Семинар №11

ФАКТ 2020

Бирюков В. А.

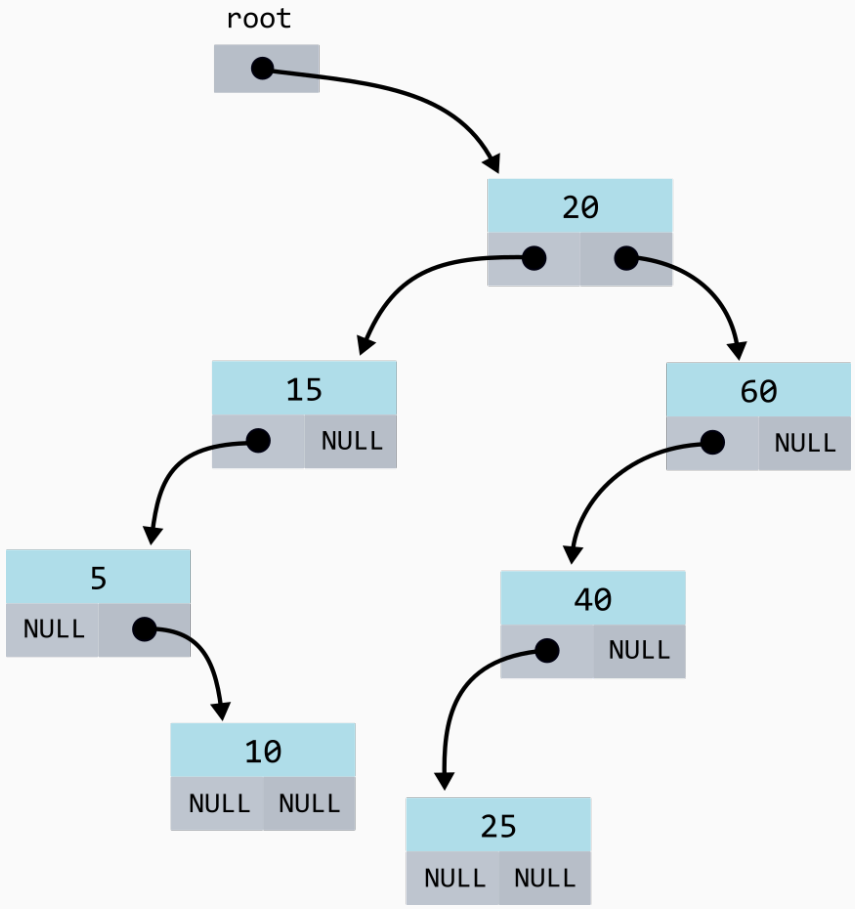
November 23, 2020

Бинарное дерево поиска

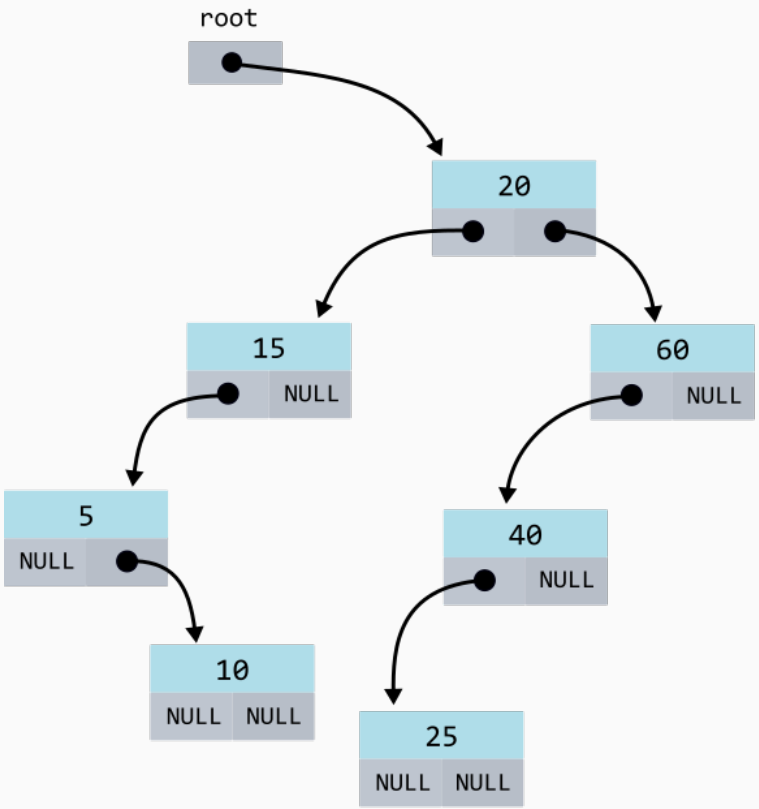
Добавление элемента в бинарное дерево поиска

Добавление элемента в бинарное дерево поиска

Попробуем добавить элемент 50 в следующее дерево поиска:



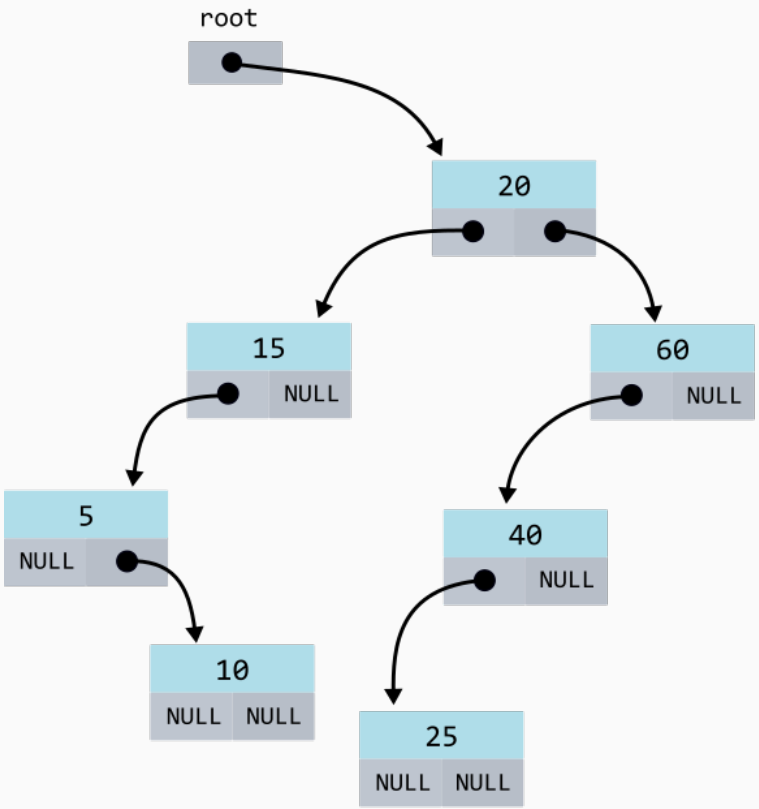
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

