gets(str);
        puts(str);
        fgets(str, 10, stdin);
        puts(str);
return 0;
}
```


// input: Probleme cu String

# CONCATENAREA A 2 ȘIRURI

```
#include <stdio.h>
#include <string.h>
int main(){
        char str[20] = "\0", str1[] = "Probleme", str2[] = " cu String";
        strcat(str, str1);
        puts(str);
        strcat(str, str2);
        puts(str);

return 0;
}
```

# CONCATENAREA A 2 ȘIRURI

```
#include <stdio.h>
#include <string.h>
int main(){
    char str[256] = "\0", str1[100], str2[100];
    unsigned int len;
        fgets(str1, 100, stdin);
        len = strlen(str1);       // calculează lungimea sirului str1
        strncat(str, str1, len-1);       // concatenează str cu str1
        fgets(str2, 100, stdin);
        strncat(str, str2, strlen(str2)-1);
        puts(str);
        len = strlen(str);
        printf("len_str: %u", len);
return 0;
}
```

# NUMĂRĂ APARIȚIILE LUI ”A”

```
#include <stdio.h>
#include <string.h>
int main(){
        char str[50], ch = ‘A';
        unsigned int len, i, count = 0;
        fgets(str, 50, stdin);
        len = strlen(str);
        for (i = 0; i < len-1; ++i){
                if (str[i] == ch) count++;    }
        printf("Caracterul %c se repeta de %u ori", ch, count);
return 0;
}
```

# EXISTENȚA LITERELOR MARI

```
#include <stdio.h>
#include <string.h>
int main(){
        char str[50];
        unsigned int len, i, count = 0;
        fgets(str, 50, stdin);
        len = strlen(str);
        for (i = 0; i < len-1; ++i){
                if ((str[i] >= 'A') && (str[i] <= 'Z')) count++;    }
        if (count != 0) {printf("Sirul contine litere mari");}
                else {printf("Sirul nu contine litere mari");}
return 0;
}
```

# SUMA CIFRELOR UNUI NUMAR

```
#include <stdio.h>
#include <string.h>
int main(){
        char str[50];
        unsigned int len, i, sum = 0;
        fgets(str, 50, stdin);
        len = strlen(str);
        for (i = 0; i < len-1; ++i){
                if ((str[i] >= '0') && (str[i] <= '9')) sum += (str[i]-48);    }
        printf("Sum = %u", sum);
return 0;
}
```

# STRTOK & STRTOK_R

```
#include <stdio.h>
#include <string.h>
int main(){
        char str[50];
        fgets(str, 50, stdin);
        char *token;
        char* rest = str;
                /* token = strtok(str, " ");
                while( token != NULL ) {
                printf( " %s\n", token );
                token = strtok(NULL, " ");   }  */
        while ((token = strtok_r(rest, ".", &rest)))
        printf("%s\n", token);
return 0;
}
```

# POINTER - DECLARAREA

```
#include <stdio.h>
int main(){
        //int *p;
        //int* p;
        int * p;
        int num;
        scanf("%i", &num);
        printf("num = %i\n", num);
        printf("p = %p\n", &num);
return 0;
}
```

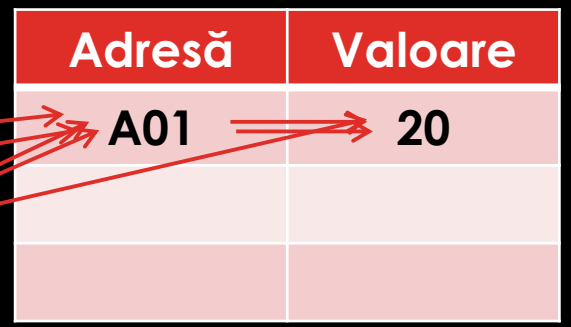
# ALOCAREA ADRESEI UNUI POINTER

```
#include <stdio.h>
int main(){
        int *p;
        int num;
        p = &num;
        printf("&p = %p\n", p);
        printf("&num = %p\n", &num);
return 0;
}
```

| Adresă | Valoare |
|---|---|
| A01 | |
| | |
| | |

# CITIREA VALORII UNUI POINTER

