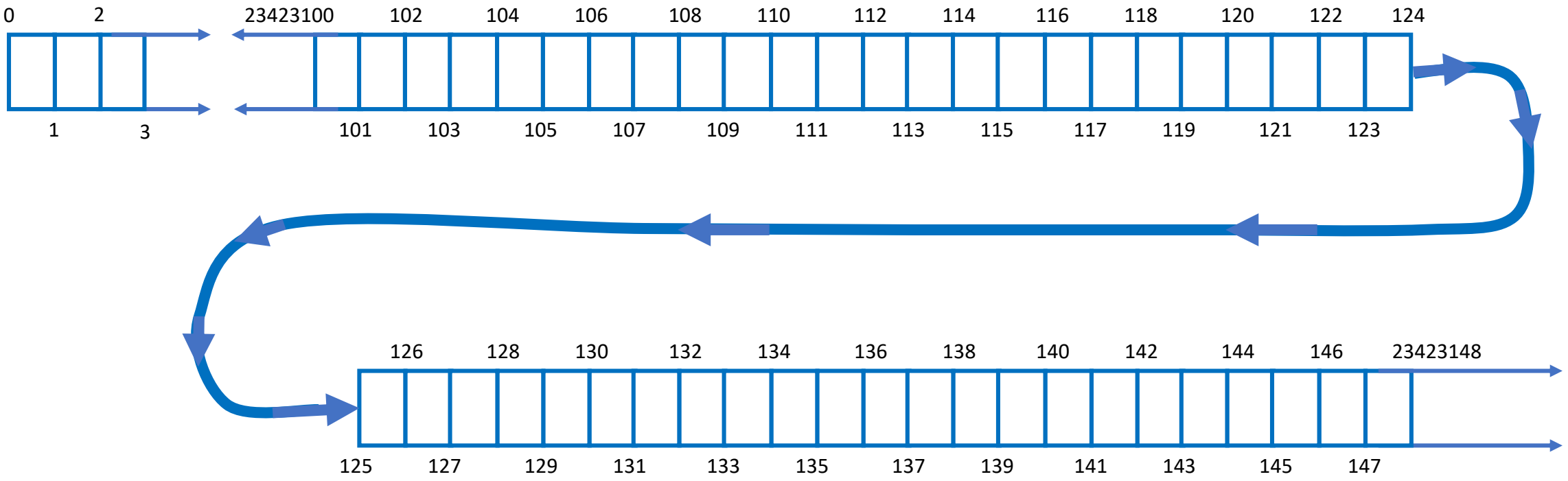
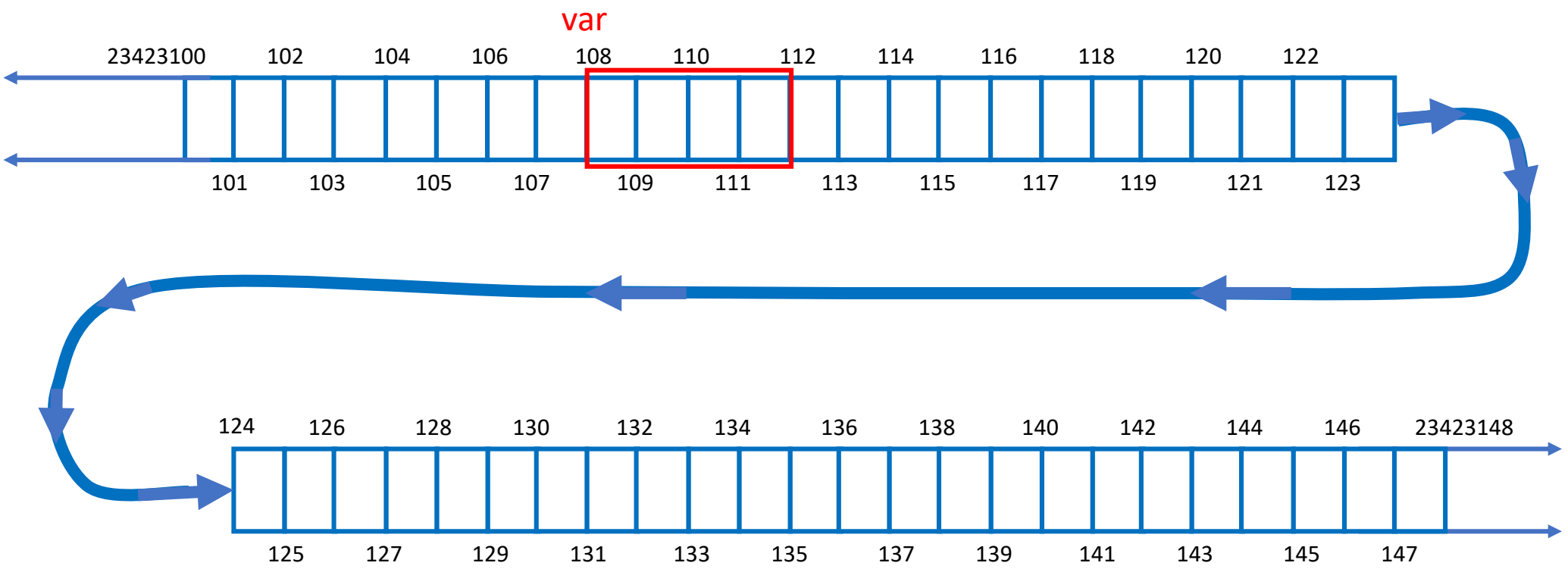
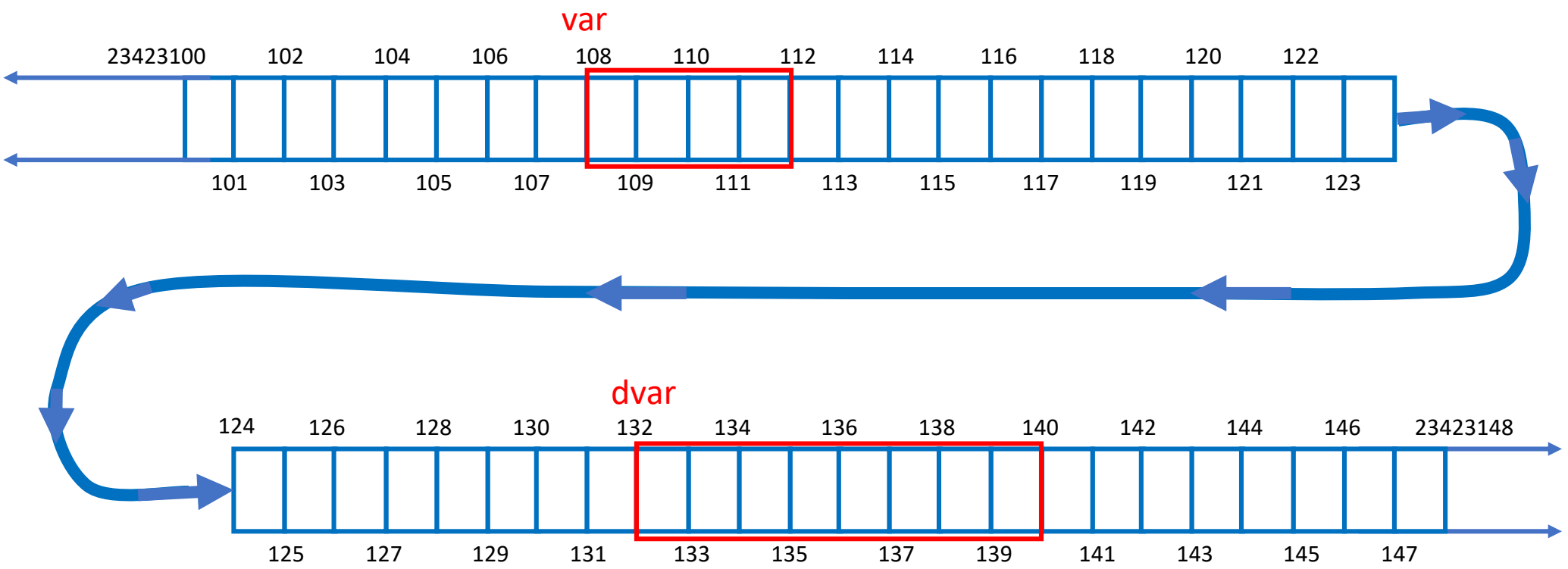
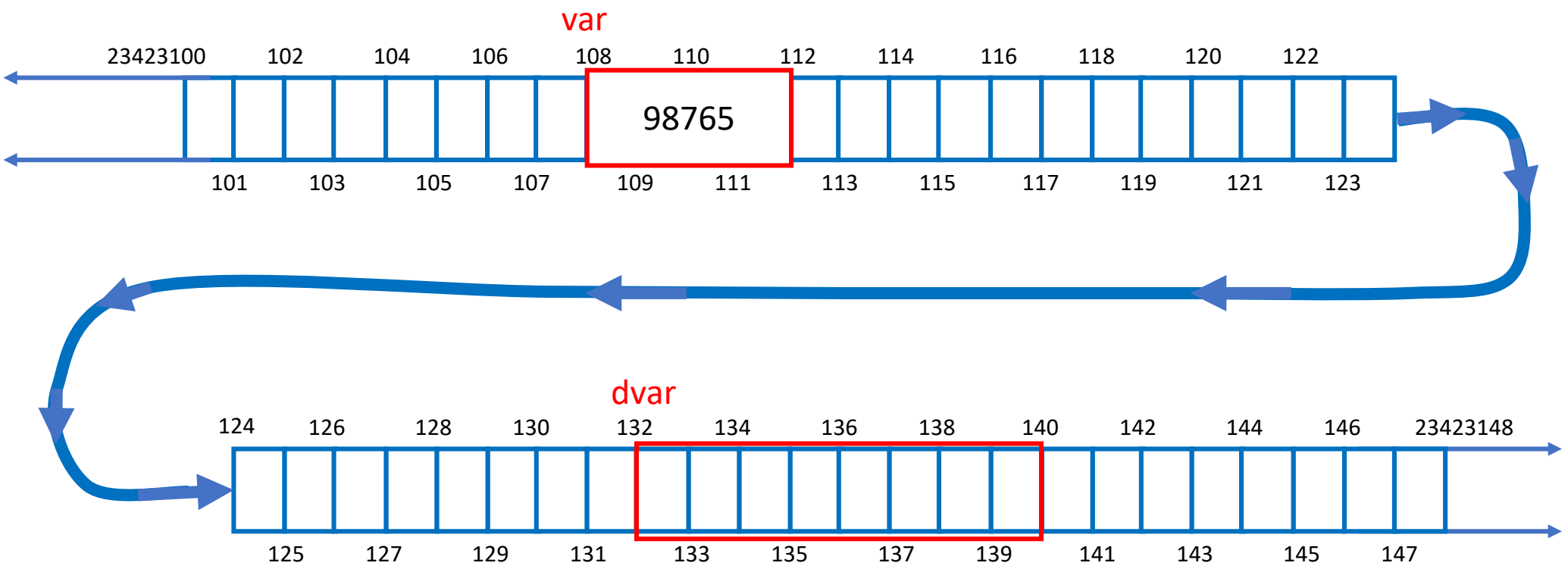

