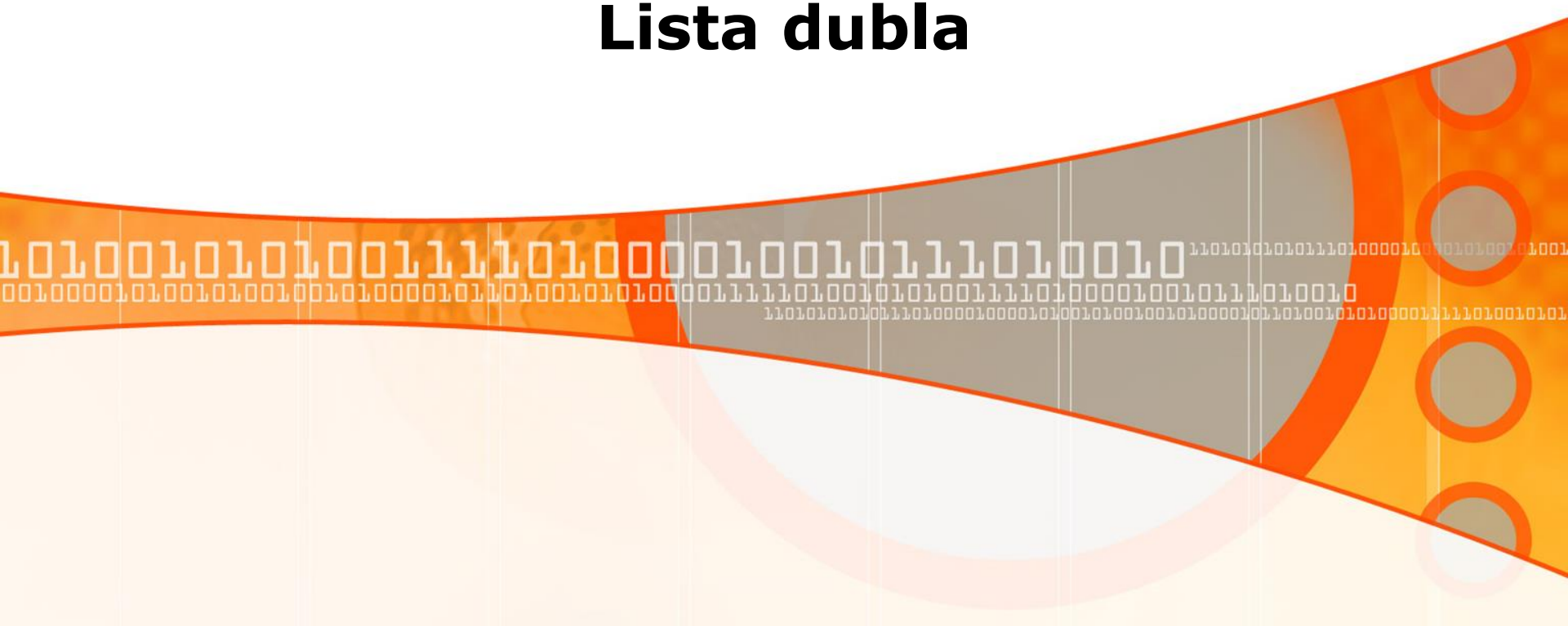
STRUCTURI DE DATE

Lista simpla
Lista dubla



LISTA LINIARA

Lista liniara:

- **Colectie de elemente denumite *noduri*;**
- **Relatie de ordine rezultata din pozitia nodurilor;**
- **Elemente de acelasi tip – structura omogena;**
- **Lungimea listei – numarul de noduri;**
- **Asemanatoare structurii de tip masiv unidimensional;**
- **Structura alocata in heap – gestionata prin variabila pointer.**

Lista liniara vs. Vector:

Lista liniara	Vector
Numar variabil de elemente, fara declararea anticipata a dimensiunii	Numar predeterminat de elemente, cu rezervarea de memorie pe dimensiunea declarata
Elemente alocate la adrese nu neaparat consecutive	Spatiu contiguu de memorie (elementele sunt adiacente logic si fizic)
Alocare memorie la executie	Alocare memorie la compilare/executie
Referire secventiala a nodurilor pornind de la adresa primului nod	Referire prin deplasament al elementului fata de adresa de inceput a zonei

LISTA LINIARA

Implementare:

- **Utilizare variabile pointer;**
- **Nodurile contin doua categorii de informatii:**

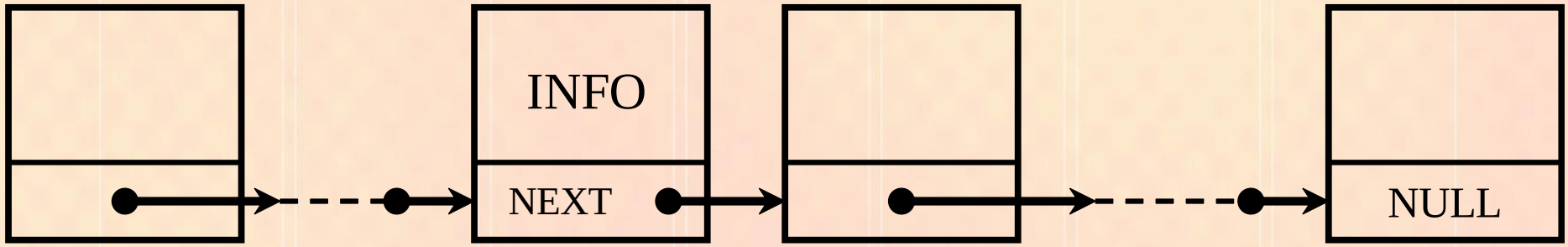
- **Campuri cu informatia structurala;**
- **Campuri cu informatia de legatura – adrese ale altor noduri din cadrul listei**

LISTA SIMPLA

Lista simpla:

- **Lista liniara;**
- **Un singur camp cu informatia de legatura – nodul succesor al campului curent;**
- **Ultimul nod nu are succesor – informatia de legatura este nula (NULL sau 0);**
- **Gestionata prin variabila pointer cu adresa primului nod din lista liniara.**

LISTA SIMPLA



Definirea structurii unui nod dintr-o lista simpla:

```
struct Nod{
        char inf;
        Nod *next;
};
```