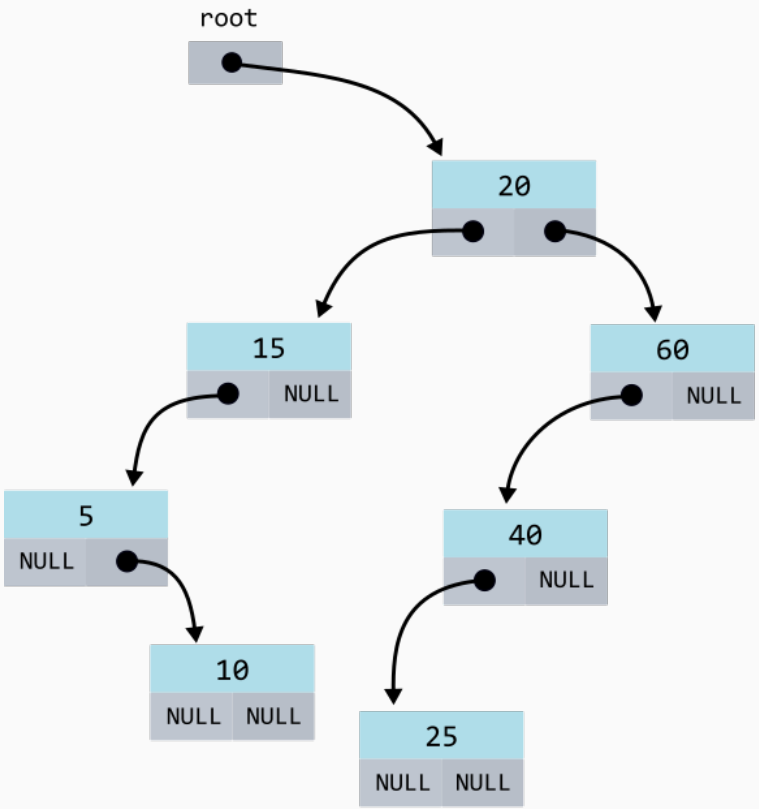
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

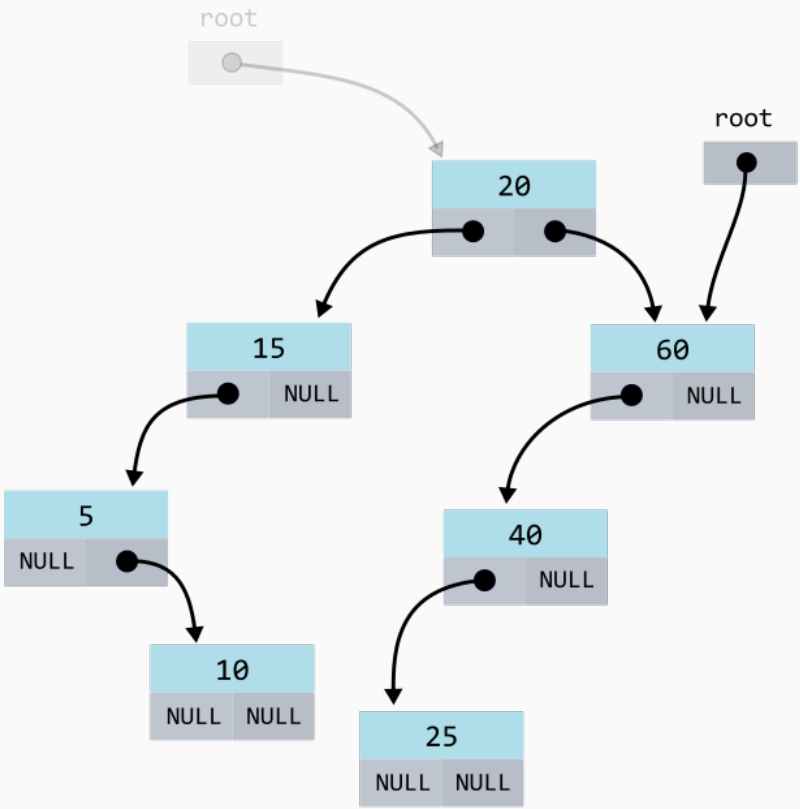
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

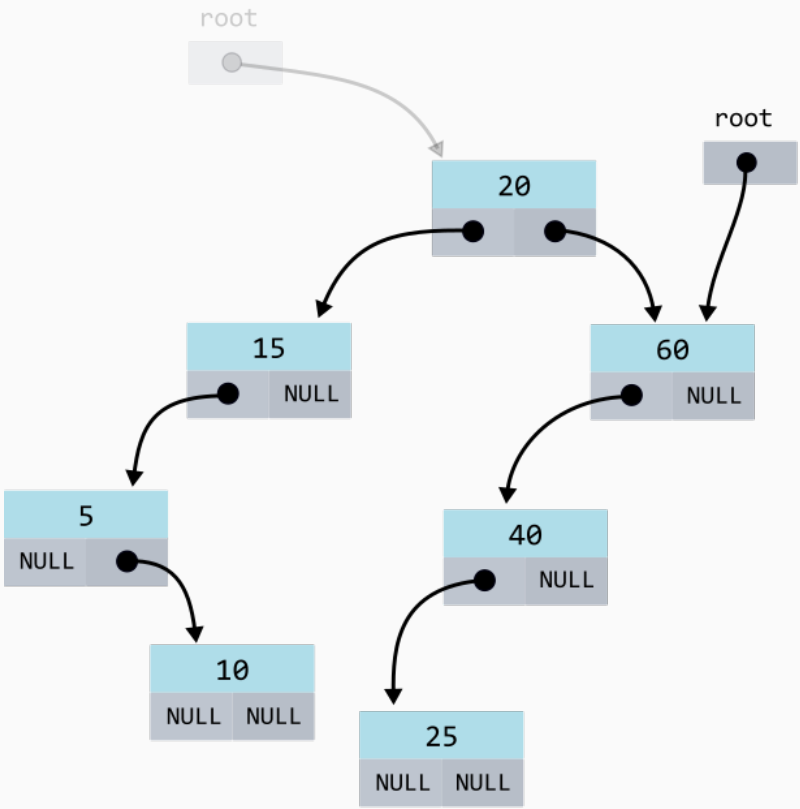
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

