Семинар №11

ФАКТ 2020

Бирюков В. А.

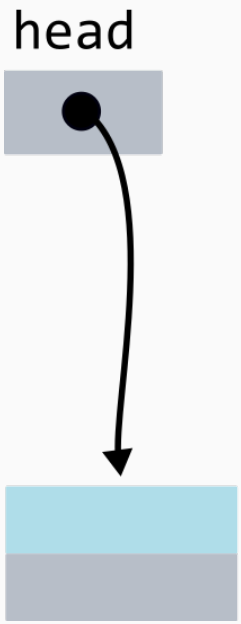
November 15, 2020

Связный список

Связный список

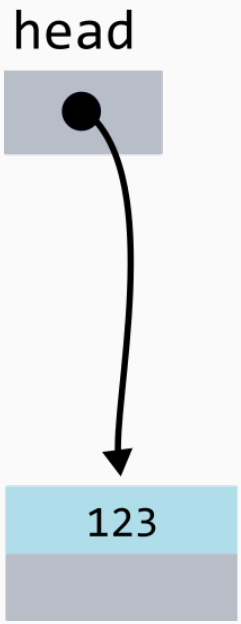

```
struct node
{
    int value;
    struct node* next;
};
typedef struct node Node;
```

Связный список



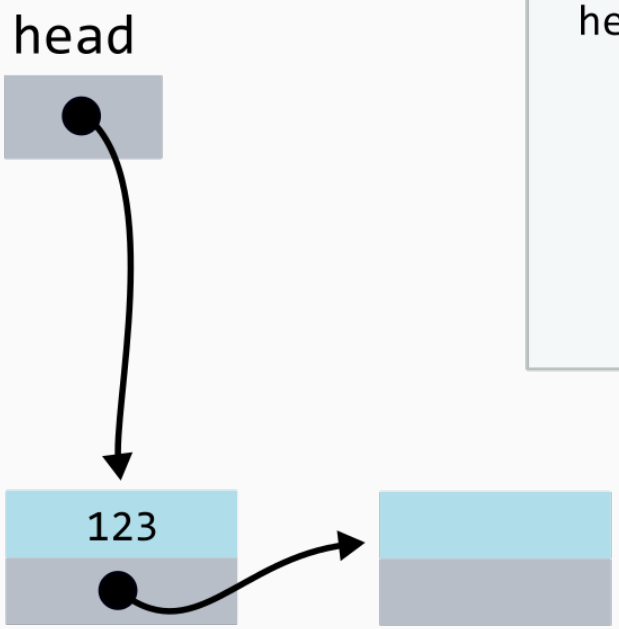

```
Node* head = malloc(sizeof(Node));
```

Связный список



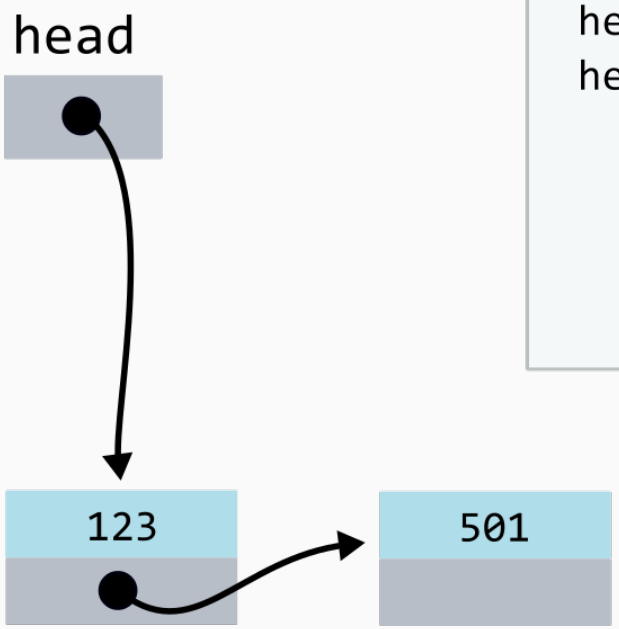

```
Node* head = malloc(sizeof(Node));
head->value = 123;
```

Связный список



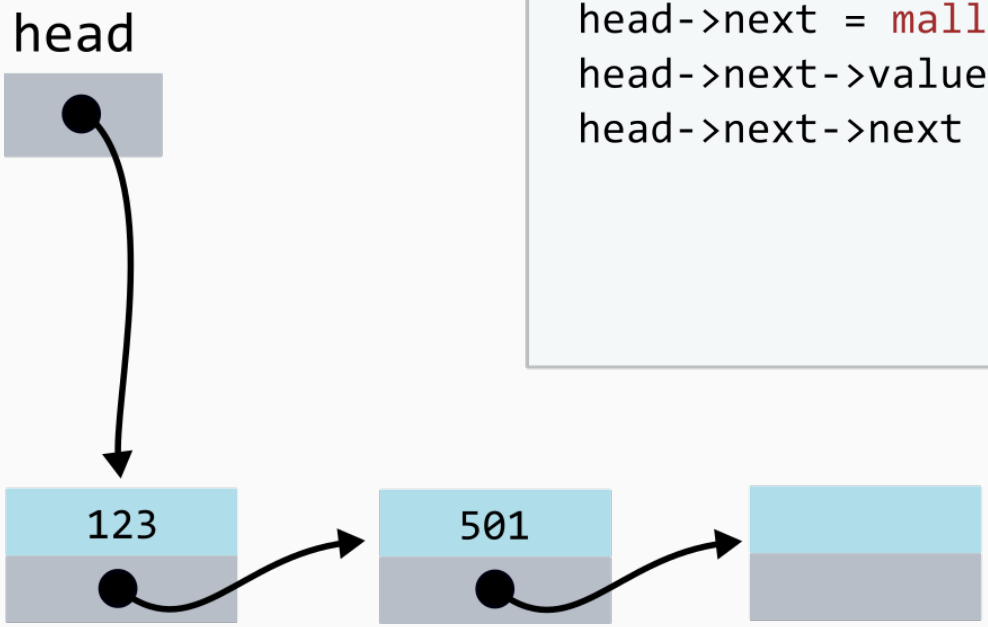

```
Node* head = malloc(sizeof(Node));
head->value = 123;
head->next = malloc(sizeof(Node));
```