LISTA SIMPLA

Declararea unei liste simple vide:

```
Nod *prim = NULL;
```

Alocarea de memorie heap pentru un nod al liste simple:

```
Nod *nou = new Nod;
```

Dezalocarea de memorie heap pentru un nod al liste simple:

```
delete nou;
```

LISTA SIMPLA

Initializarea si referirea informatiei utile dintr-un nod al listei simple:

```
nou->inf = 'A';
```

Initializarea si referirea informatiei de legatura dintr-un nod al listei simple:

```
nou->next = NULL;
```

LISTA SIMPLA

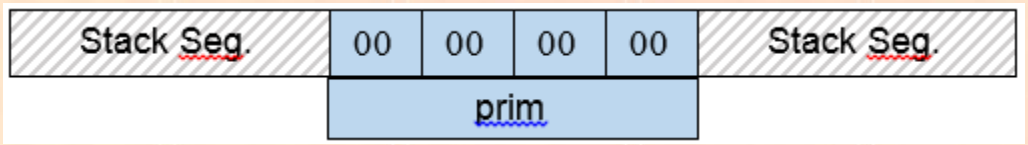
Operatii cu liste simple:

- **Creare listei;**
- **Inserare nod;**
- **Traversare;**
- **Cautare;**
- **Interschimbare noduri;**
- **Stergere nod;**
- **Stergerea listei;**
- **Sortarea listei;**
- **Concatenarea de liste;**
- **Interclasare liste.**

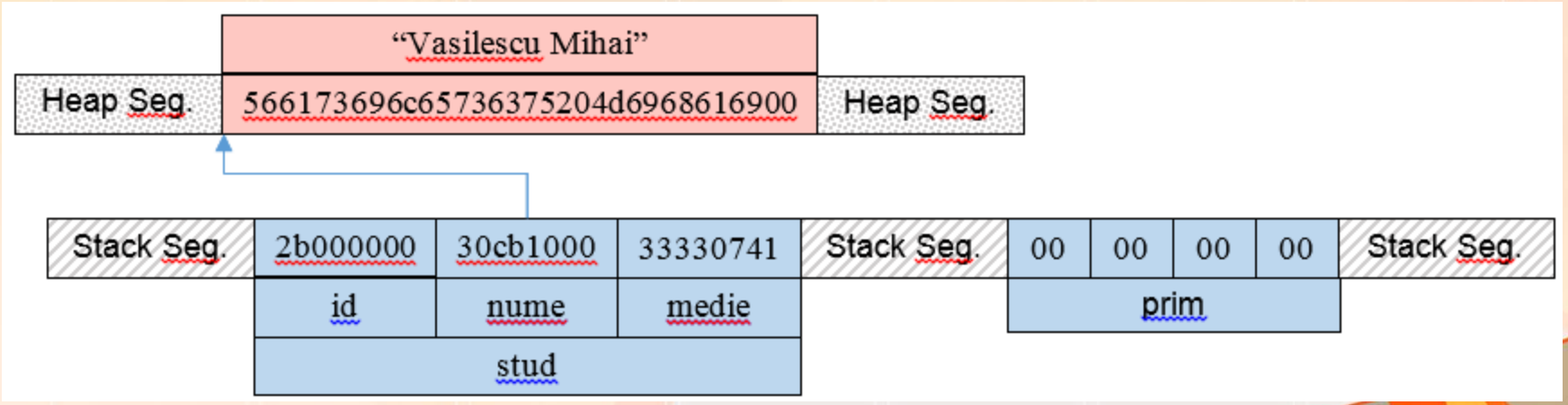
LISTA SIMPLA

Creare listei

```
Nod *prim = 0;
```

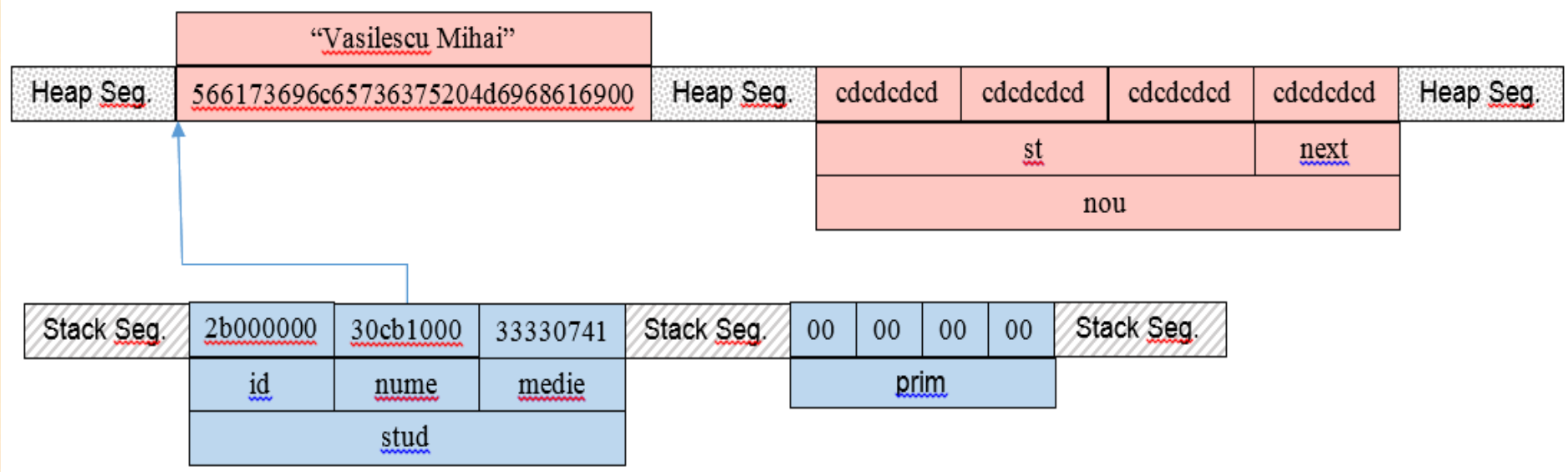



```
stud.id = atoi(token); // 43
stud.nume = (char*)malloc((strlen(token) + 1) * sizeof(char));
strcpy(stud.nume, token); // Vasilescu Mihai
stud.medie = atof(token); // 8.45
```



