Разберем каждую часть кода:

1. \*\*Структура Node\*\*:

```cpp

struct Node

{

string str;

int v1;

double v2;

Node\* next;

};

```

Это определение структуры `Node`, которая содержит:

- `str`: строку (имя или описание).

- `v1`: целочисленное значение.

- `v2`: значение с плавающей запятой.

- `next`: указатель на следующий элемент в списке.

2. \*\*Функция pushFront\*\*:

```cpp

void pushFront(Node\*\* head, const string& str, int v1, double v2)

{

Node\* newNode = new Node{str, v1, v2, \*head};

\*head = newNode;

}

```

Эта функция добавляет новый элемент в начало списка. Она принимает указатель на указатель на голову списка и данные для нового узла. Новый узел создается и устанавливается в качестве новой головы списка.

3. \*\*Функция pushBack\*\*:

```cpp

void pushBack(Node\*\* head, const string& str, int v1, double v2)

{

Node\* newNode = new Node{str, v1, v2, nullptr};

if (\*head == nullptr)

\*head = newNode;

else

{

Node\* current = \*head;

while (current->next != nullptr)

current = current->next;

current->next = newNode;

}

}

```

Функция добавляет новый элемент в конец списка. Если список пуст, новый узел становится головой. В противном случае, функция проходит по списку до последнего элемента и добавляет новый узел после него.

4. \*\*Функция insertAfter\*\*:

```cpp

void insertAfter(Node\*\* head, int index, const string& str, int v1, double v2)

{

Node\* current = \*head;

for (int i = 0; current != nullptr && i < index; i++)

current = current->next;

if (current != nullptr)

{

Node\* newNode = new Node{str, v1, v2, current->next};

current->next = newNode;

}

}

```

Эта функция вставляет новый элемент после указанного индекса в списке. Если индекс больше длины списка, новый элемент не добавляется.

5. \*\*Функция insertBefore\*\*:

```cpp

void insertBefore(Node\*\* head, int index, const string& str, int v1, double v2)

{

if (index == 0)

pushFront(head, str, v1, v2);

else

{

Node\* current = \*head;

for (int i = 0; current != nullptr && i < index - 1; i++)

current = current->next;

if (current != nullptr)

{

Node\* newNode = new Node{str, v1, v2, current->next};

current->next = newNode;

}

}

}

```

Функция вставляет новый элемент перед указанным индексом в списке. Если индекс равен 0, используется `pushFront`. В противном случае, функция находит элемент перед указанным индексом и вставляет новый узел перед ним.

6. \*\*Функция removeByName\*\*:

```cpp

void removeByName(Node\*\* head, const string& str)

{

Node\* current = \*head;

Node\* prev = nullptr;

while (current != nullptr)

{

if (current->str == str)

{

if (prev == nullptr)

\*head = current->next;

else

prev->next = current->next;

delete current;

return;

}

prev = current;

current = current->next;

}

}

```

Функция удаляет элемент из списка по его строковому значению. Если удаляемый элемент является головой списка, голова меняется на следующий элемент. В противном случае, предыдущий элемент указывает на следующий за удаляемым.

7. \*\*Функция printList\*\*:

```cpp

void printList(const Node\* head)

{

while (head != nullptr)

{

cout << "String: " << head->str << ", Int: " << head->v1 << ", Double: " << head->v2 << endl;

head = head->next;

}

}

```

Эта функция выводит все элементы списка на экран. Она проходит по списку, начиная с головы, и выводит данные каждого узла.

8. \*\*Основная функция main\*\*:

```cpp

int main()

{

Node\* head = nullptr;

pushFront(&head, "First", 1, 1.5);

pushBack(&head, "Second", 2, 2.5);

pushFront(&head, "Third", 3, 3.5);

pushBack(&head, "Fourth", 4, 4.5);

printList(head);

insertBefore(&head, 3,"Fifth", 5, 5.5);

insertAfter(&head, 3,"Sixth", 6, 6.5);

printList(head);

removeByName(&head, "Fifth");

printList(head);

while (head != nullptr)

{

Node\* temp = head;

head = head->next;

delete temp;

}

return 0;

}

```

В функции `main` создается пустой список, добавляются элементы с помощью `pushFront` и `pushBack`, выводятся на экран, вставляются новые элементы с помощью `insertBefore` и `insertAfter`, удаляется элемент с помощью `removeByName`, и список выводится на экран еще раз. Затем происходит очистка списка, удаляются все созданные узлы, чтобы избежать утечки памяти.