0
2
23423100
102
104
106
108
110
112
114
116
118
120
122
124
1
3
101
103
105
107
109
111
113
115
117
119
121
123
126
128
130
132
134
136
138
140
142
144
146
23423148
125
127
129
131
133
135
137
139
141
143
145
147


var
23423100
102
104
106
108
110
112
114
116
118
120
122
101
103
105
107
109
111
113
115
117
119
121
123
124
126
128
130
132
134
136
138
140
142
144
146
23423148
125
127
129
131
133
135
137
139
141
143
145
147

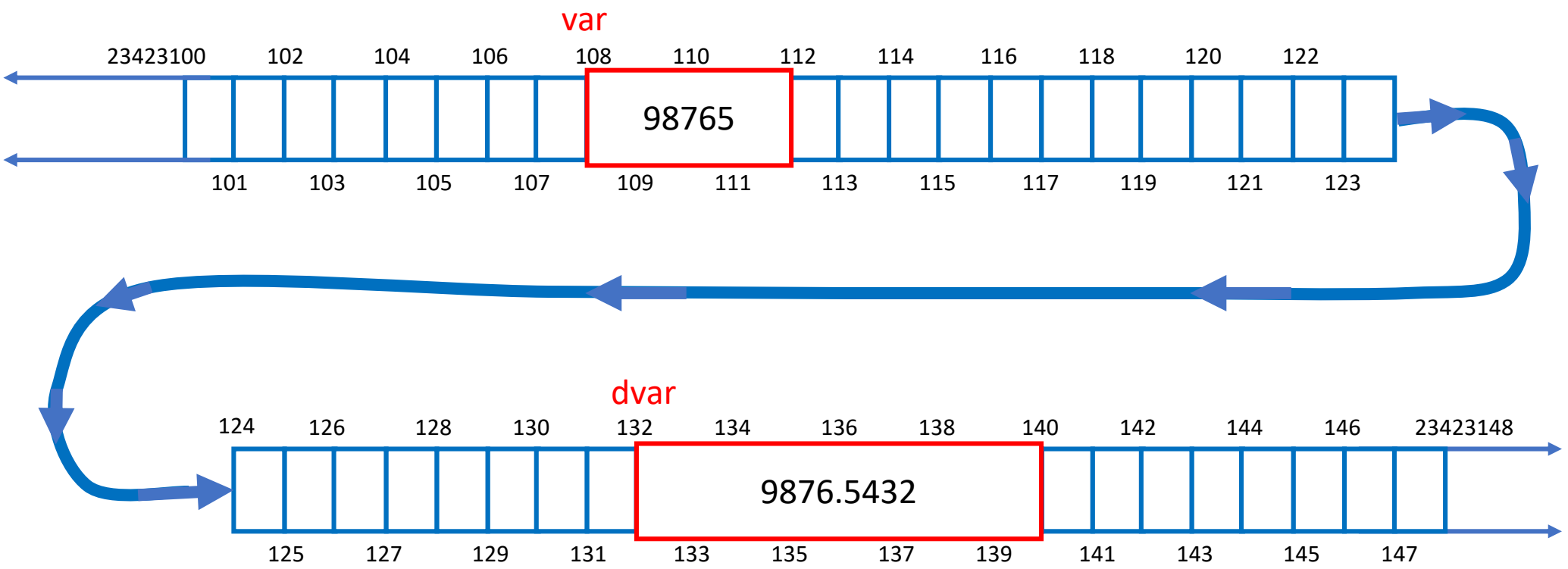
```
int var;
```



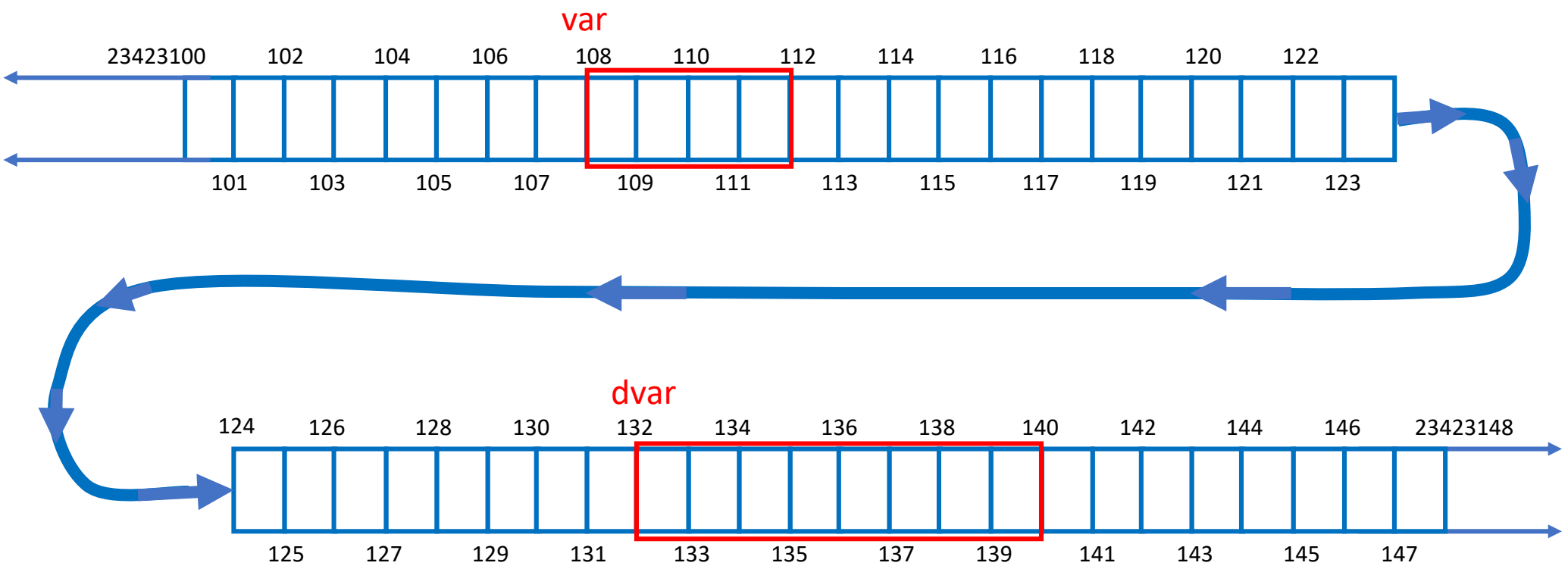

```
int var;
double dvar;
```



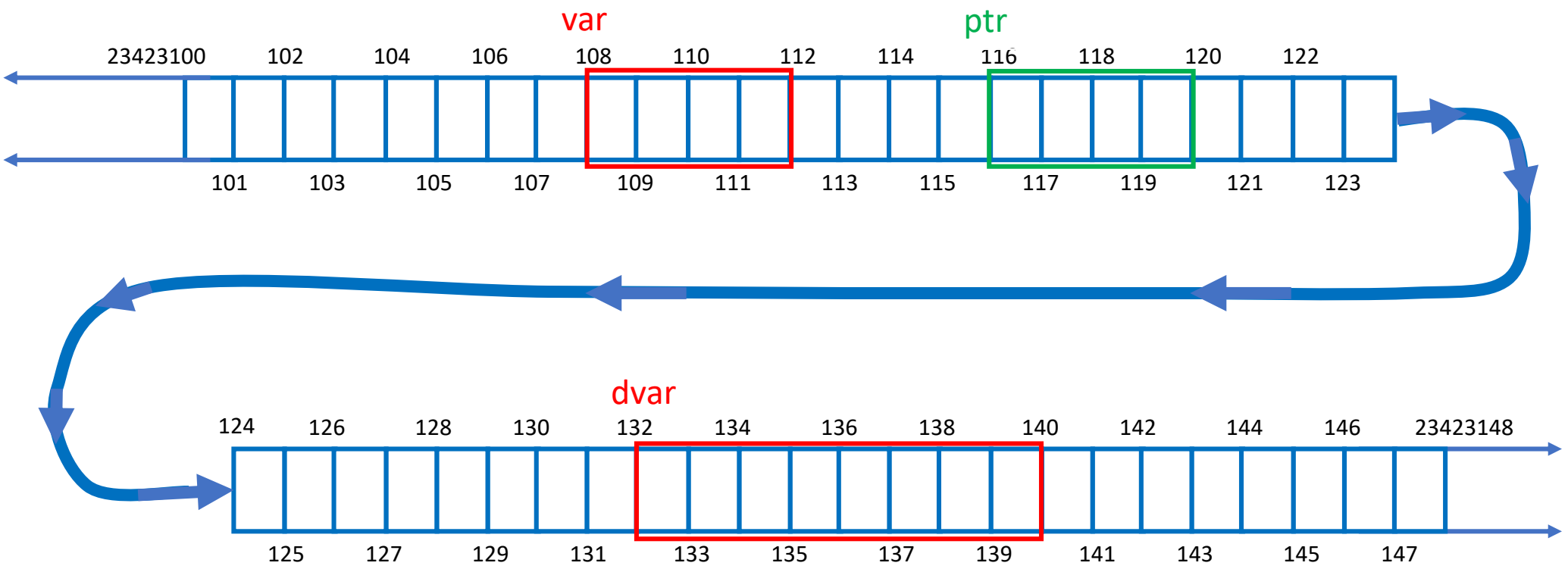

```
int var;
double dvar;
var = 98765;
```



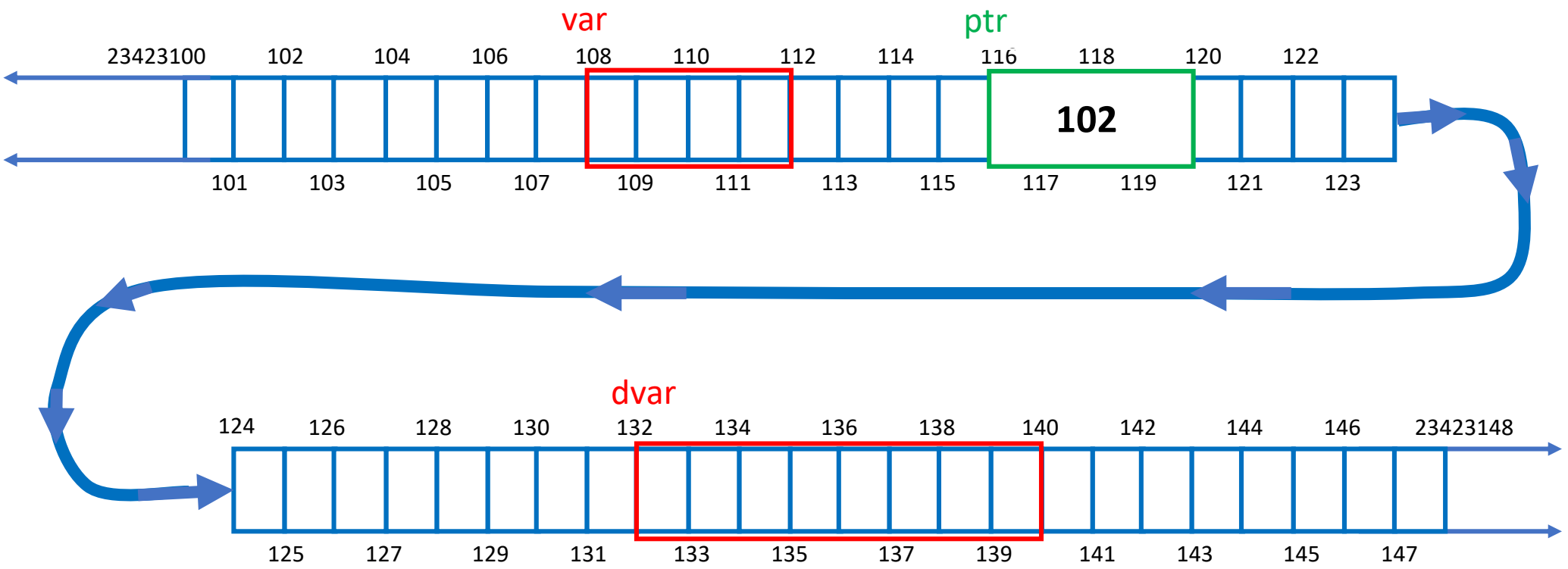

```
int var;
double dvar;
var = 98765;
dvar = 9876.5432;
```



