```c
#include <cstdio>

int X,Y,Z;

int main(){

    int x=1,y=2,z=3;
    printf("int x: %d \n",x);
    printf("int y: %d \n",y);
    printf("int z: %d \n",z);

    printf("\n");
    printf("&x: %X \n",&x);
    printf("&y: %X \n",&y);
    printf("&z: %X \n",&z);

    int *p;
    int * q;
    int* r;

    p=&x;
    q=&y;
    r=&z;

    printf("\n");
    printf("atribuit p=&x \n");
    printf("atribuit q=&y \n");
    printf("atribuit r=&z \n");

    printf("\n");
    printf("int* p: %X \n",p);
    printf("int* q: %X \n",q);
    printf("int* r: %X \n",r);

    printf("\n");
    printf("&p: %X \n",&p);
    printf("&q: %X \n",&q);
    printf("&r: %X \n",&r);

    *p=15;
    *q=16;
    *r=17;

    printf("atribuit *p=15 \n");
    printf("atribuit *q=16 \n");
    printf("atribuit *r=17 \n");

    printf("\n");
    printf("x: %d \n",x);
    printf("y: %d \n",y);
    printf("z: %d \n",z);

    int * pointerLaAdresaZero = NULL;
    //*pointerLaAdresaZero=333;  //eroare


    int * pointer;

    pointer = new int;

    printf("\n");
    printf("pointer: %X\n",pointer);
    *pointer = 333;
    printf("valoarea de la adresa din
pointer: %d\n",*pointer);

    printf("\n");
    printf("adresa globalei X: %X\n",&X);
    printf("adresa globalei Y: %X\n",&Y);
    printf("adresa globalei Z: %X\n",&Z);
    return 0;
}
```

```
int x: 1
int y: 2
int z: 3

&x: 62FEF4
&y: 62FEF0
&z: 62FEEC

atribuit p=&x
atribuit q=&y
atribuit r=&z

int* p: 62FEF4
int* q: 62FEF0
int* r: 62FEEC

&p: 62FEE8
&q: 62FEE4
&r: 62FEE0
atribuit *p=15
atribuit *q=16
atribuit *r=17

x: 15
y: 16
z: 17

pointer: 800DE0
valoarea de la adresa din pointer: 333

adresa globalei X: 41E008
adresa globalei Y: 41E00C
adresa globalei Z: 41E010

Process returned 0 (0x0)   execution
time : 0.027 s
Press any key to continue.
```

```c
#include <cstdio>

struct pereche{
    int a,b; //alaturare de doi int
```

```
};

struct lista{
    int info;
    lista *next; //alaturare de int si
pointer
} *prim; //pointer catre un primul
element dintr-o lista noua, global, deci
are valoarea 0

void parcurgere(lista *pointer){
    printf("%d ",(*pointer).info);
//folosind * ne ducem in memorie la
structura, apoi luam primul camp cu .
    //printf("%d ",pointer->info);
//alternativ, sintaxa cu -> ne permite
acelasi lucru
    if ((*pointer).next!=NULL)

    parcurgere(pointer->next);
    //parcurgere((*pointer).next);
    //parcurgere((&(*pointer))->next);
    //parcurgere((*(&(*pointer))).next);
    //....
}

void insert_at_begin(int x){
    lista *new_lista= new lista;
//alocare spatiu
    new_lista->info=x; //atribuire camp1
cu int
    new_lista->next=prim; //atribuire
camp2 cu lista*
    prim=new_lista; // atribuire prim cu
lista*
}

int main(){

    pereche T,U,V;
    T.a=4;
    T.b=6;

    printf("\n%d %d",T.a,T.b);

    printf("\n &T: %X",&T);
    printf("\n &U: %X",&U);
    printf("\n &V: %X",&V);

    printf("\n\n &T.a: %X",&T.a);
    printf("\n &T.b: %X",&T.b);
    printf("\n\n &U.a: %X",&U.a);
    printf("\n &U.b: %X",&U.b);
    printf("\n\n &V.a: %X",&V.a);
    printf("\n &V.b: %X",&V.b);
    printf("\n");

    lista L,M,N;
    L.info=1;
    L.next=&M;
    M.info=2;
    M.next=&N;
    N.info=3;
    N.next=NULL;

    parcurgere(&L);

    printf("\n");

    insert_at_begin(7);
    insert_at_begin(8);
    insert_at_begin(9);

    parcurgere(prim);

    return 0;
}
```

```
4 6
 &T: 62FEF8
 &U: 62FEF0
 &V: 62FEE8

 &T.a: 62FEF8
 &T.b: 62FEFC

 &U.a: 62FEF0
 &U.b: 62FEF4

 &V.a: 62FEE8
 &V.b: 62FEEC
1 2 3
9 8 7
Process returned 0 (0x0)   execution
time : 0.059 s
Press any key to continue.
```