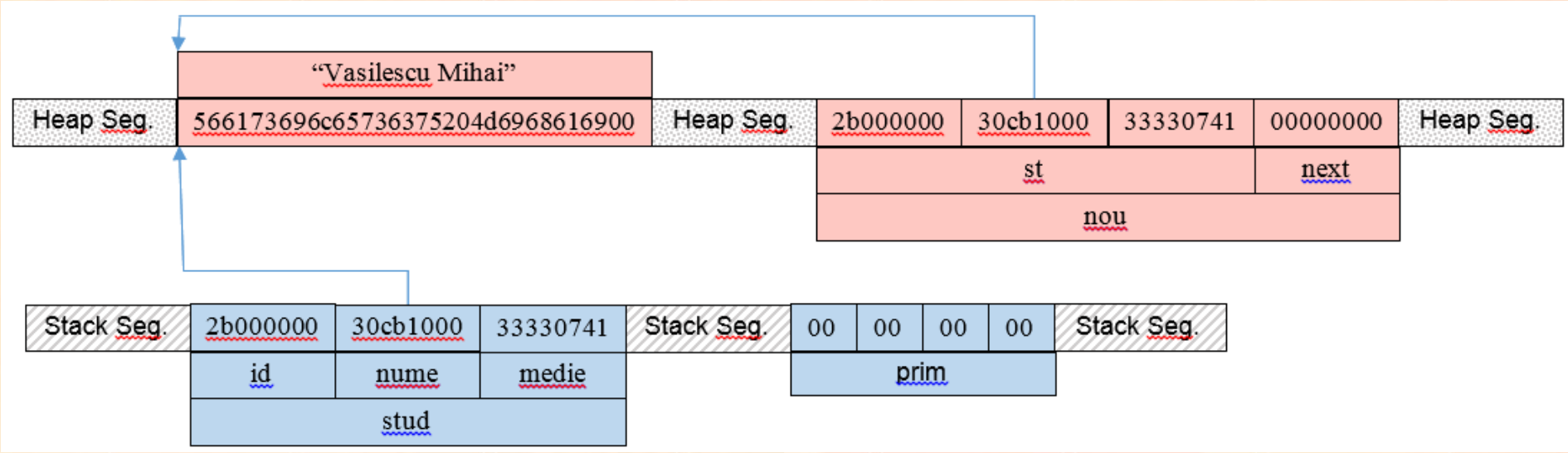
LISTA SIMPLA

```
Nod* nou;
nou = (Nod*)malloc(1 * sizeof(Nod));
```



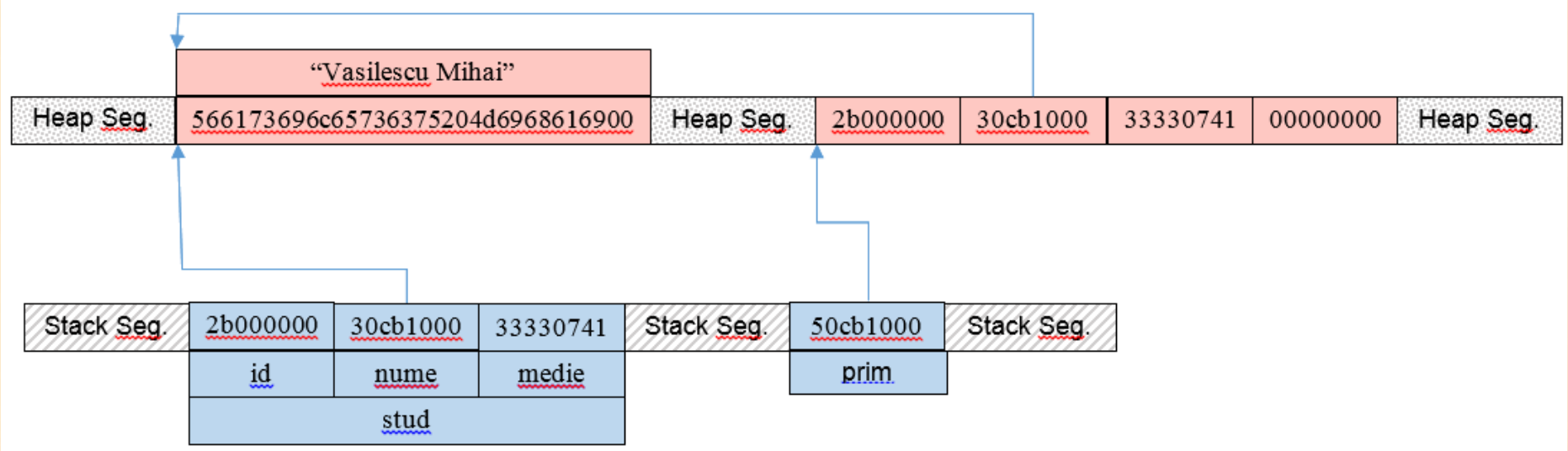
LISTA SIMPLA

```
nou->st = s;
nou->next = p;
```