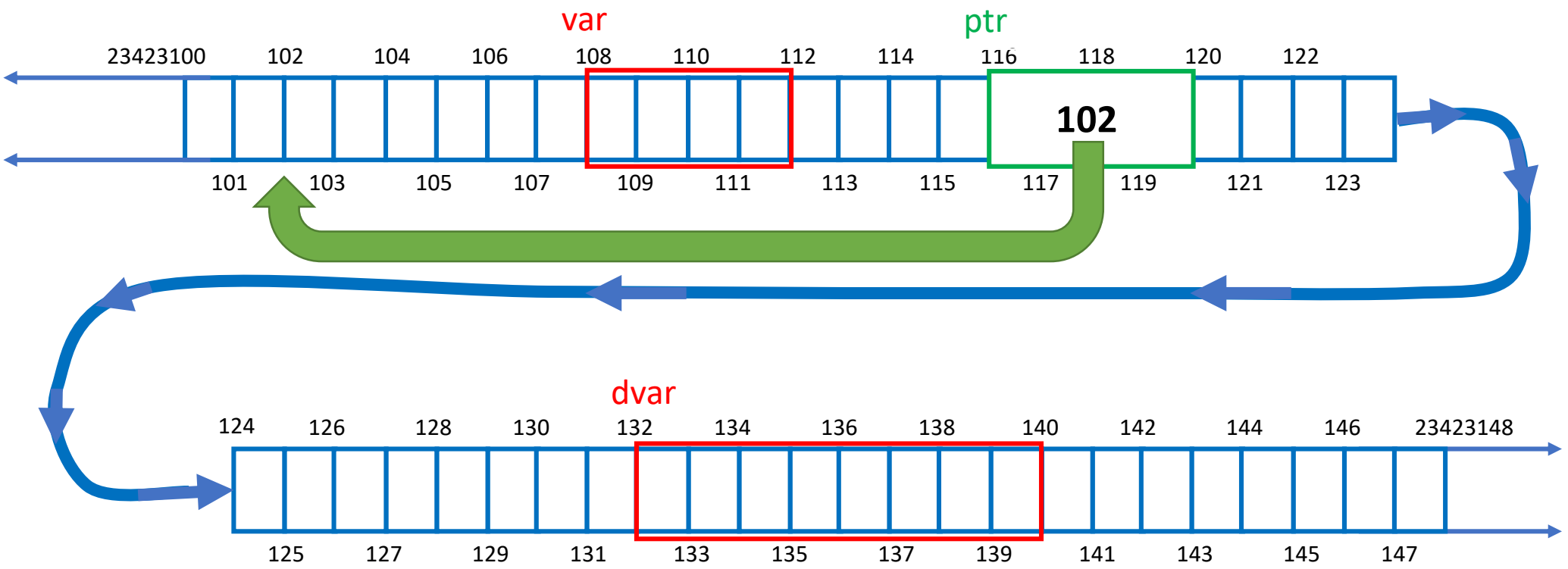

```
int var;
double dvar;
```



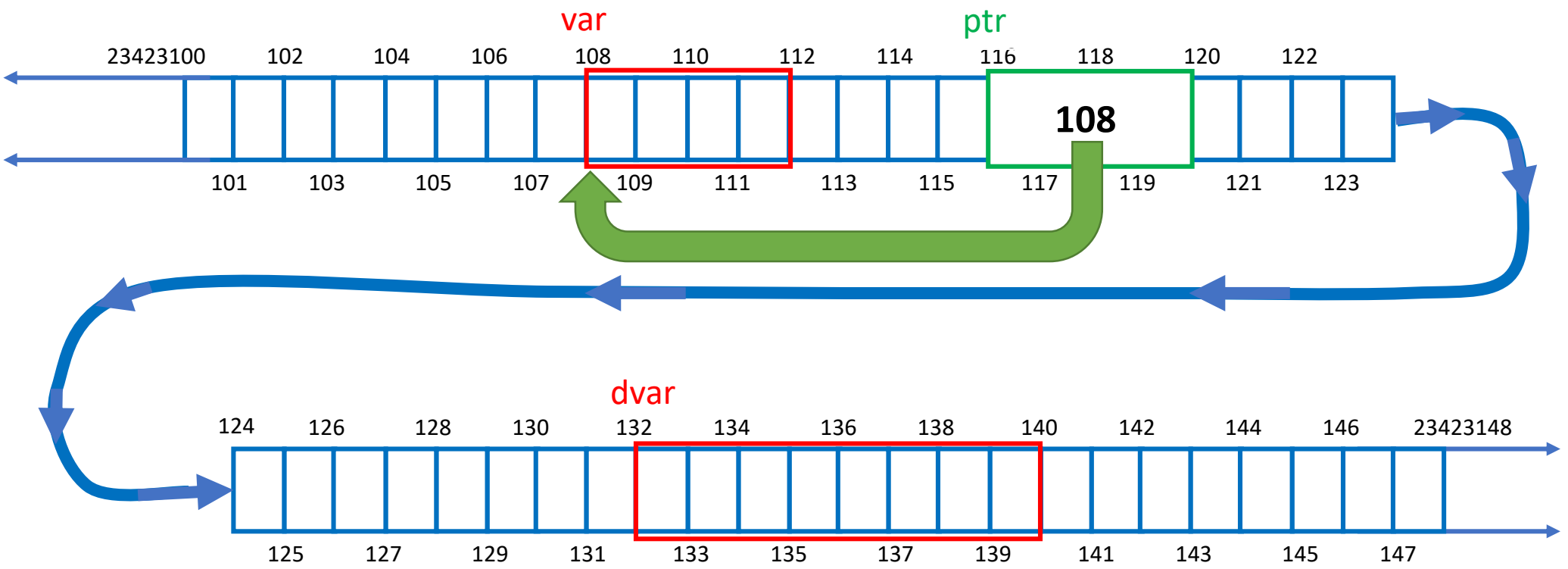

```
int var;
double dvar;
Pointer ptr;
```



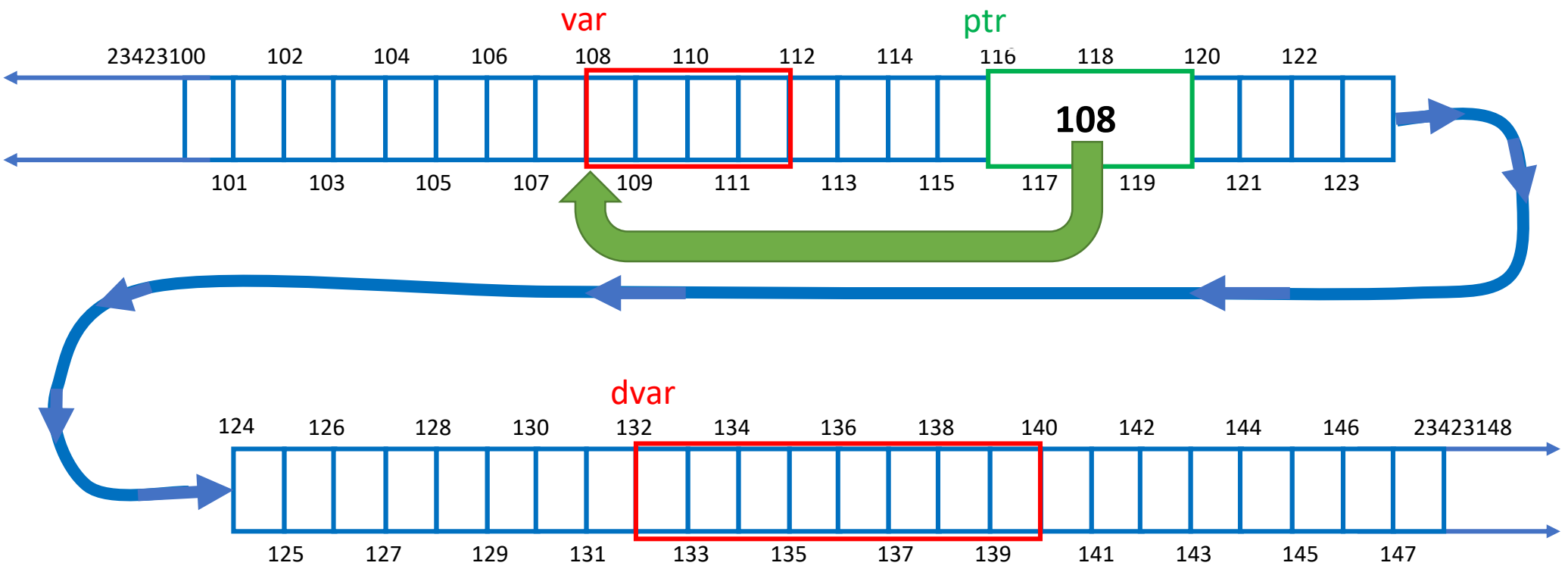

```
int var;
double dvar;
Pointer ptr;
ptr = 102;
```