Связный список



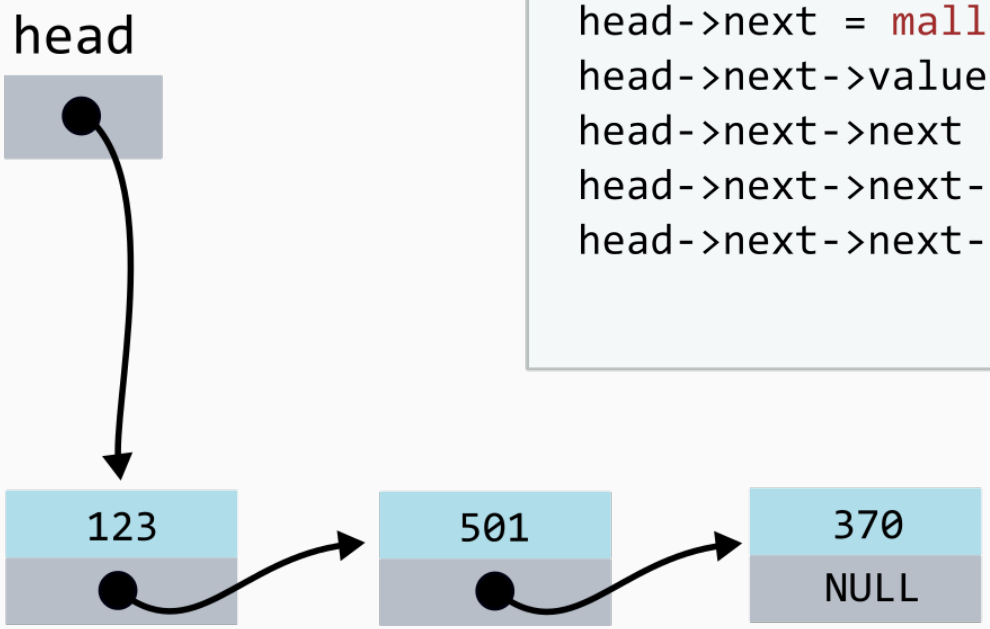

```
Node* head = malloc(sizeof(Node));
head->value = 123;
head->next = malloc(sizeof(Node));
head->next->value = 501;
```

Связный список




```
Node* head = malloc(sizeof(Node));
head->value = 123;
head->next = malloc(sizeof(Node));
head->next->value = 501;
head->next->next = malloc(sizeof(Node));
```

Связный список

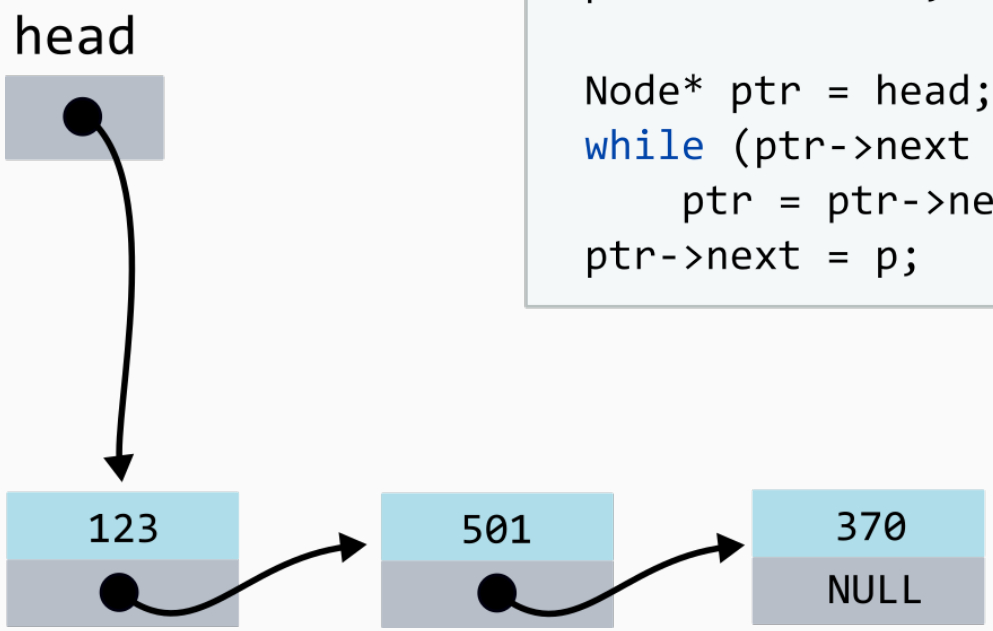



```
Node* head = malloc(sizeof(Node));
head->value = 123;
head->next = malloc(sizeof(Node));
head->next->value = 501;
head->next->next = malloc(sizeof(Node));
head->next->next->value = 370;
head->next->next->next = NULL;
```