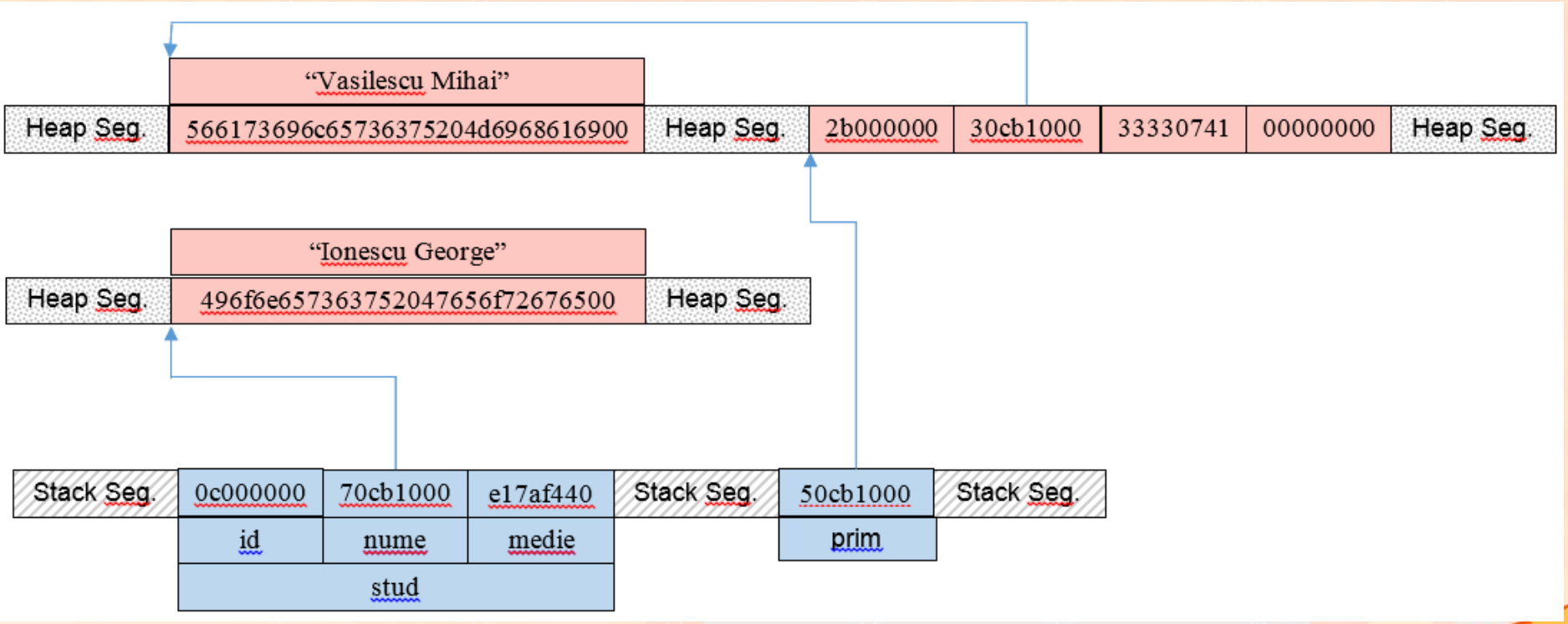
LISTA SIMPLA

```
prim = inserareLista(prim, stud);
```



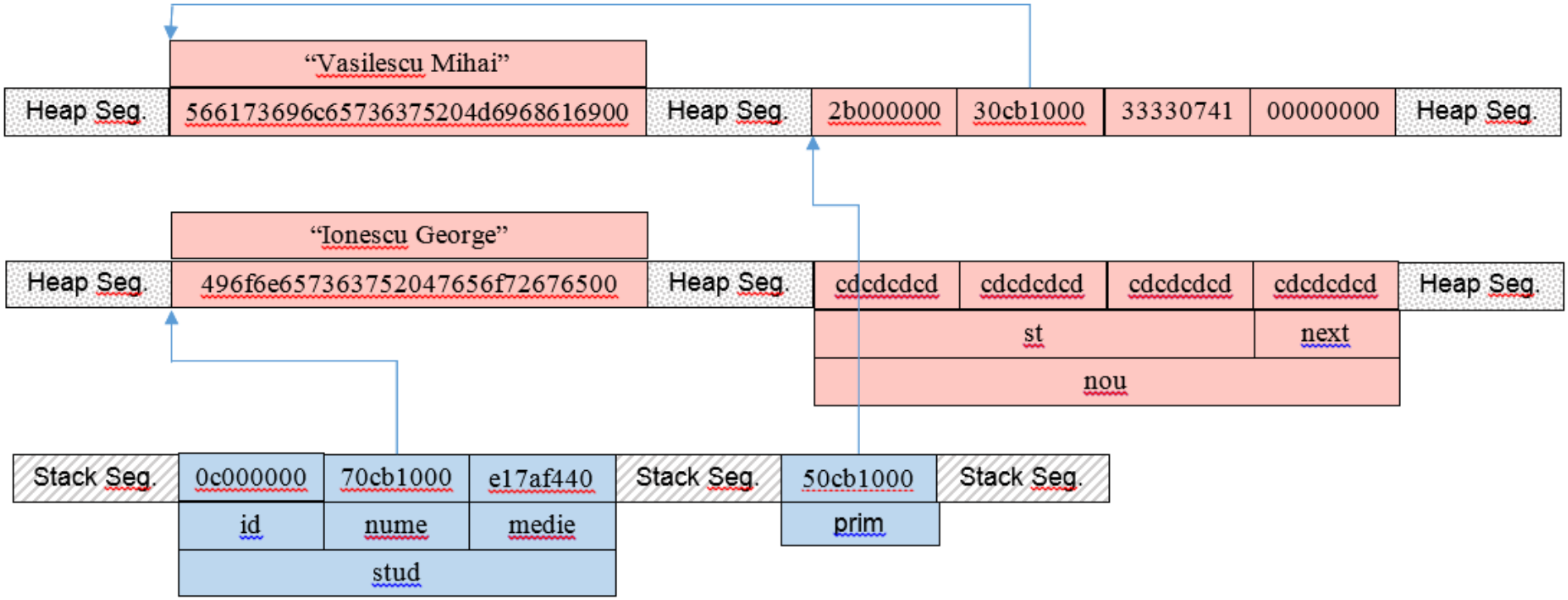
LISTA SIMPLA

```
stud.id = atoi(token); // 12
stud.nume = (char*)malloc((strlen(token) + 1) * sizeof(char));
strcpy(stud.nume, token); // Ionescu George
stud.medie = atof(token); // 7.64
```