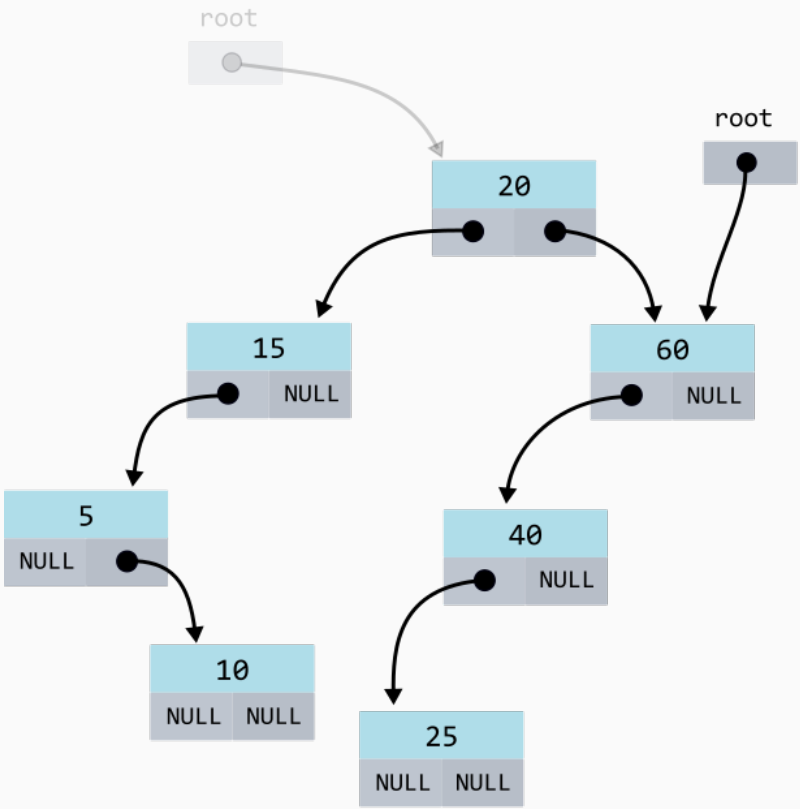
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

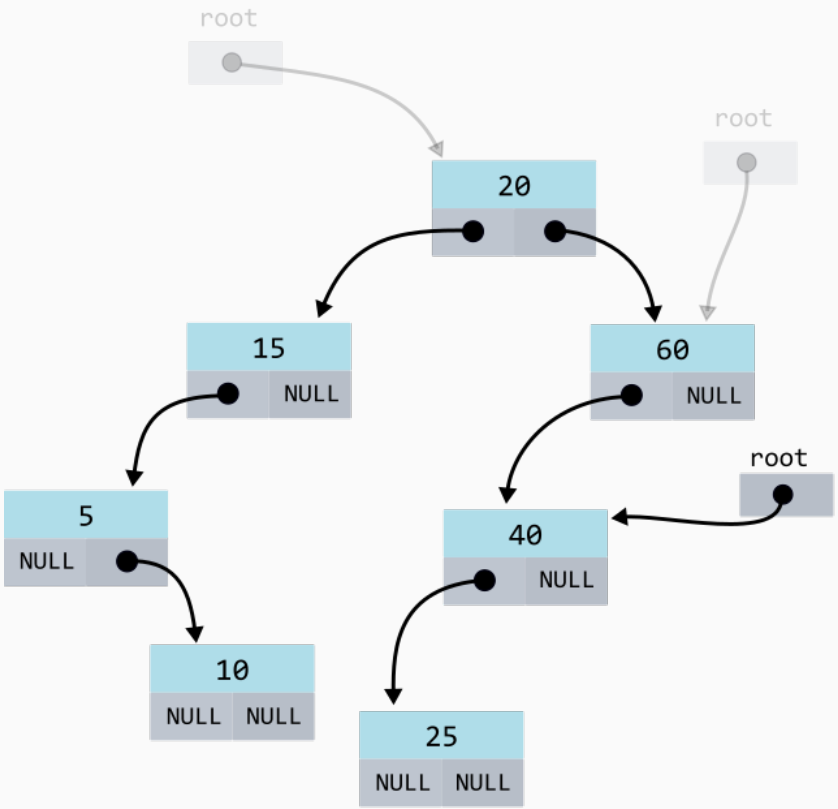
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

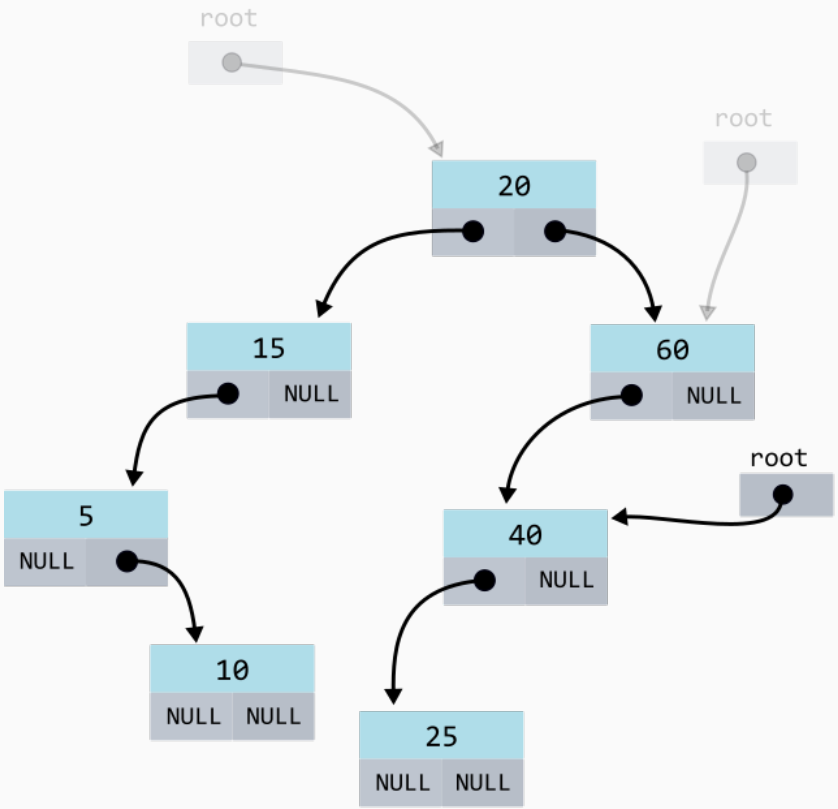
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
      bst_insert(root->right->left, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