Добавление элемента в конец связного списка

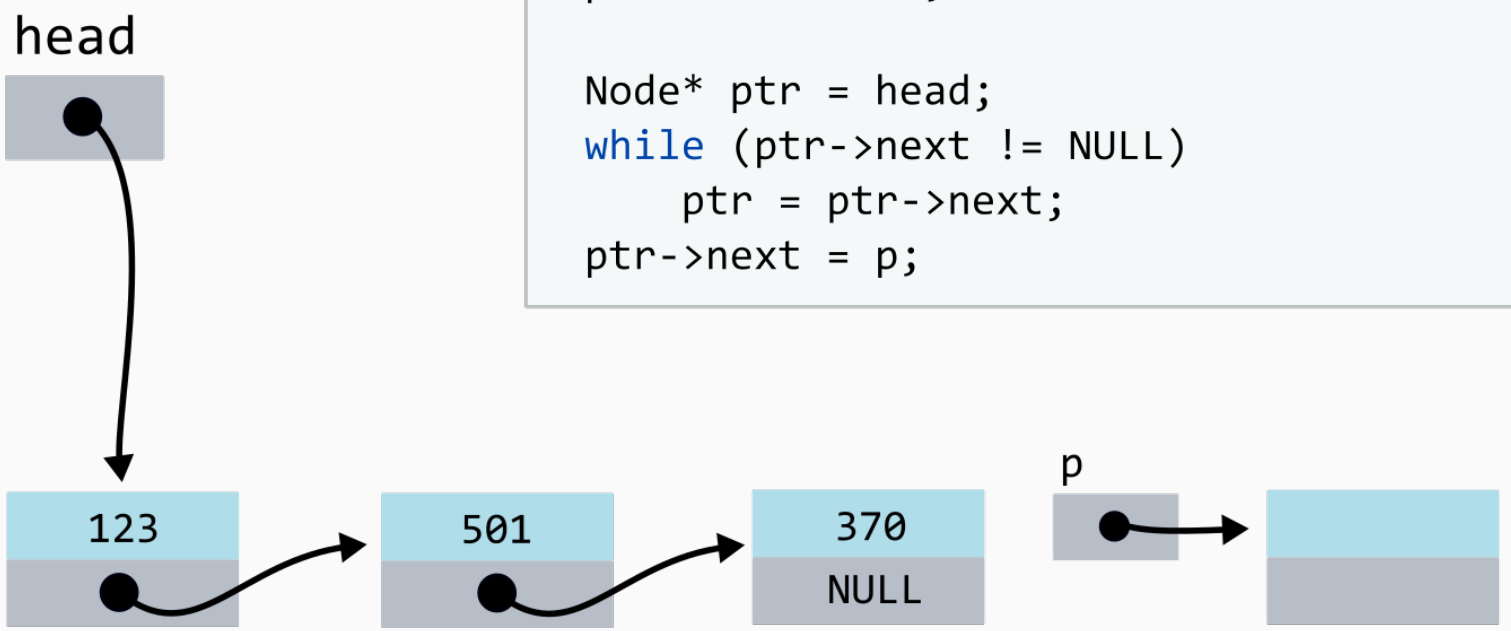
Добавление элемента в конец связного списка

```
Node* p = malloc(sizeof(Node));
p->value = 100;
p->next = NULL;

Node* ptr = head;
while (ptr->next != NULL)
    ptr = ptr->next;
ptr->next = p;
```



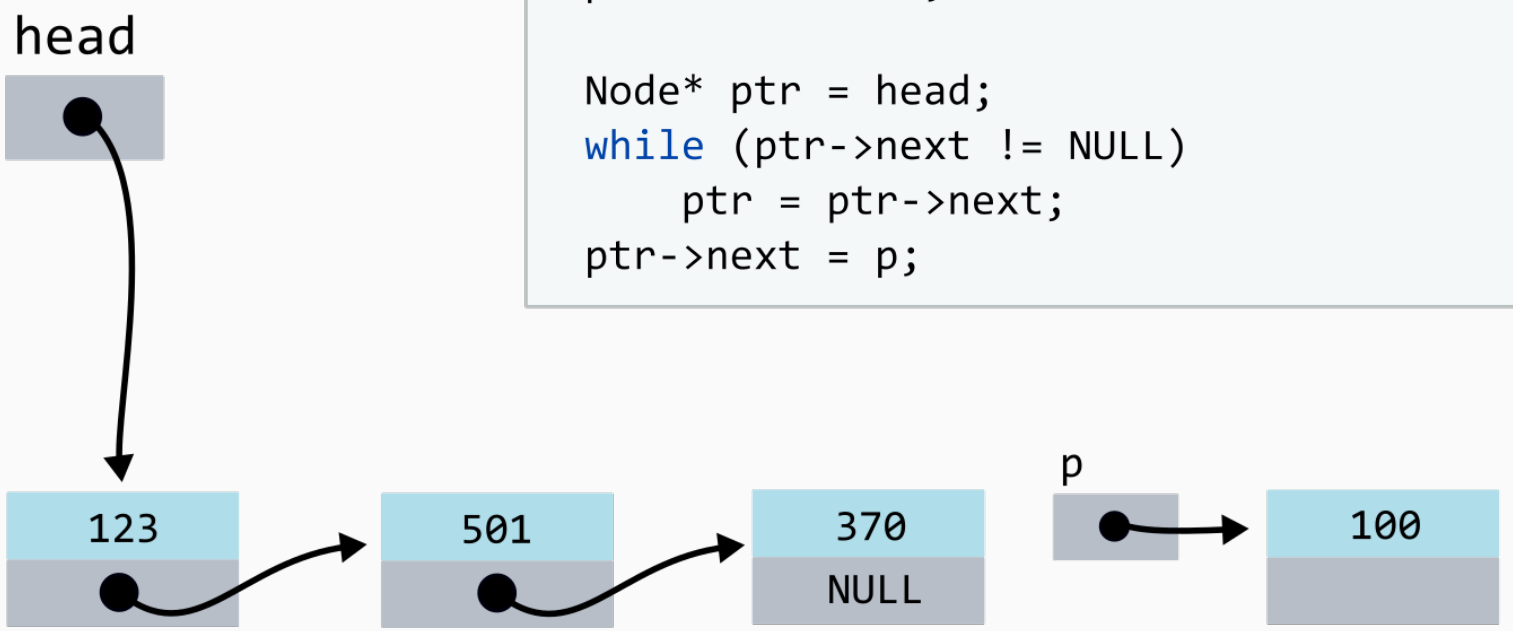
Добавление элемента в конец связного списка

```
Node* p = malloc(sizeof(Node));
p->value = 100;
p->next = NULL;

Node* ptr = head;
while (ptr->next != NULL)
    ptr = ptr->next;
ptr->next = p;
```