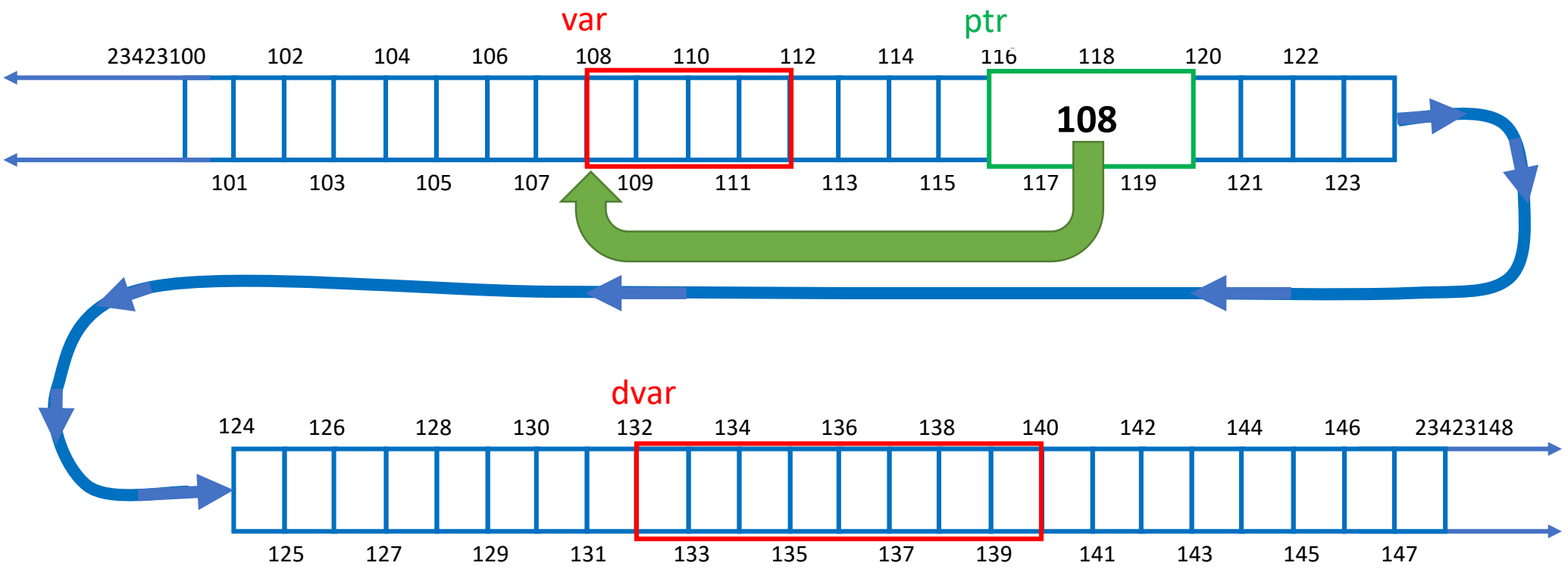

```
int var;
double dvar;
Pointer ptr;
ptr = 102;
```




```
int var;
double dvar;
Pointer ptr;
ptr = 108;
```

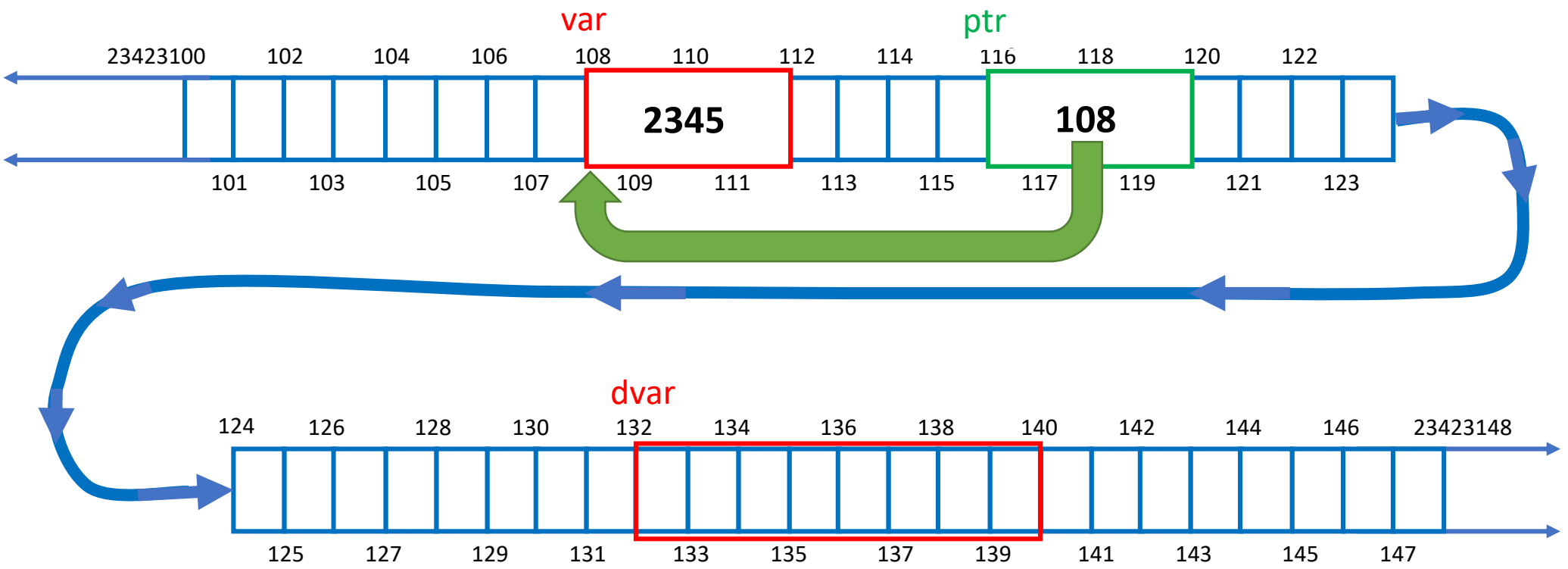



```
int var;
double dvar;
Pointer ptr;
ptr = AddressOf var;
```



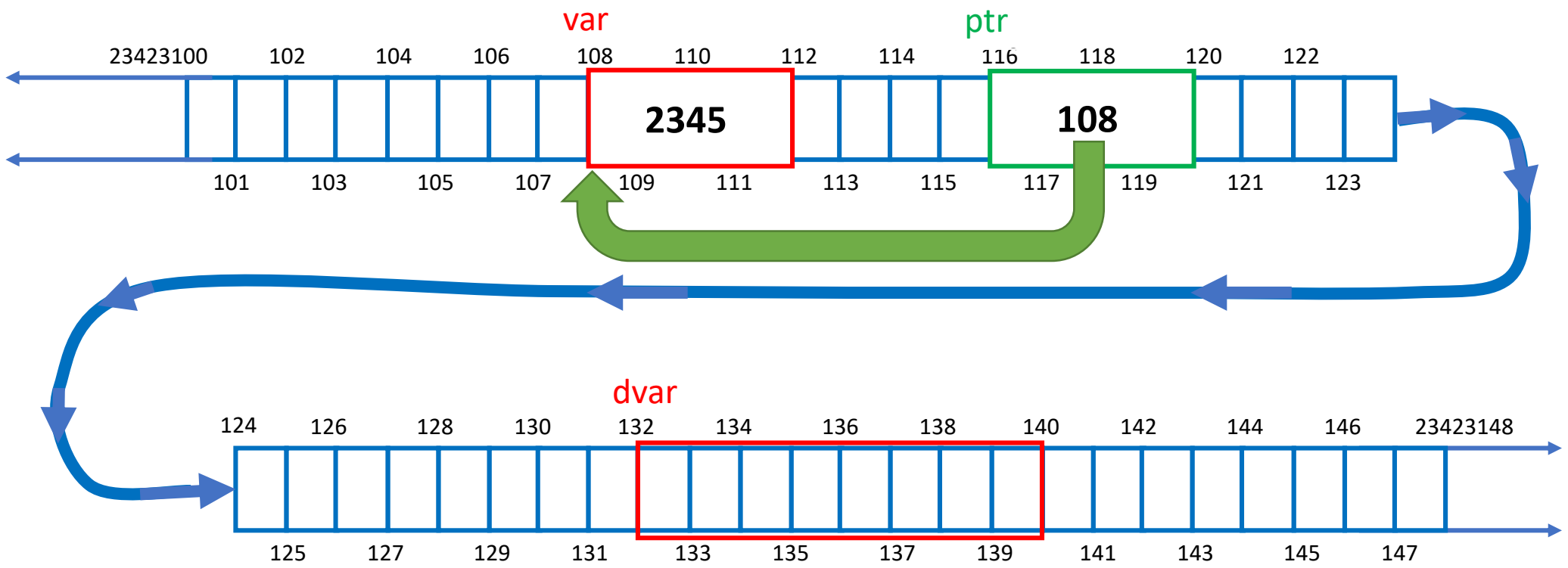

```
int var;
double dvar;
Pointer ptr;
ptr = AddressOf var;
```

```
TargetOf ptr = 2345;
```