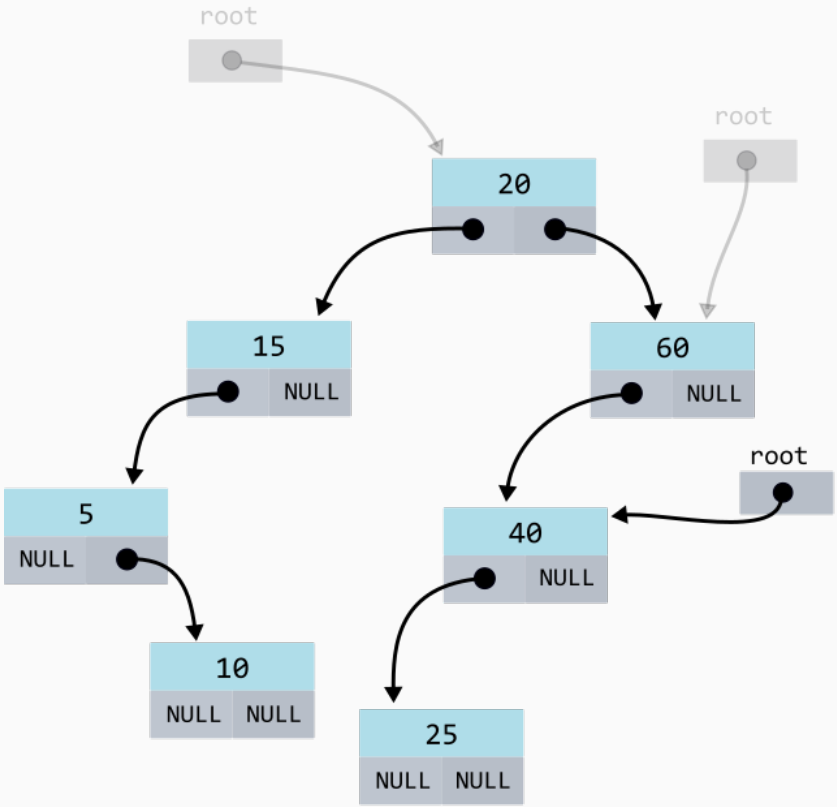
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
      bst_insert(root->right->left, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

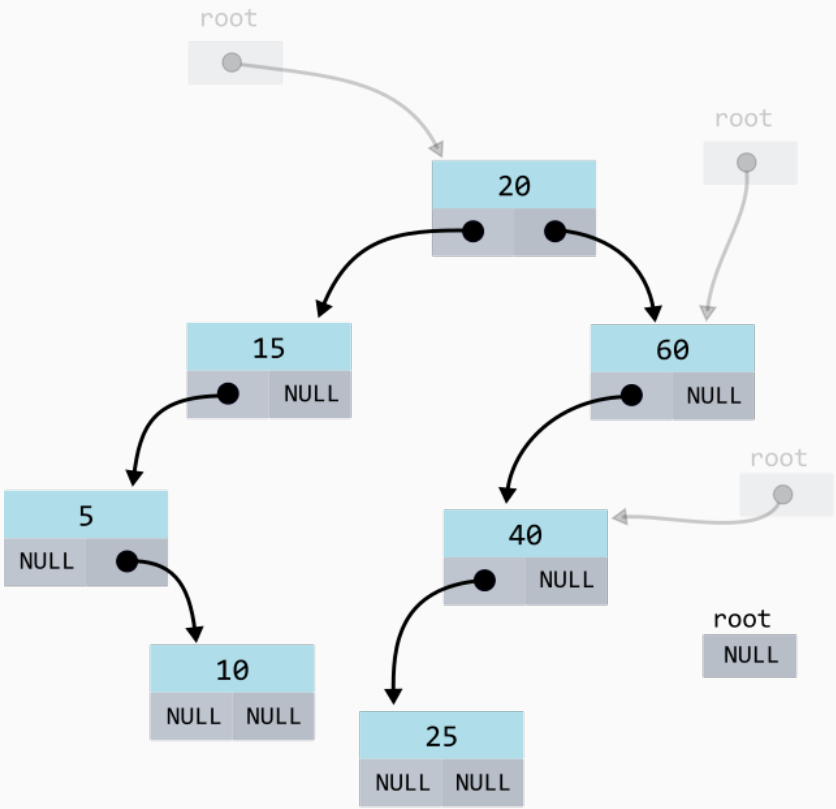
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
      bst_insert(root->right->left, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

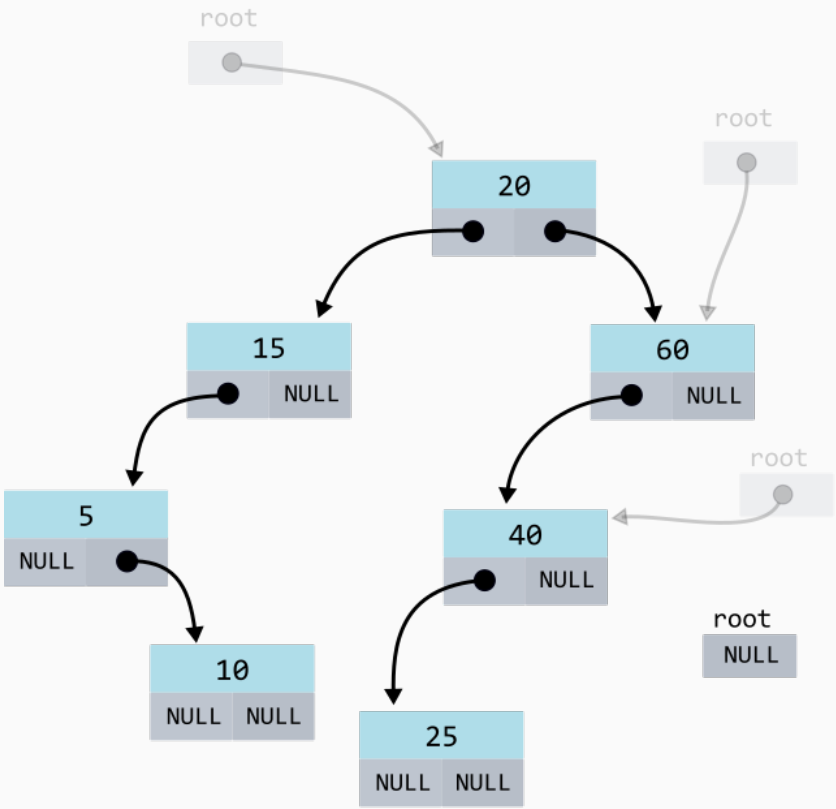
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
      bst_insert(root->right->left, 50)
         bst_insert(NULL, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