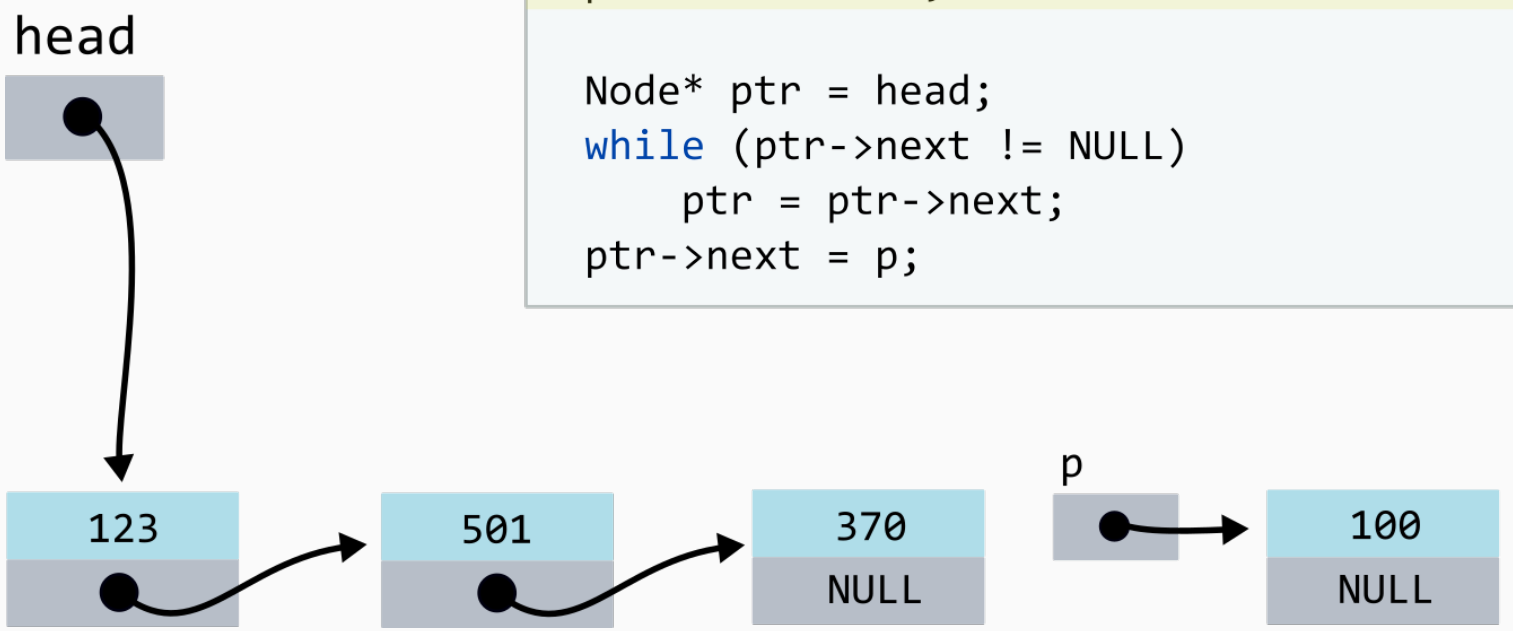
Связный список

```
Node* p = malloc(sizeof(Node));
p->value = 100;
p->next = NULL;

Node* ptr = head;
while (ptr->next != NULL)
    ptr = ptr->next;
ptr->next = p;
```