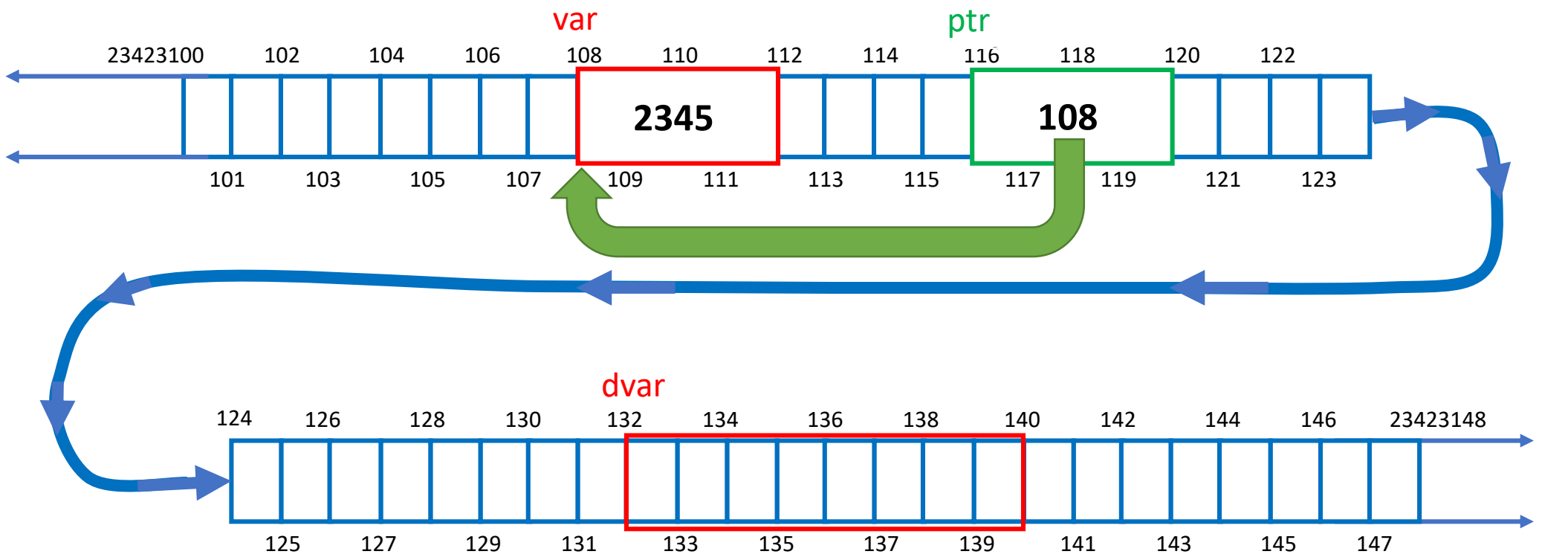

```
int var;
double dvar;
Pointer ptr;
ptr = AddressOf var;
```

```
TargetOf ptr = 2345;
```




```
int var;
double dvar;
Pointer ptr;
ptr = AddressOf var;
```

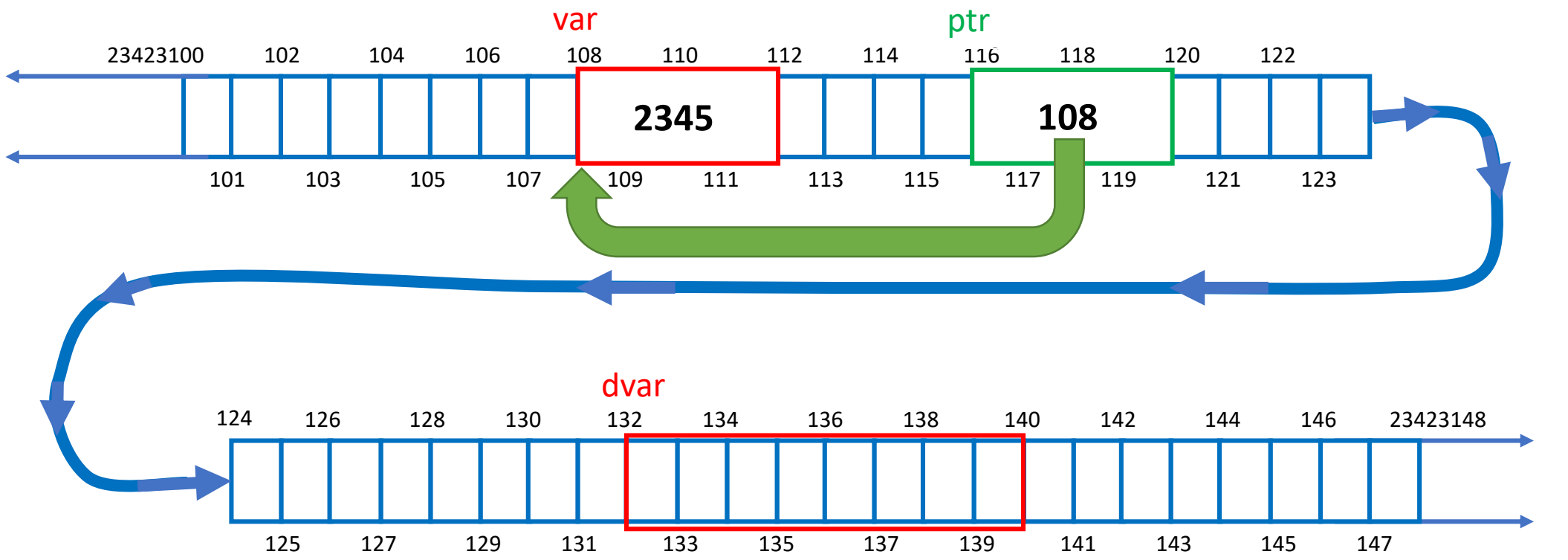
```
TargetOf ptr = 2345;
printf("%d\n", var);
```




```
int var;
double dvar;
Pointer ptr;
ptr = AddressOf var;
```

```
TargetOf ptr = 2345;
printf("%d\n", var);
```

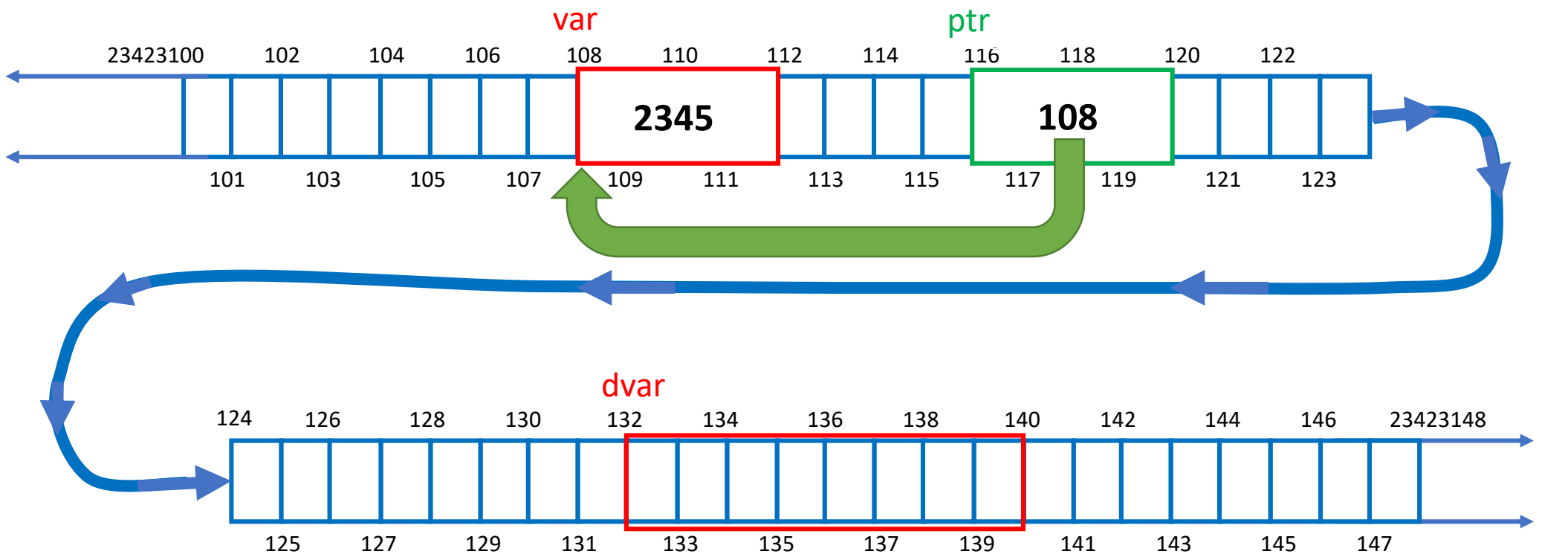
OUTPUT:
2345




```
int var;
double dvar;
Pointer ptr;
ptr = AddressOf var;
```

```
TargetOf ptr = 2345;
printf("%d\n", var);
printf("%d\n", TargetOf ptr );
```