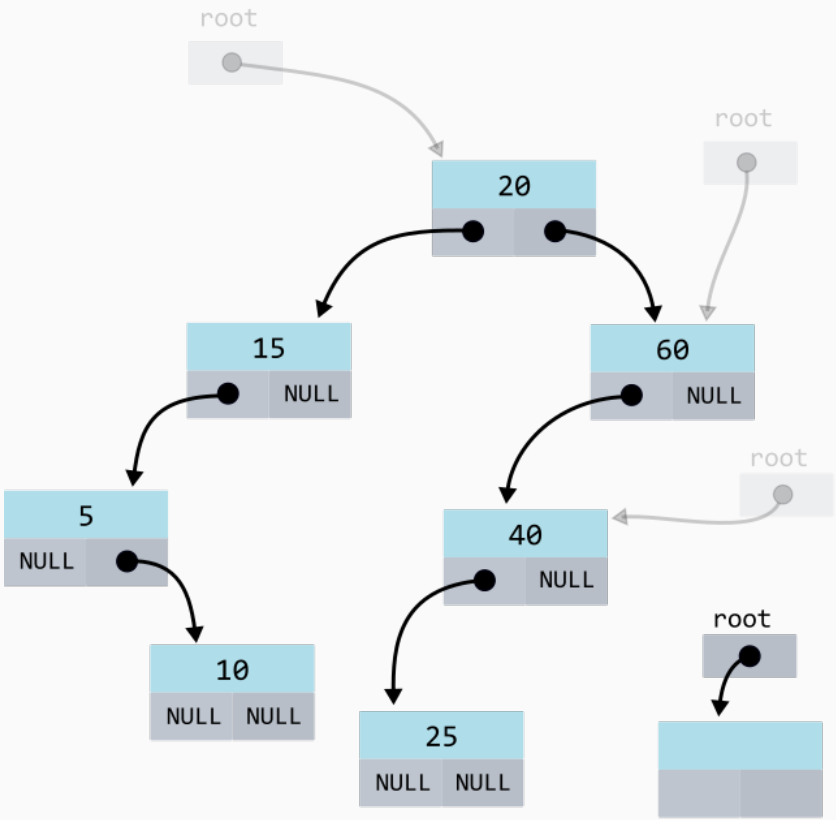
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
      bst_insert(root->right->left, 50)
         bst_insert(NULL, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

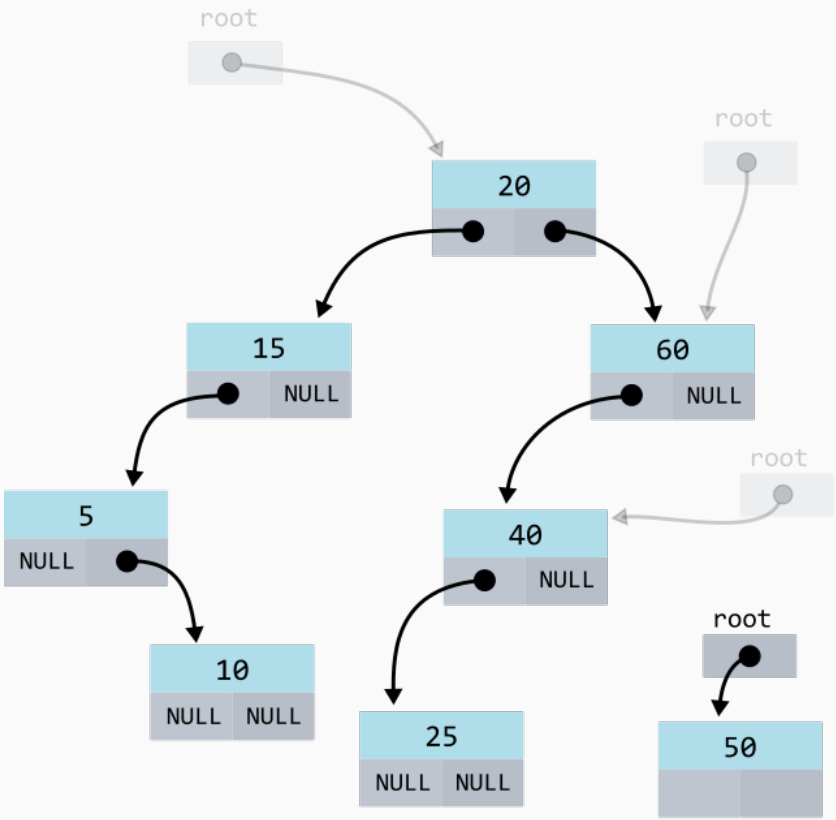
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
      bst_insert(root->right->left, 50)
         bst_insert(NULL, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

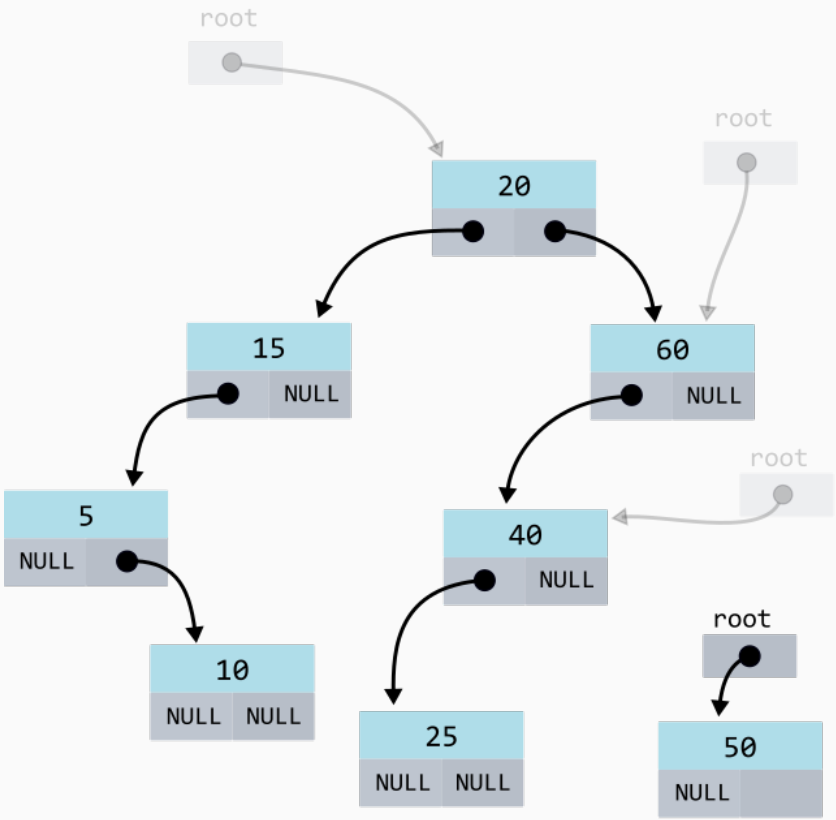
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
      bst_insert(root->right->left, 50)
         bst_insert(NULL, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