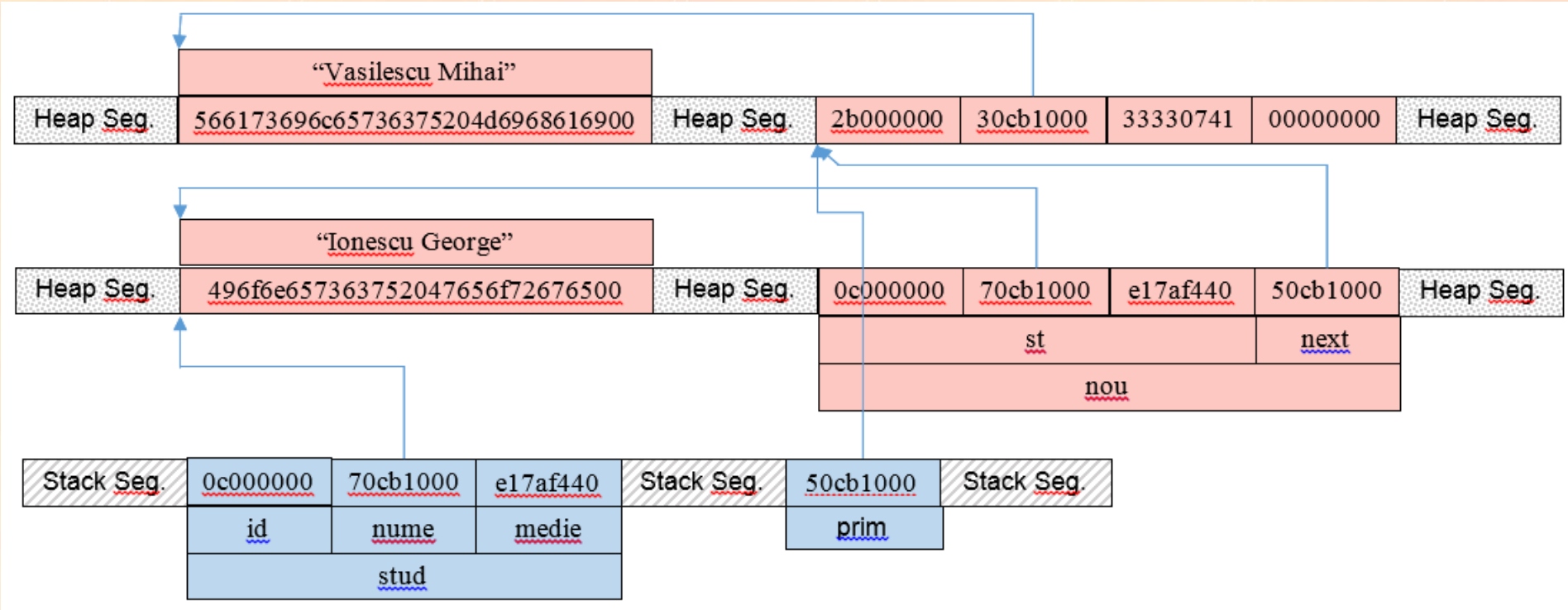
LISTA SIMPLA

```
Nod* nou;
nou = (Nod*)malloc(1 * sizeof(Nod));
```



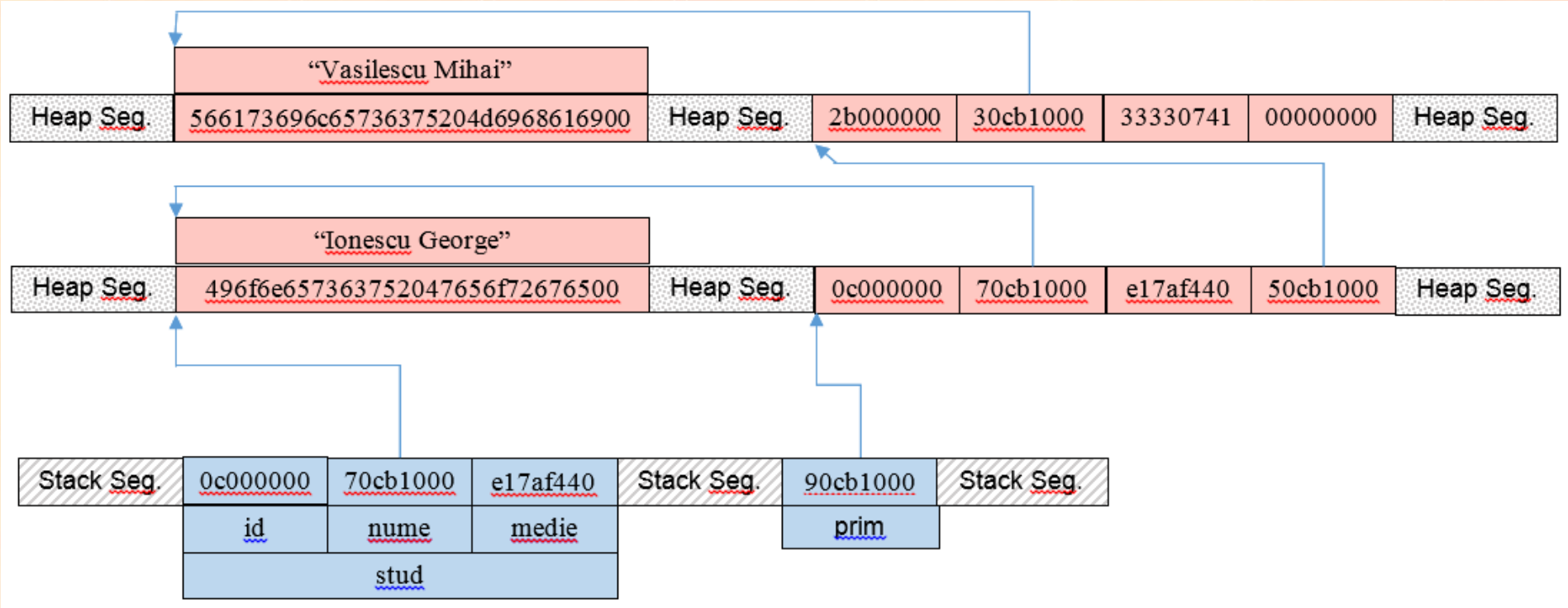
LISTA SIMPLA

```
nou->st = s;
nou->next = p;
```