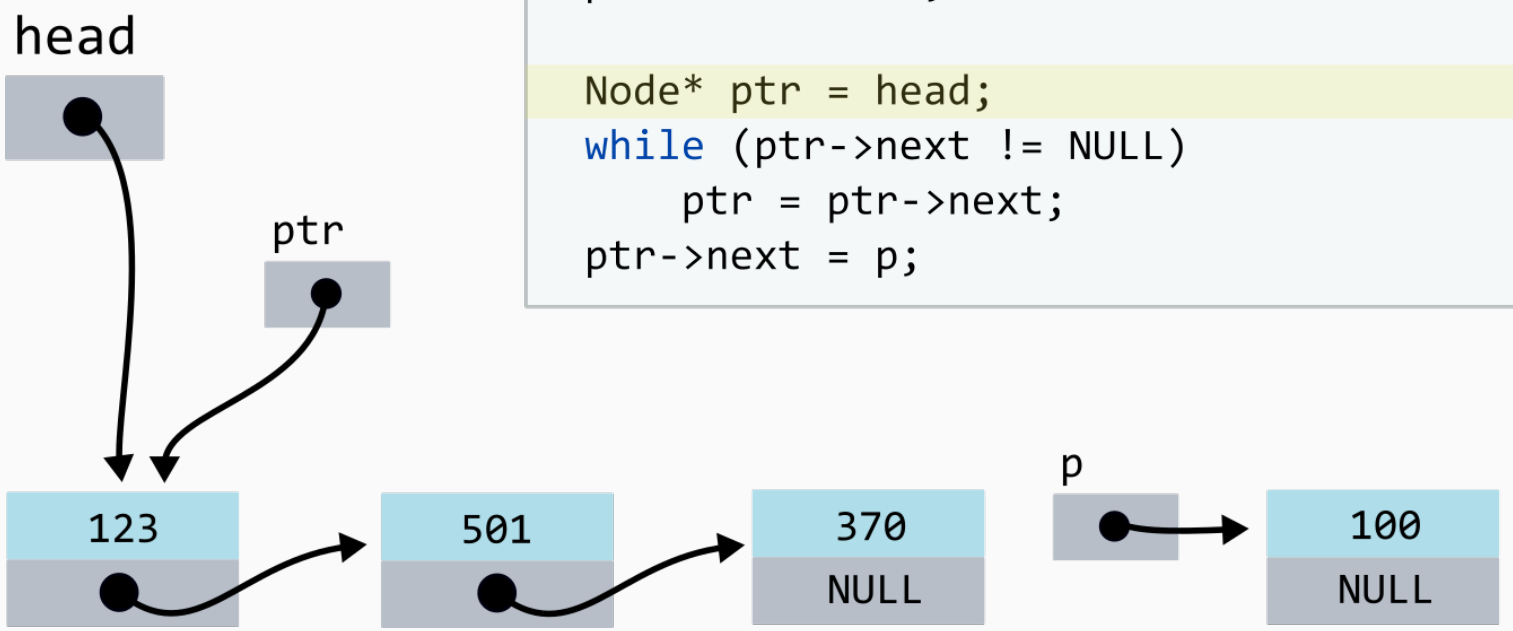
Связный список

```
Node* p = malloc(sizeof(Node));
p->value = 100;
p->next = NULL;

Node* ptr = head;
while (ptr->next != NULL)
    ptr = ptr->next;
ptr->next = p;
```



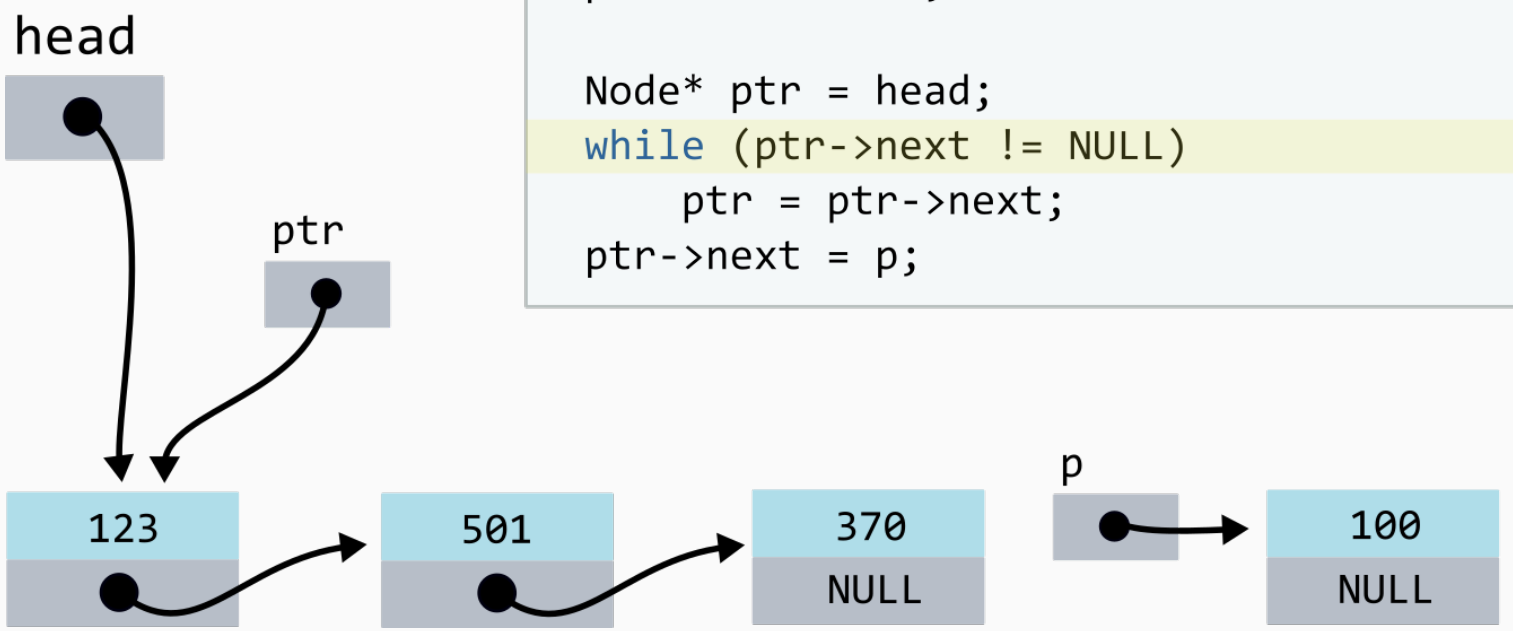
Связный список

```
Node* p = malloc(sizeof(Node));
p->value = 100;
p->next = NULL;

Node* ptr = head;
while (ptr->next != NULL)
    ptr = ptr->next;
ptr->next = p;
```