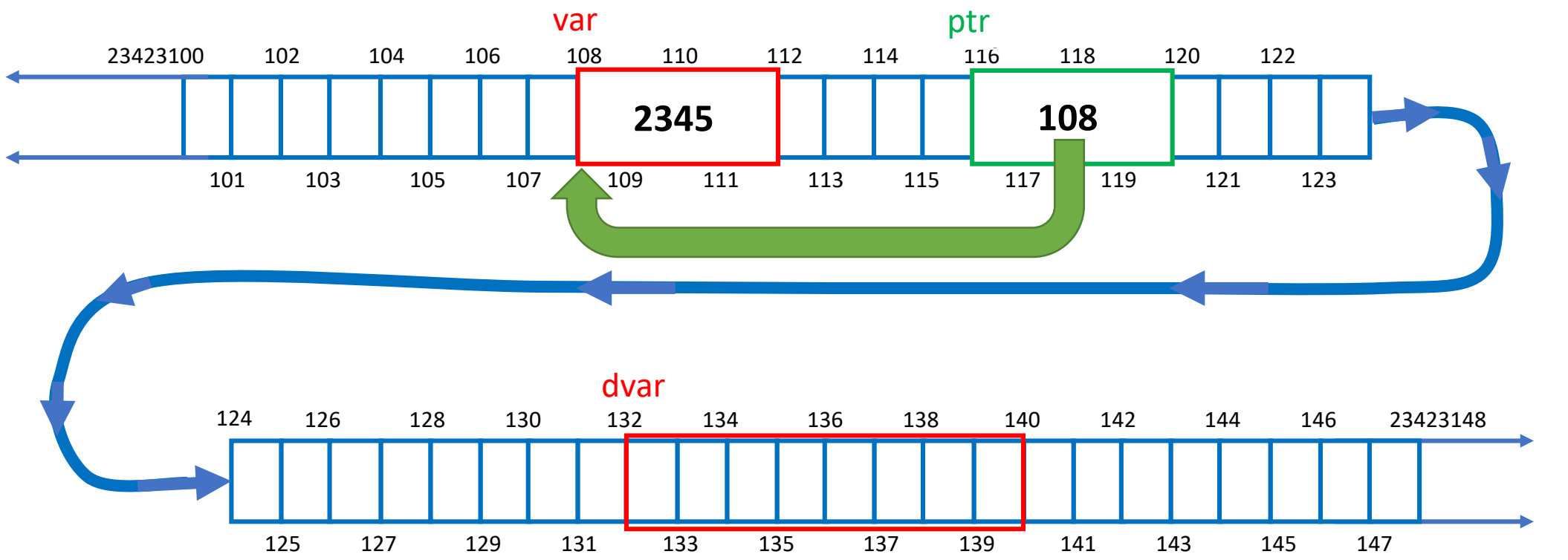
OUTPUT:
2345




```
int var;
double dvar;
Pointer ptr;
ptr = AddressOf var;
```

```
TargetOf ptr = 2345;
printf("%d\n", var);
printf("%d\n", TargetOf ptr );
```

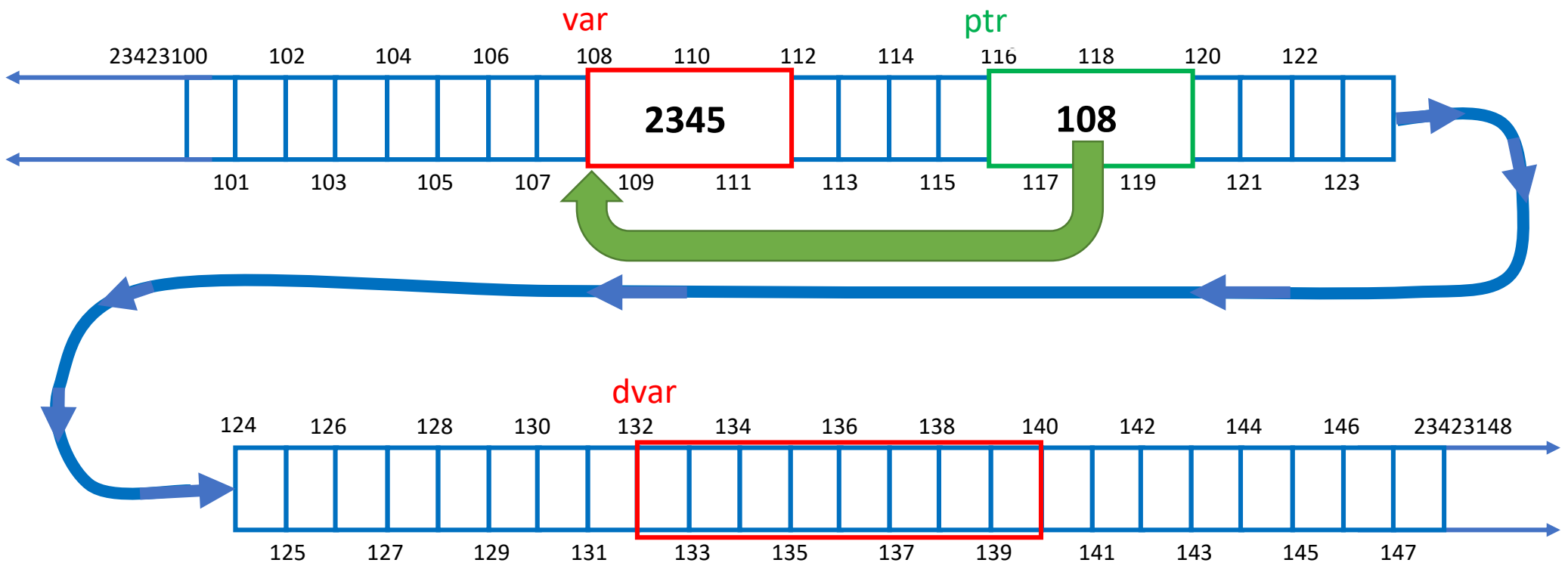
OUTPUT:
2345
2345




```
int var;
double dvar;
Pointer ptr;
ptr = AddressOf var;
```

```
TargetOf ptr = 2345;
printf("%d\n", var);
printf("%d\n", TargetOf ptr );
printf("%u\n", ptr );
```

OUTPUT:
2345
2345




```
int var;
double dvar;
Pointer ptr;
ptr = AddressOf var;
```