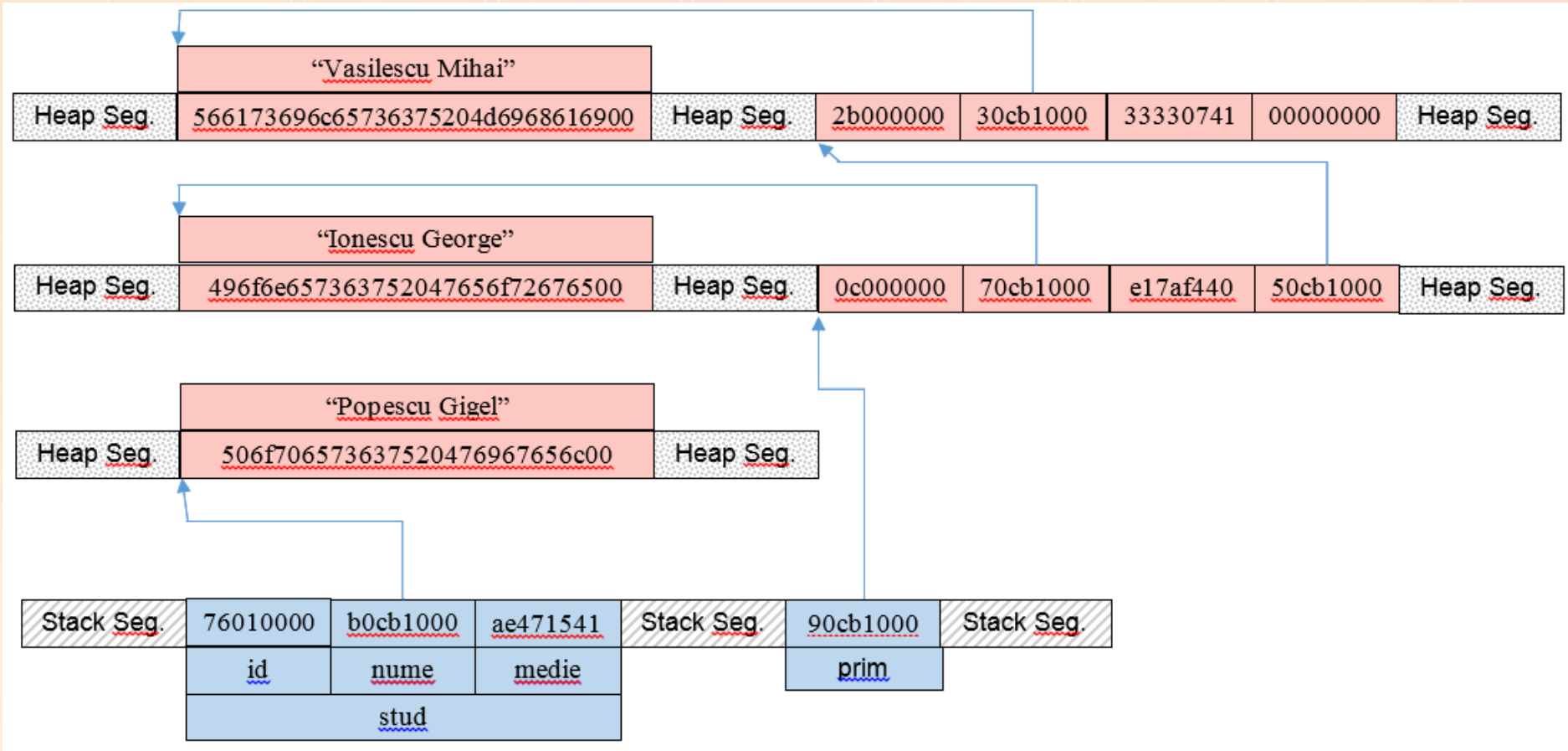
LISTA SIMPLA

```
prim = inserareLista(prim, stud);
```



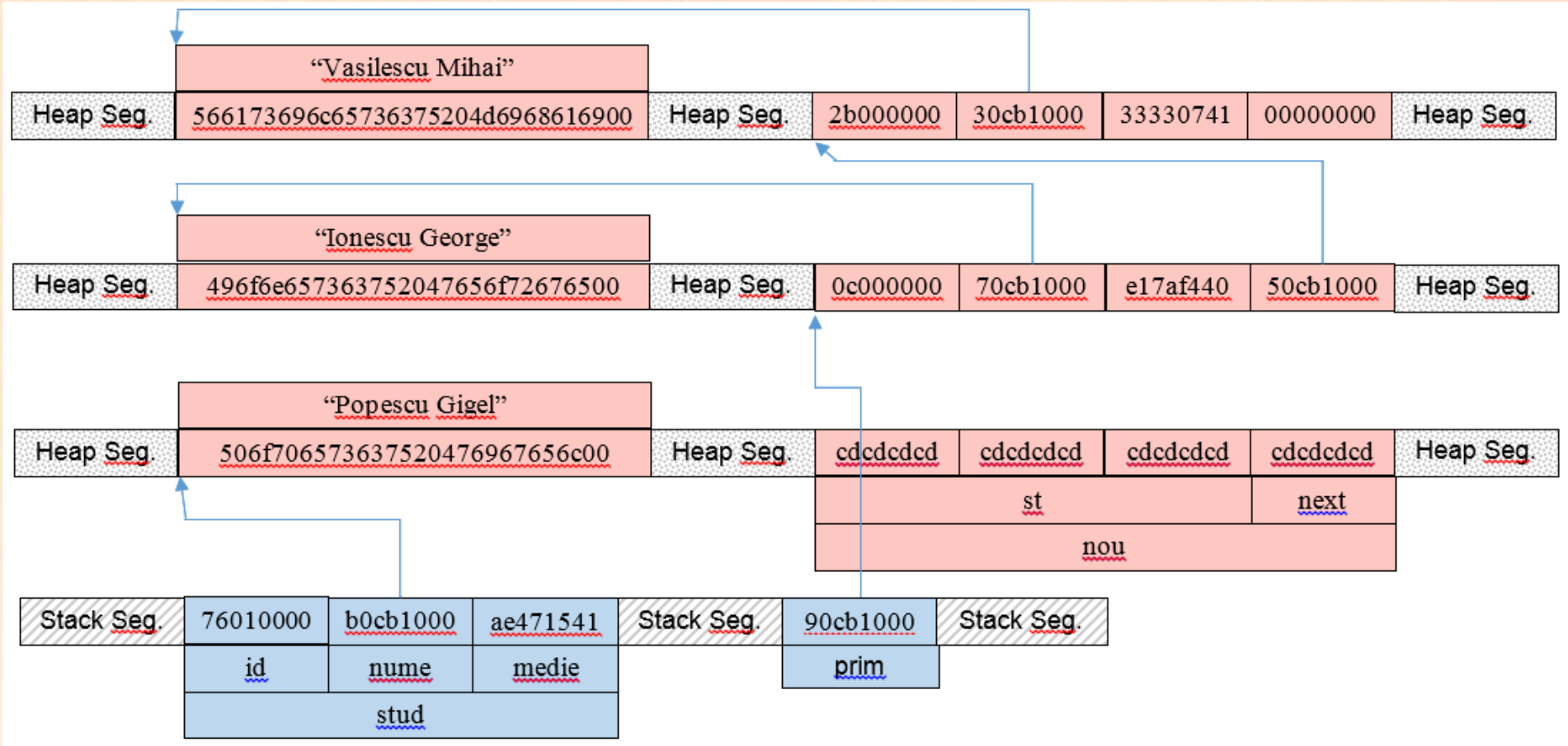
LISTA SIMPLA

```
stud.id = atoi(token); // 374
stud.nume = (char*)malloc((strlen(token) + 1) * sizeof(char));
strcpy(stud.nume, token); // Popescu Gigel
stud.medie = atof(token); // 9.33
```