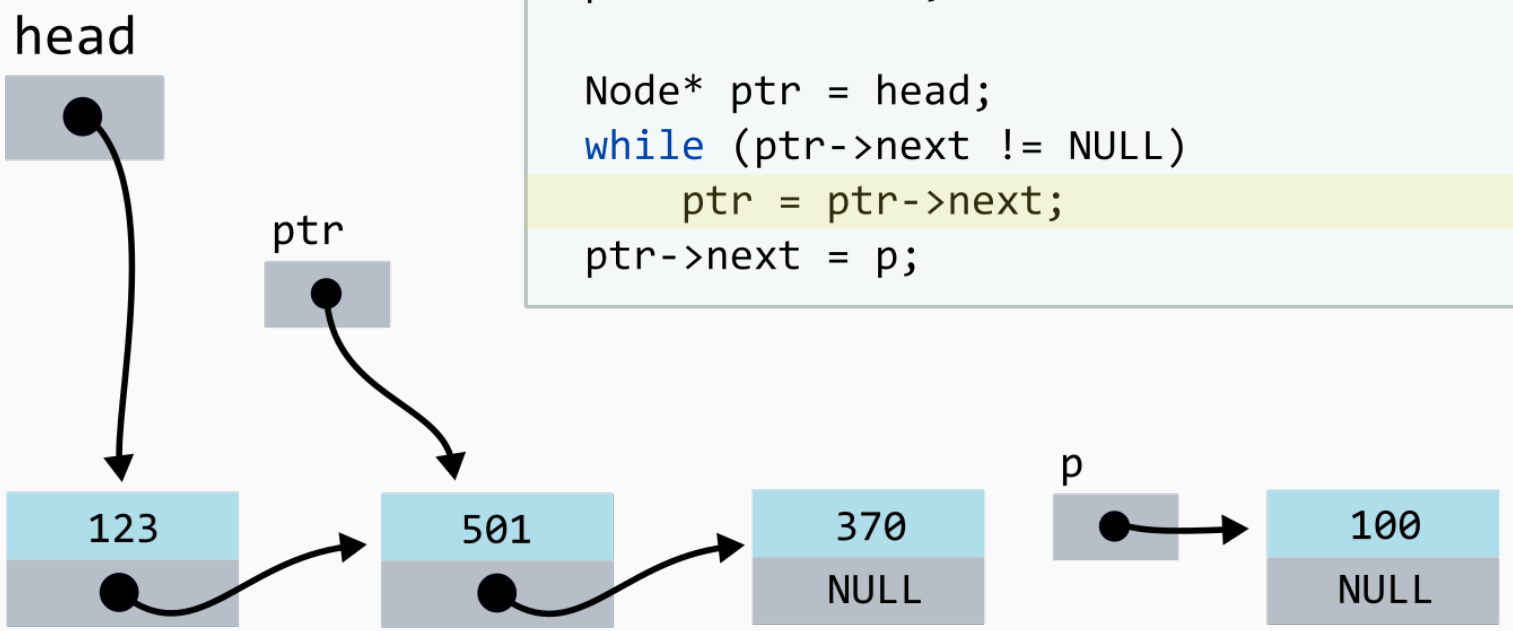
Связный список

```
Node* p = malloc(sizeof(Node));
p->value = 100;
p->next = NULL;

Node* ptr = head;
while (ptr->next != NULL)
    ptr = ptr->next;
ptr->next = p;
```



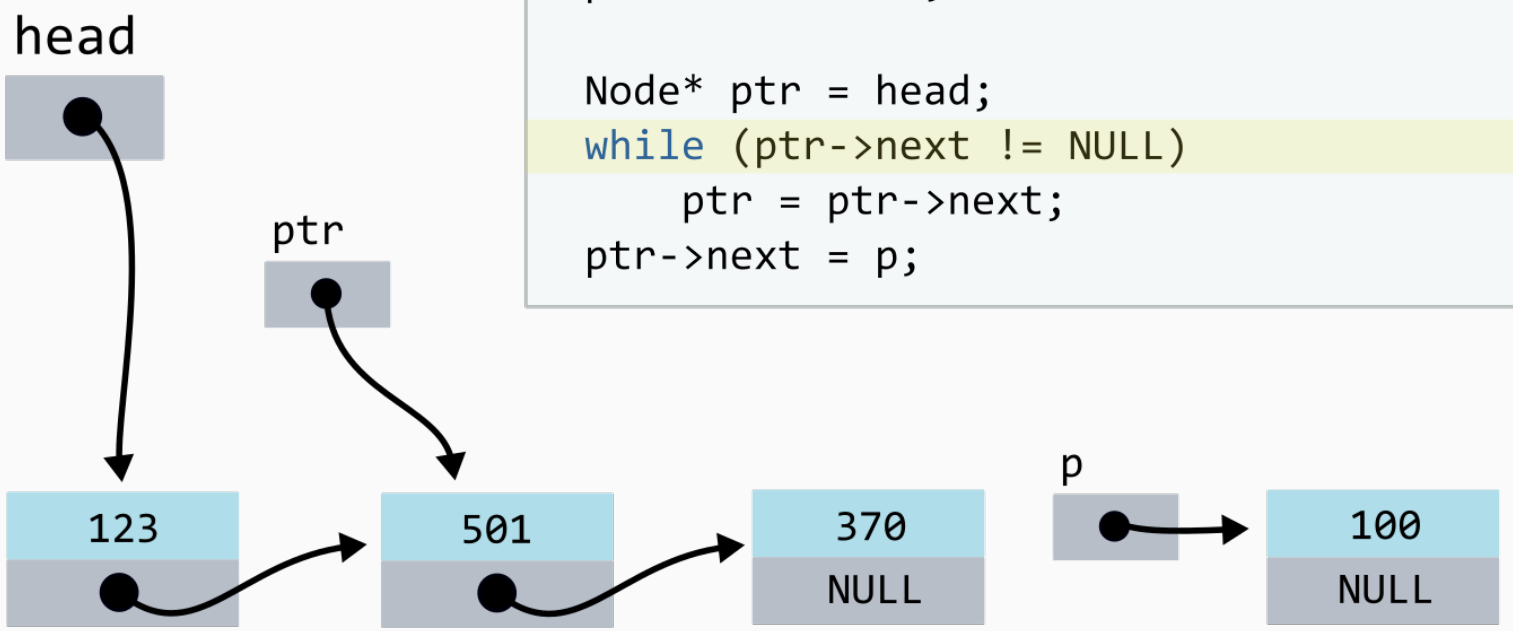
Связный список

```
Node* p = malloc(sizeof(Node));
p->value = 100;
p->next = NULL;

Node* ptr = head;
while (ptr->next != NULL)
    ptr = ptr->next;
ptr->next = p;
```