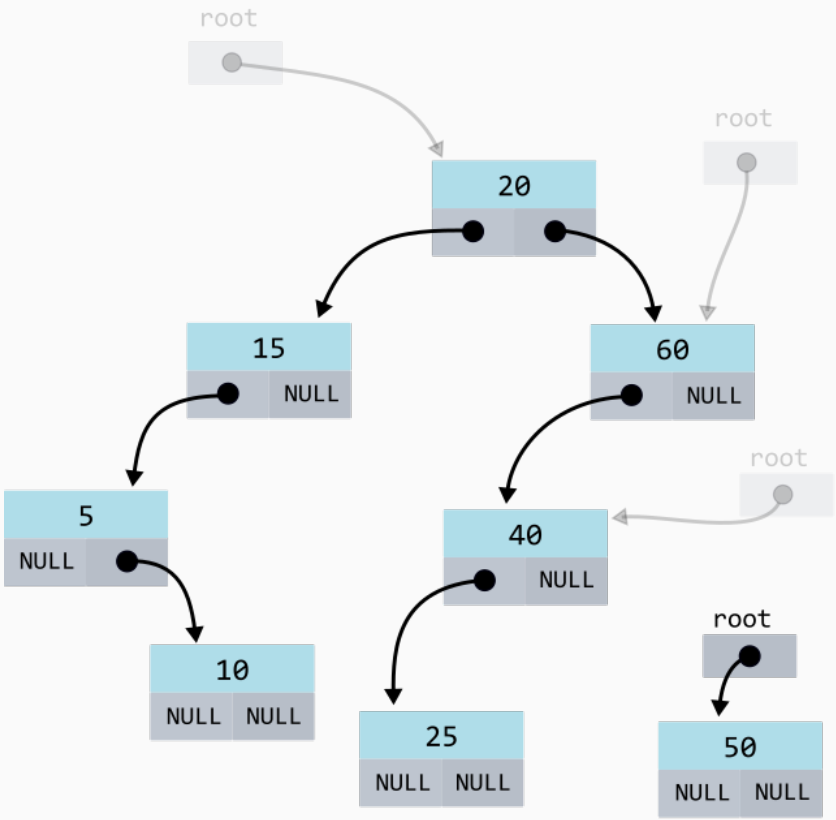
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
      bst_insert(root->right->left, 50)
         bst_insert(NULL, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

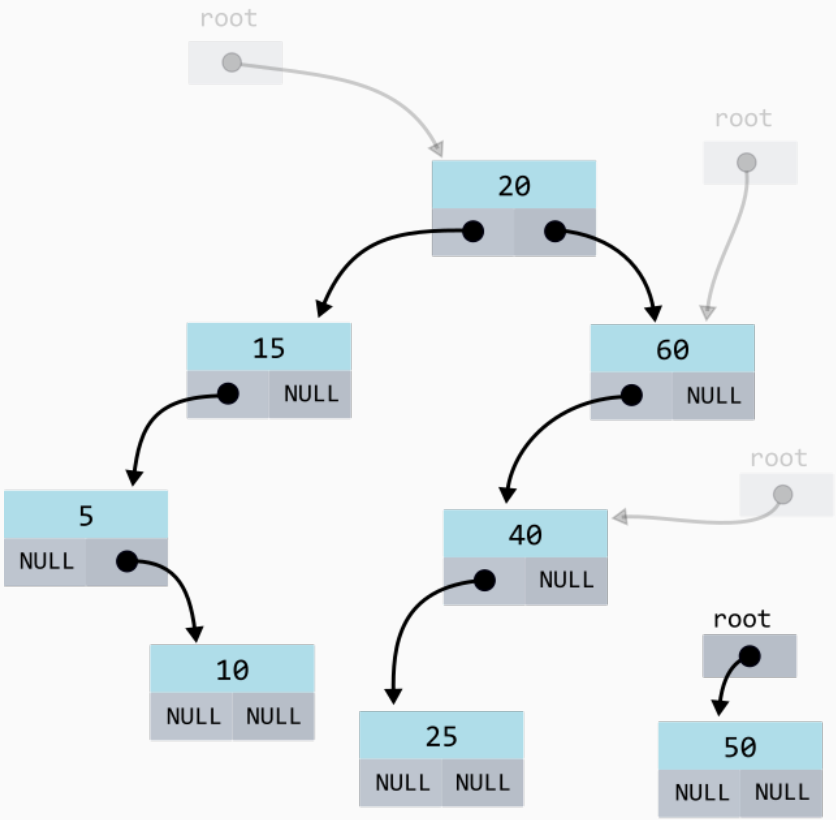
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
      bst_insert(root->right->left, 50)
         bst_insert(NULL, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

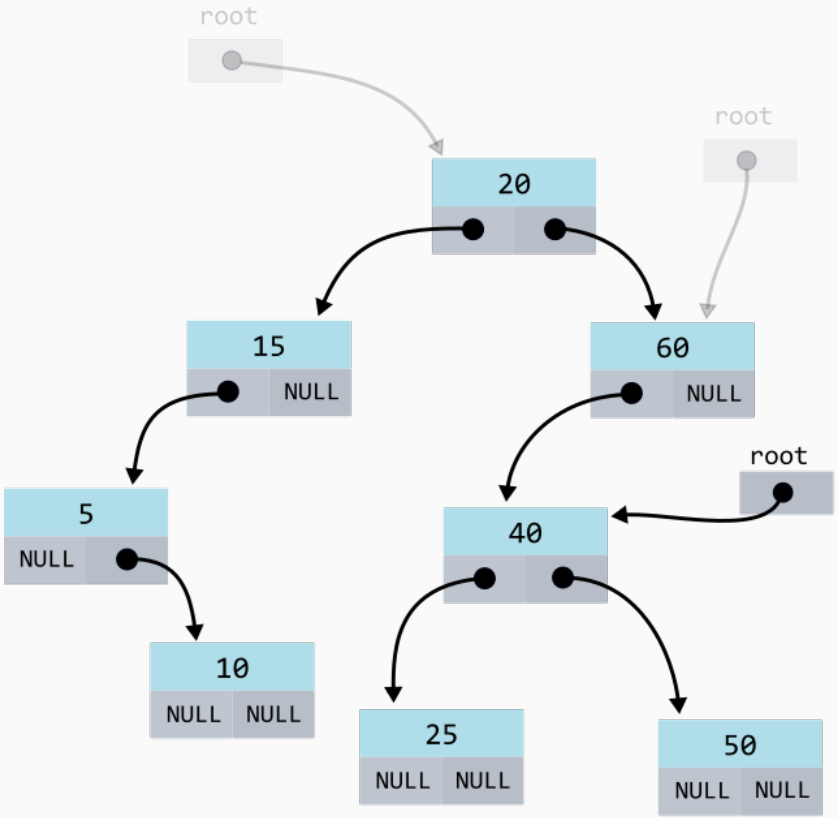
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
      bst_insert(root->right->left, 50)
         bst_insert(NULL, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