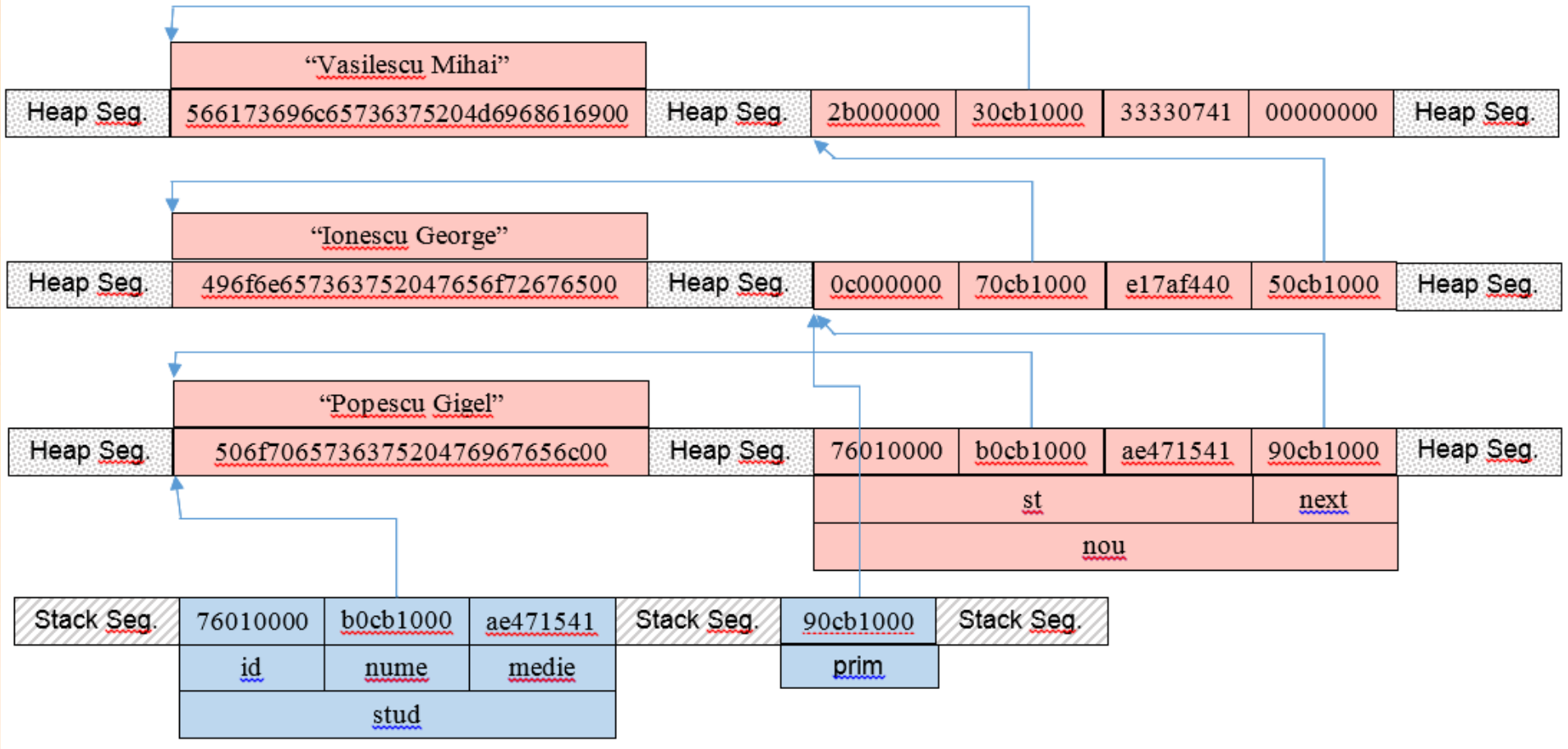
LISTA SIMPLA

```
Nod* nou;
nou = (Nod*)malloc(1 * sizeof(Nod));
```