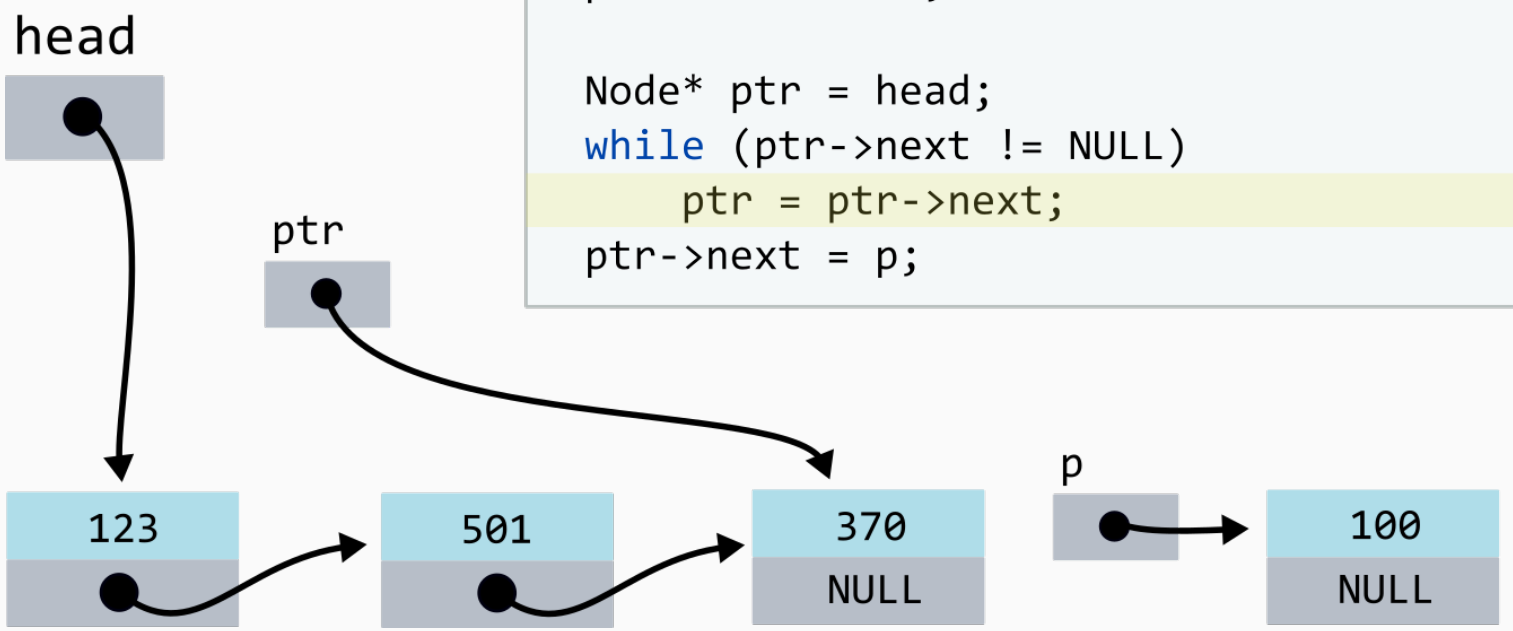
Связный список

```
Node* p = malloc(sizeof(Node));
p->value = 100;
p->next = NULL;

Node* ptr = head;
while (ptr->next != NULL)
    ptr = ptr->next;
ptr->next = p;
```