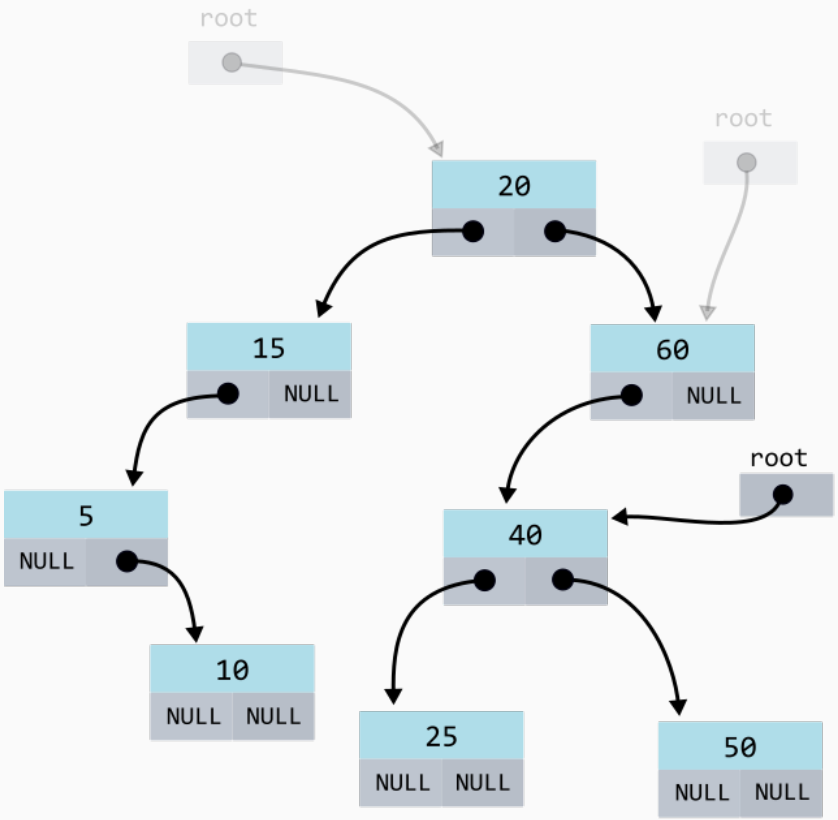
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
      bst_insert(root->right->left, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

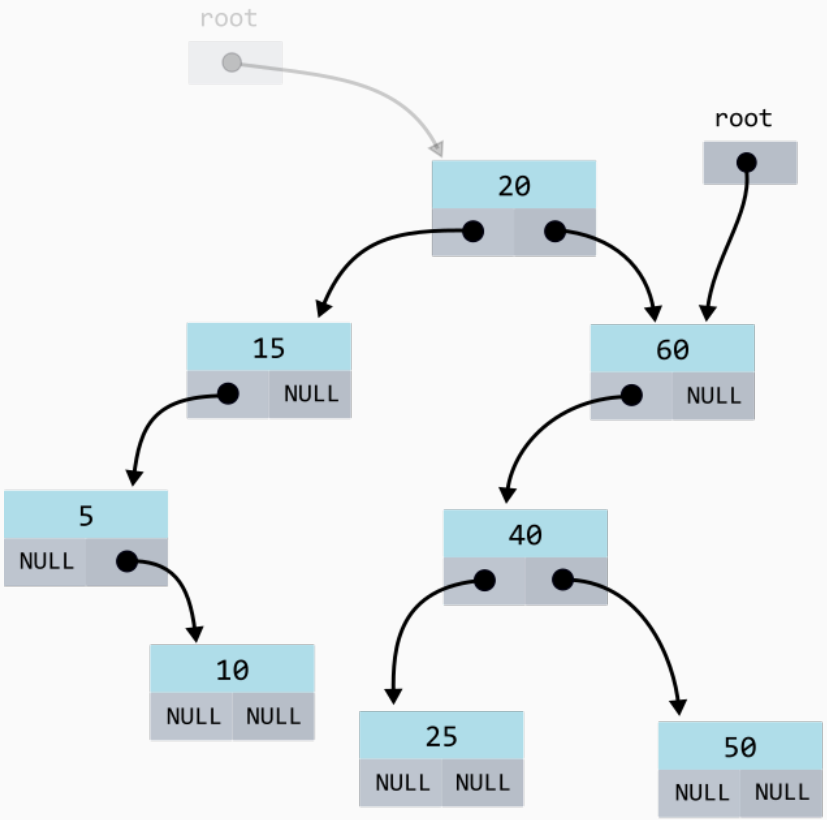
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
      bst_insert(root->right->left, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