```
#include <stdio.h>
int main(){
        int *p;
        int num;
        p = &num;
        scanf("%i", &num);
        printf("num = %i\n", num);
        printf("p = %i\n", *p);
return 0;
}
```



# DE REȚINUT!

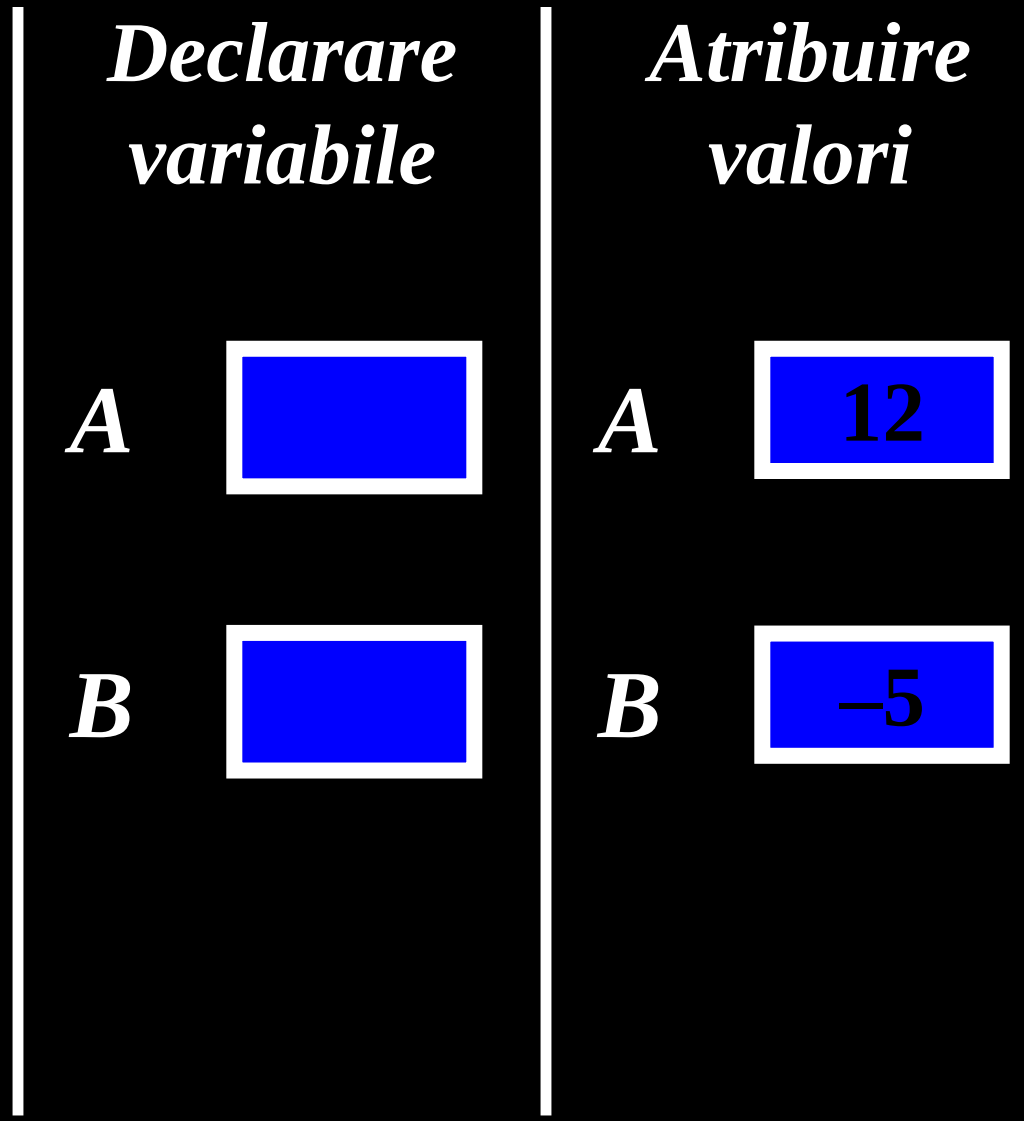
int *p;
int a, b;

p, &a, &b  indică adresa;

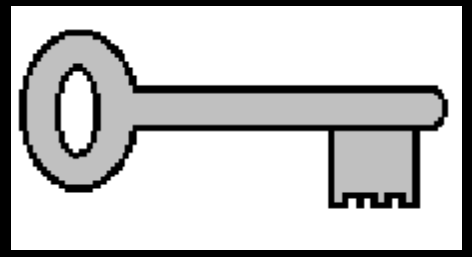
*p, a, b  indică valoarea stocată în memorie.

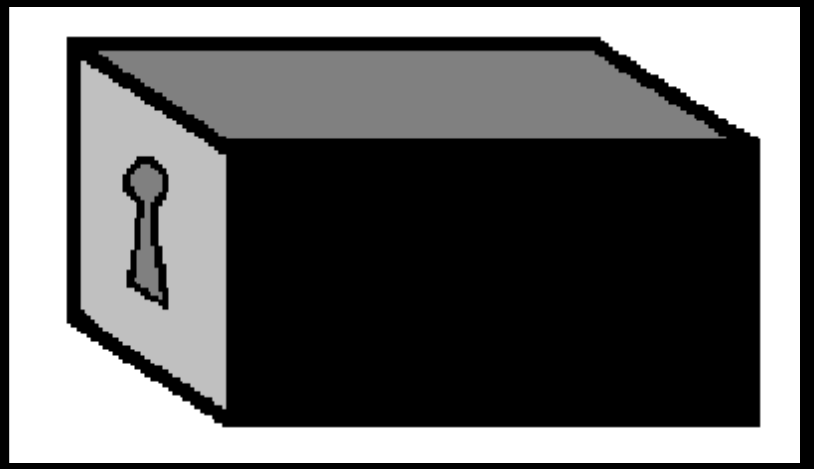
```
#include <stdio.h>
int main(){
        int A, B;
        scanf("%i", &A);
        scanf("%i", &B);