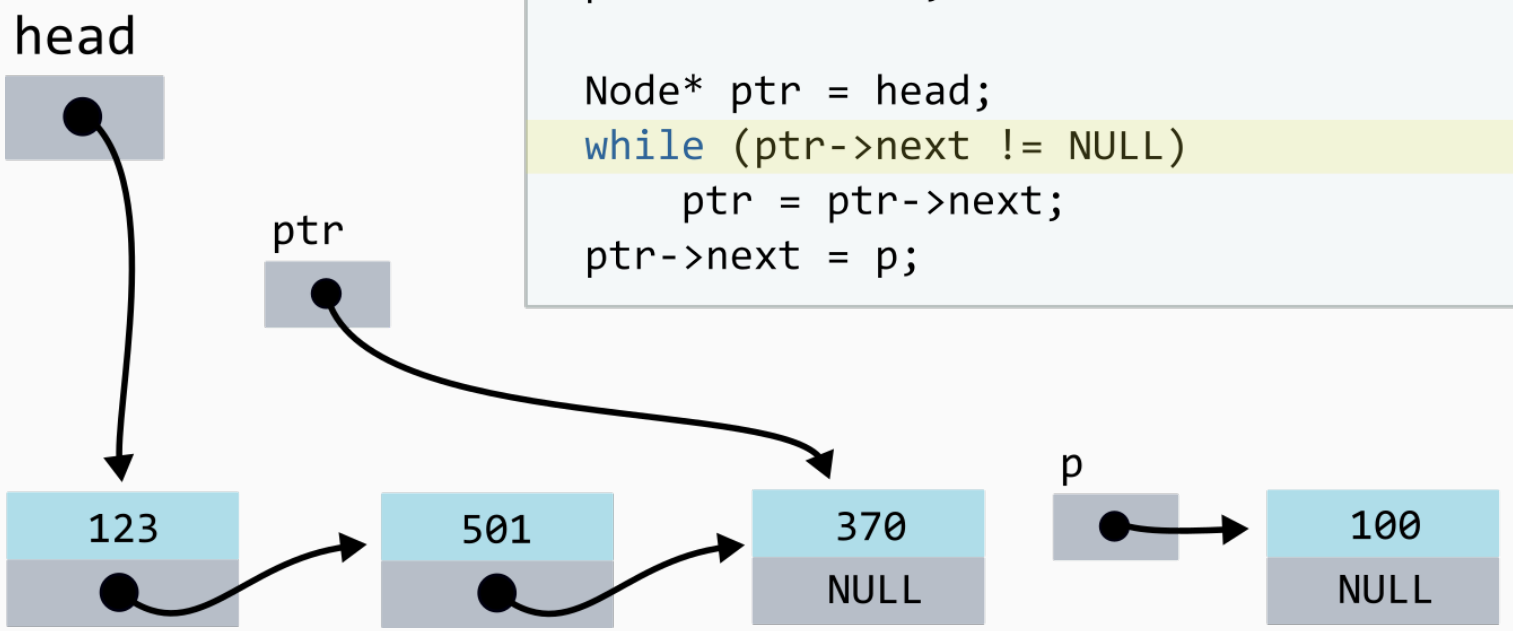
Связный список

```
Node* p = malloc(sizeof(Node));
p->value = 100;
p->next = NULL;

Node* ptr = head;
while (ptr->next != NULL)
    ptr = ptr->next;
ptr->next = p;
```



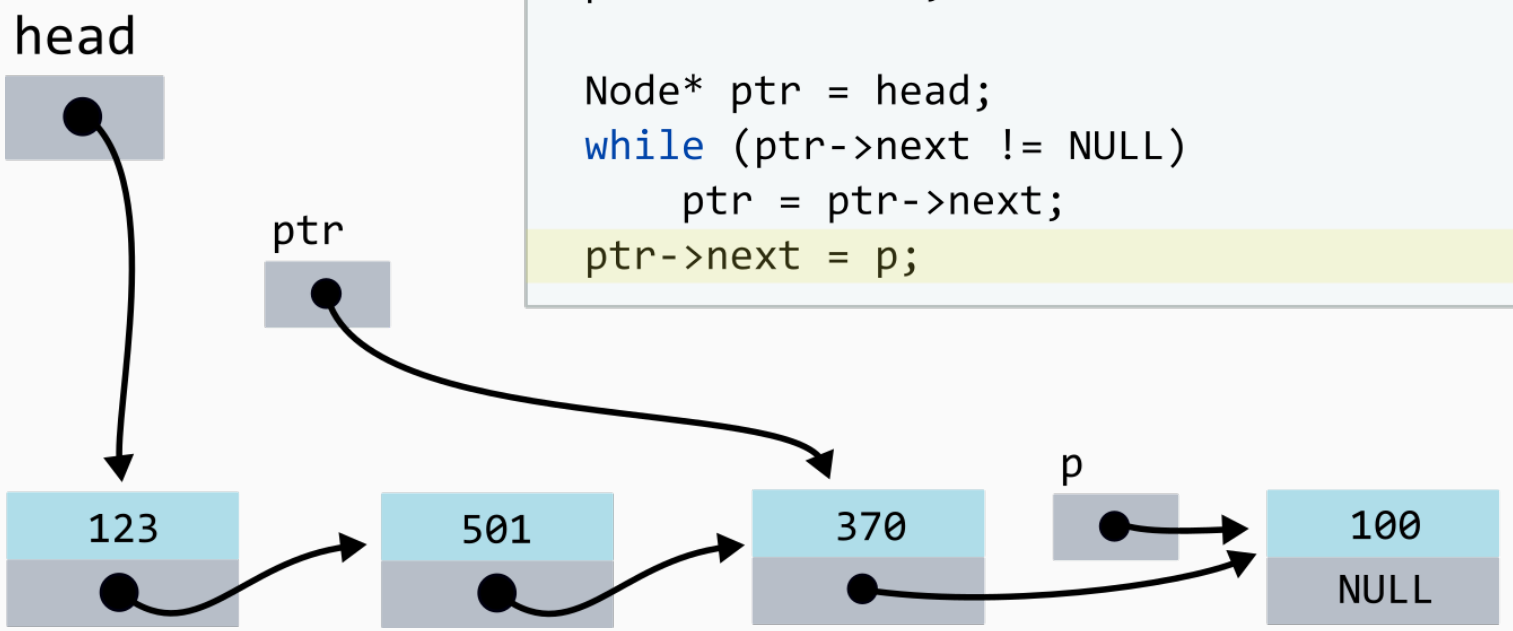
Связный список

```
Node* p = malloc(sizeof(Node));
p->value = 100;
p->next = NULL;

Node* ptr = head;
while (ptr->next != NULL)
    ptr = ptr->next;
ptr->next = p;
```



Связный список

```
Node* p = malloc(sizeof(Node));
p->value = 100;
p->next = NULL;

Node* ptr = head;
while (ptr->next != NULL)
    ptr = ptr->next;
ptr->next = p;
```



Связный список

```
Node* p = malloc(sizeof(Node));
p->value = 100;
p->next = NULL;

Node* ptr = head;
while (ptr->next != NULL)
    ptr = ptr->next;
ptr->next = p;
```