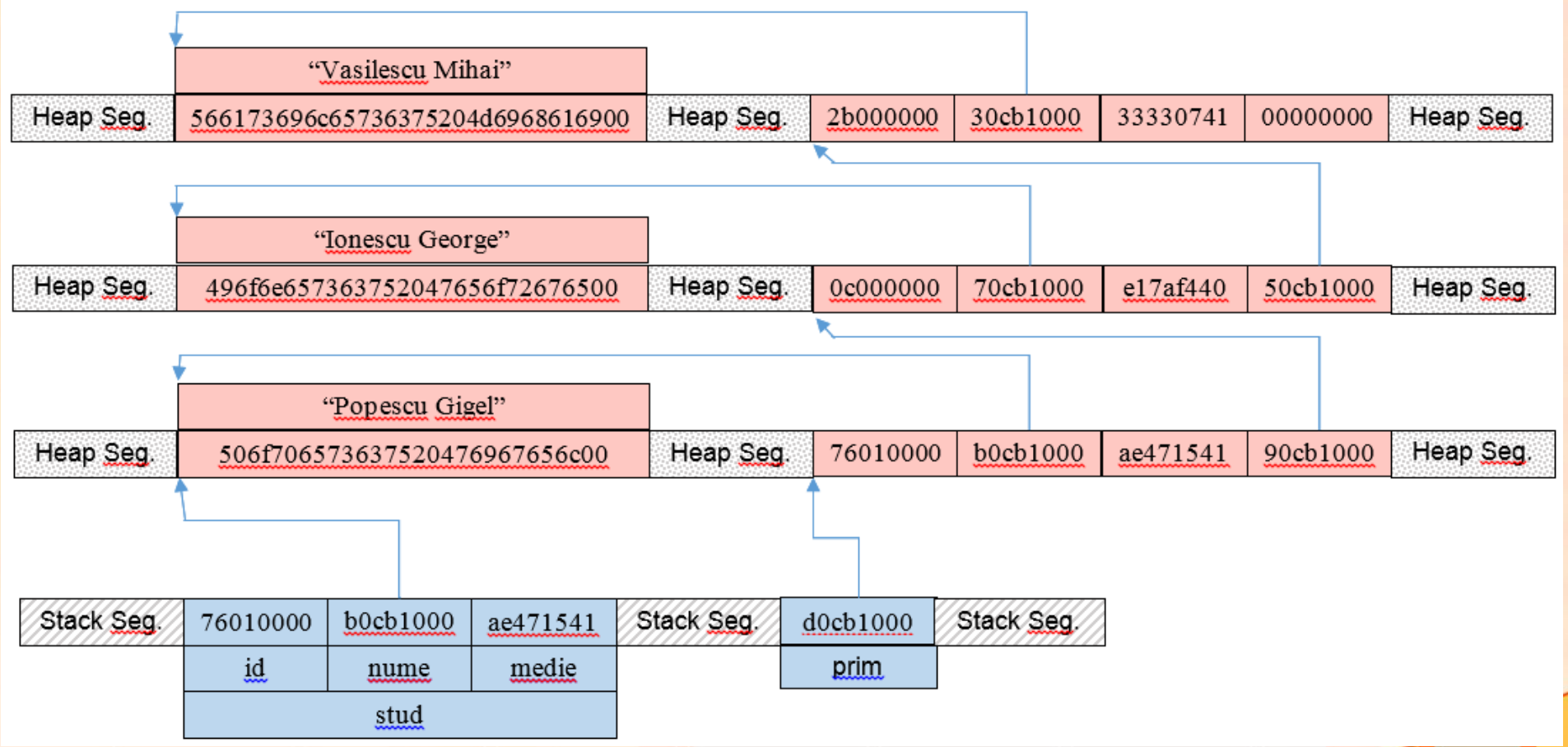
LISTA SIMPLA

```
nou->st = s;
nou->next = p;
```



LISTA SIMPLA

```
prim = inserareLista(prim, stud);
```



LISTA DUBLA

Lista dubla:

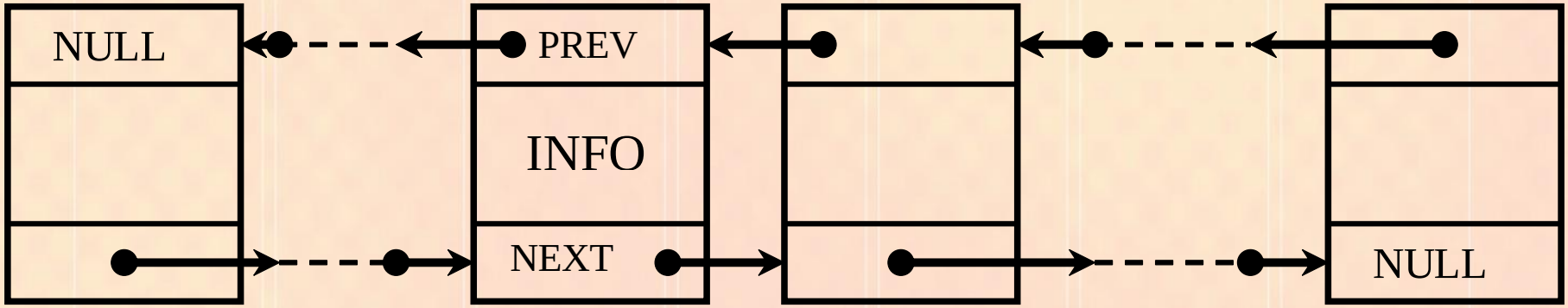
- **Lista liniara;**
- **Doua campuri cu informatii de legatura – nodurile succesor, respectiv predecesor ale campului curent;**
- **Ultimul nod nu are succesor – informatia de legatura este nula (NULL sau 0);**
- **Primul nod nu are predecesor - informatia de legatura este nula (NULL sau 0);**

LISTA DUBLA

Lista dubla (continuare):

- **Gestionata prin variabile pointer: una retine adresa primului nod din lista liniara, iar cealalta retine adresa ultimului nod din lista liniara;**
- **Posibilitatea gestionarii structurii printr-o structura articol ce incapsuleaza cei doi pointeri.**

LISTA DUBLA



Definirea structurii unui nod dintr-o lista dubla:

```
struct NodD{
        char inf;
        NodD *prev, *next;
};
```

LISTA DUBLA

Incapsularea adreselor primului si ultimului nod ale unei liste duble vide:

```
struct ListaD{
        NodD *prim, *ultim;
};
```

Declararea unei liste duble vide:

```
ListaD lst;
lst.prim = NULL;
lst.ultim = NULL;
```

Alocarea de memorie heap pentru un nod al liste duble:

```
NodD *Nou = new NodD;
```

LISTA DUBLA

Dezalocarea de memorie heap pentru un nod al liste duble:

```
delete Nou;
```

Initializarea si referirea informatiei utile dintr-un nod al listei duble:

```
Nou->inf = 'Z';
```

Initializarea si referirea informatiilor de legatura dintr-un nod al listei duble:

```
Nou->next = NULL;
Nou->prev = NULL;
```

LISTA DUBLA

Referirea informatiilor din primul nod al listei duble:

```
lst.prim->inf;
lst.prim->next;
lst.prim->prev;
```

Referirea informatiilor din ultimul nod al listei duble:

```
lst.ultim->inf;
lst.ultim->next;
lst.ultim->prev;
```

LISTA DUBLA

Operatii cu liste duble – similare operatiilor cu liste simple