return 0;
}
```



# POINTER



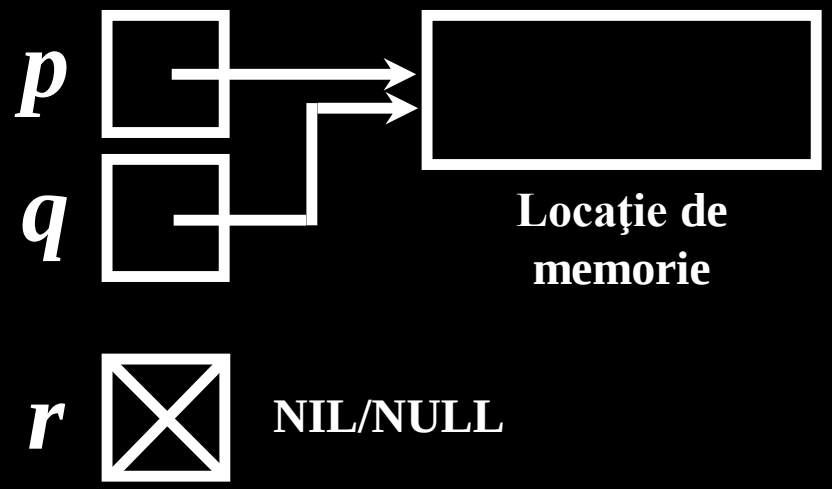


**Pointer / adresă**

**Valoarea**

# POINTERI

```
#include <stdio.h>
int main(){
        int p, *q, *r;
        q = &p;
return 0;
}
```



# POINTERI

```
#include <stdio.h>
int main(){
        int r, *p, *q;
        int num;
        p = &num;
        scanf("%i", &num);   // se citeste 9
        *p = 5;
        printf("num = %i\n", num);
        printf("p = %i\n", *p);
return 0;
}
```

# POINTERI

```
#include <stdio.h>
int main(){
        int r = 0, *p, *q, num;
        p = &num;
        *p = 7;
        p = &r;
        printf("num = %i\n", r);
        printf("p = %i\n", *p);
return 0;
}
```

# POINTERI

```
#include <stdio.h>
int main(){
        int r = 0, *p, *q, num;
        *p = 7;
        printf("num = %i\n", r);
        printf("p = %i\n", *p);
return 0;
}
```

# POINTERI

```
#include <stdio.h>
int main(){
        int r = 0, *p, *q, num;
        p = &num;
        p = 2;
        q = &r;
        printf("num = %i\n", *q);
        printf("p = %i\n", *p);
return 0;
}
```