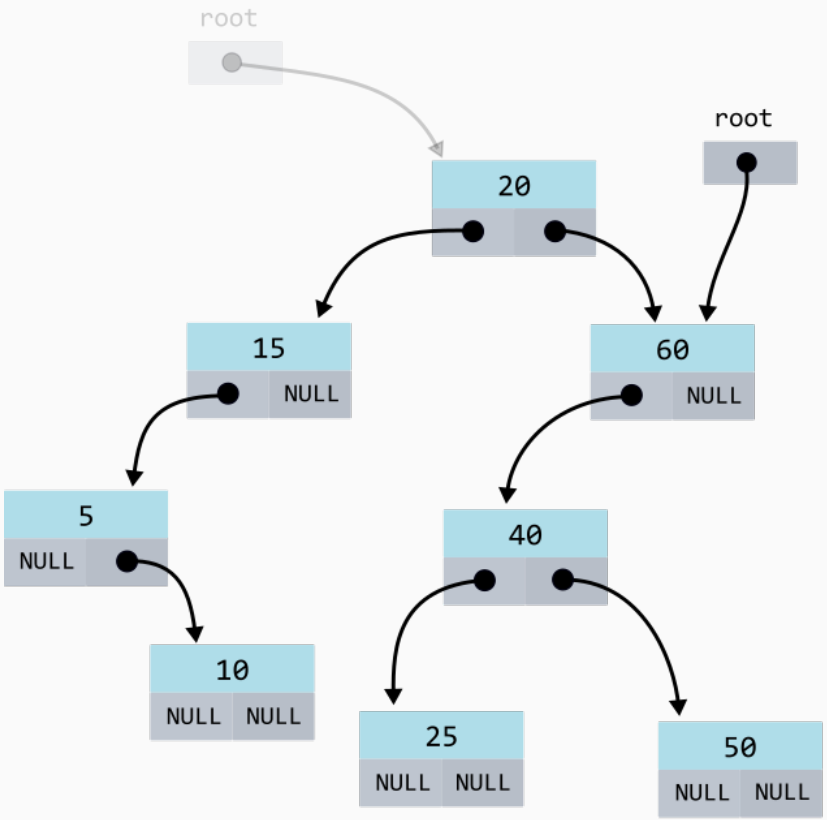
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

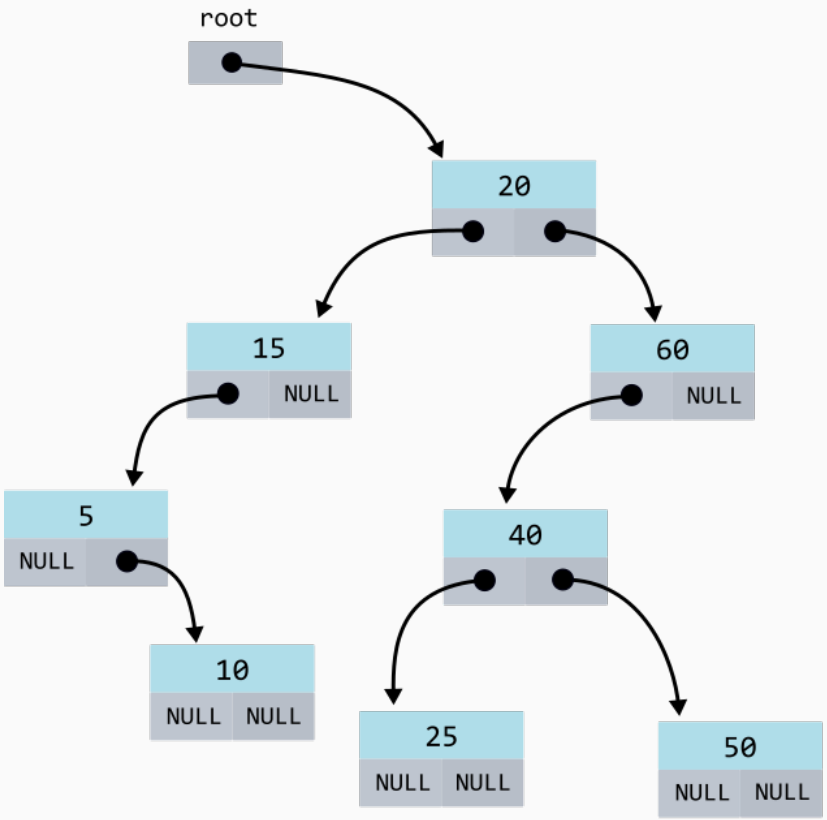
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
   bst_insert(root->right, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

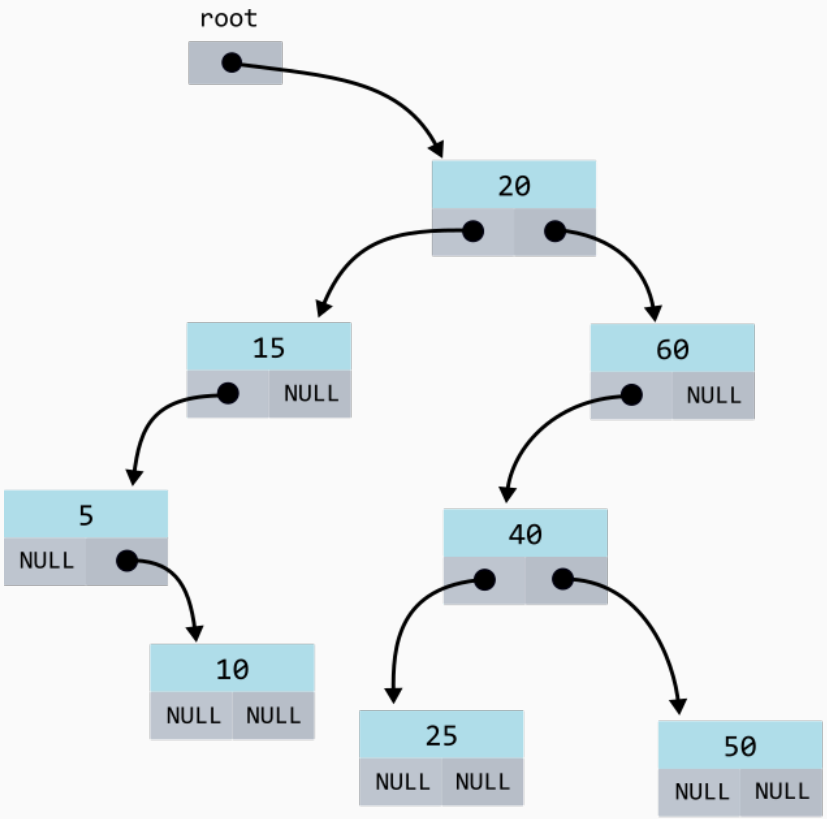
Добавление элемента в бинарное дерево поиска




```
bst_insert(root, 50)
```

```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```

Добавление элемента в бинарное дерево поиска




```
Node* bst_insert(Node* root, int x)
{
  if (root == NULL)
  {
    root = (Node*)malloc(sizeof(Node));
    root->value = x;
    root->left = NULL;
    root->right = NULL;
  }
  else if (x < root->value)
  {
    root->left = bst_insert(root->left, x);
  }
  else if (x > root->value)
  {
    root->right = bst_insert(root->right, x);
  }
  return root;
}
```