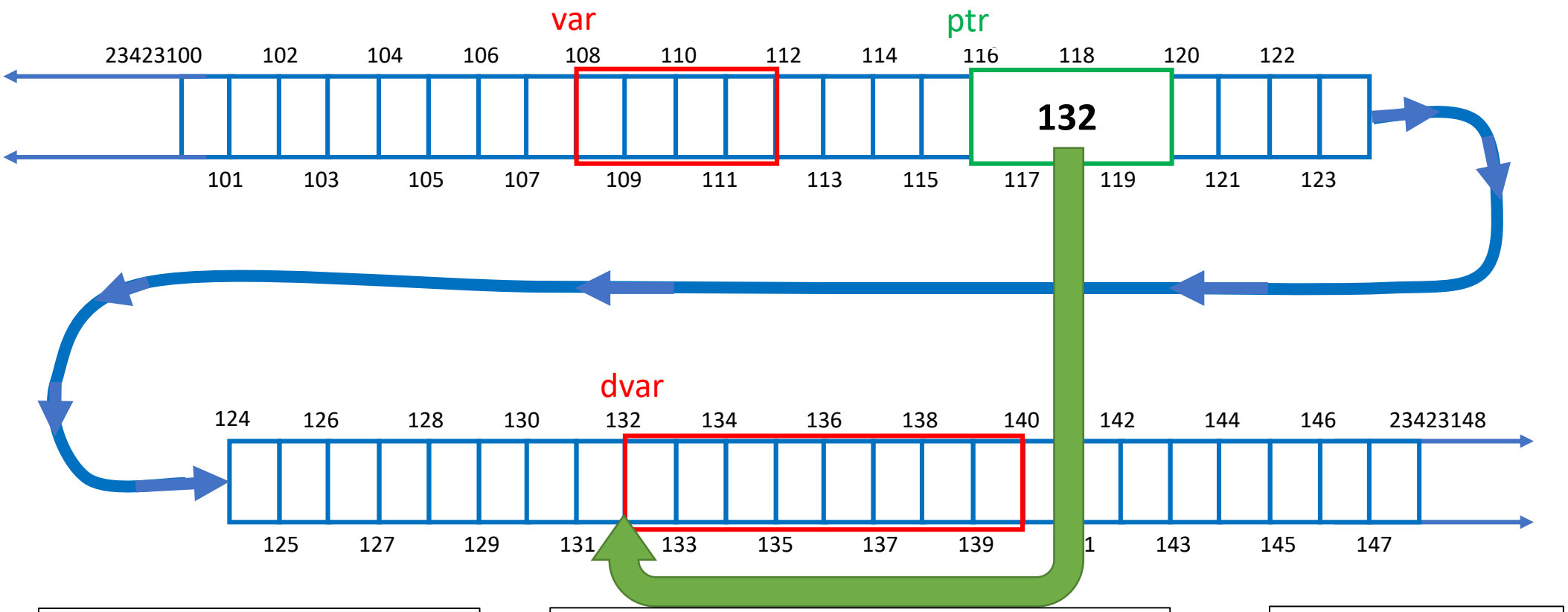
```
TargetOf ptr = 2345;
printf("%d\n", var);
printf("%d\n", TargetOf ptr );
printf("%u\n", ptr );
```

OUTPUT:

2345

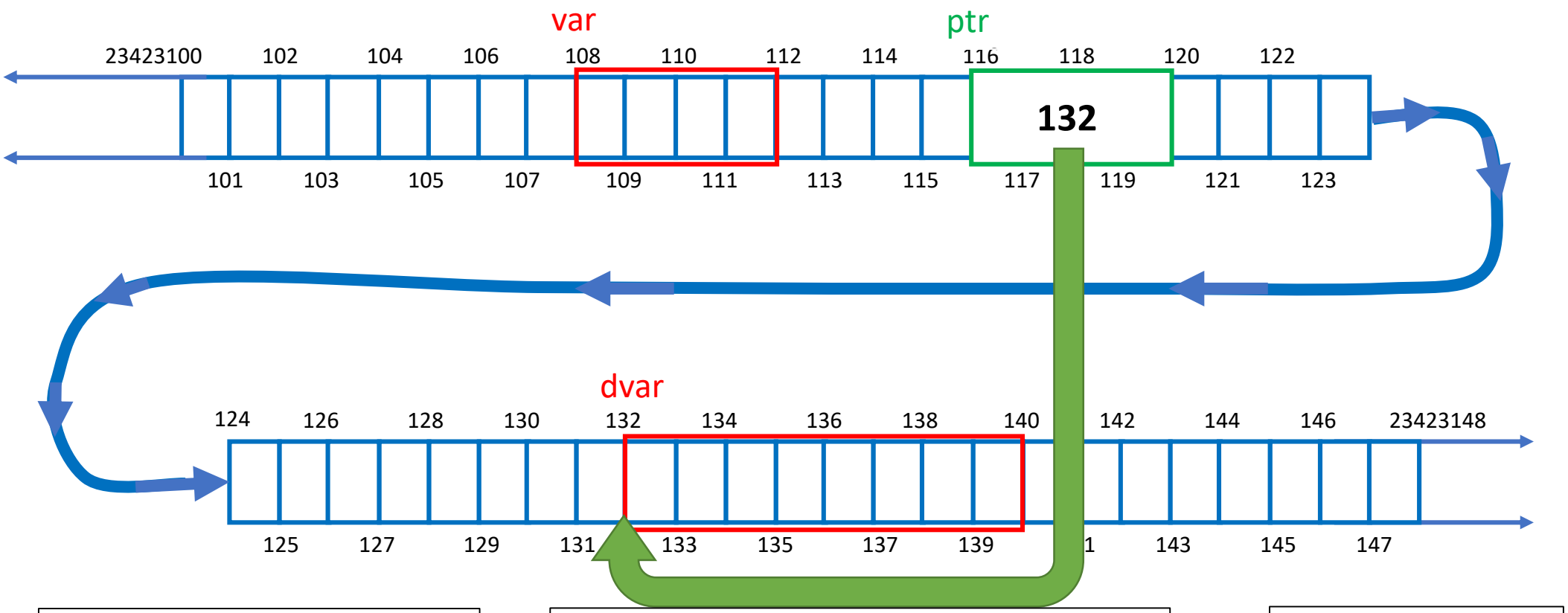
2345

23423**108**




```
int var;
double dvar;
Pointer ptr;
ptr = AddressOf dvar;
```

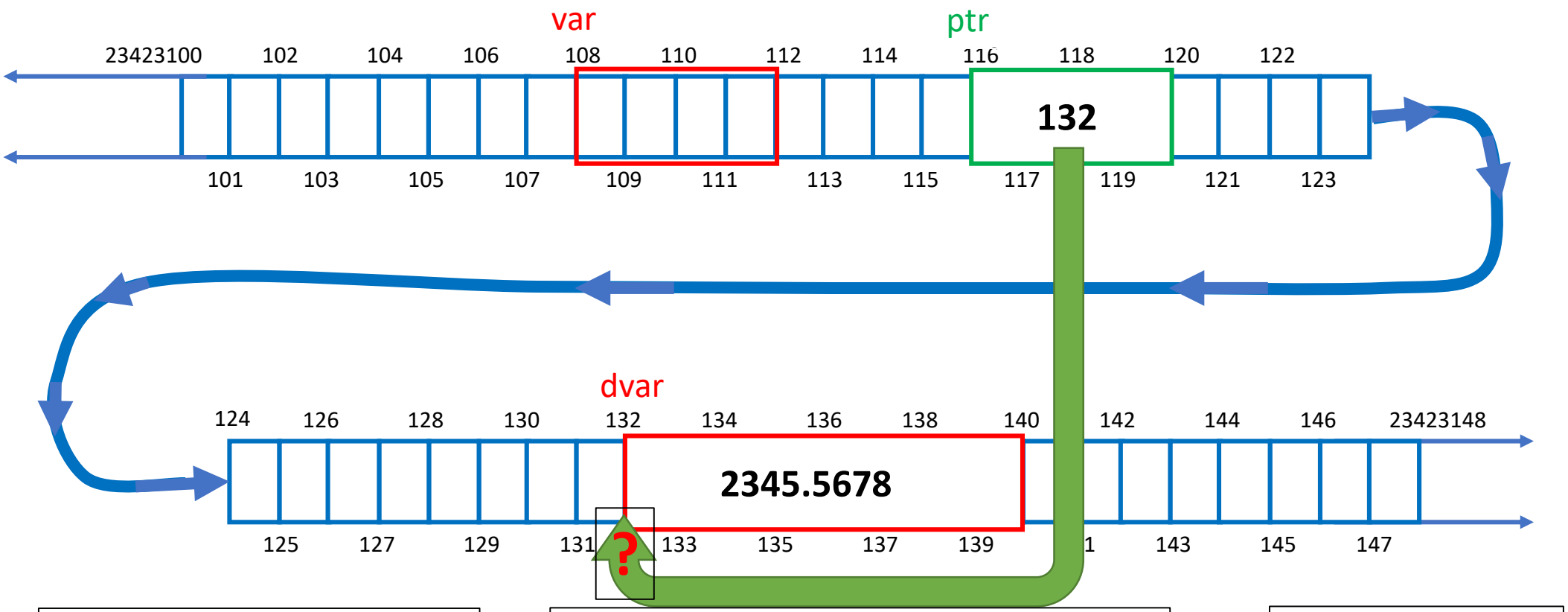
OUTPUT:




```
int var;
double dvar;
Pointer ptr;
ptr = AddressOf dvar;
```

```
TargetOf ptr = 2345.5678;
```

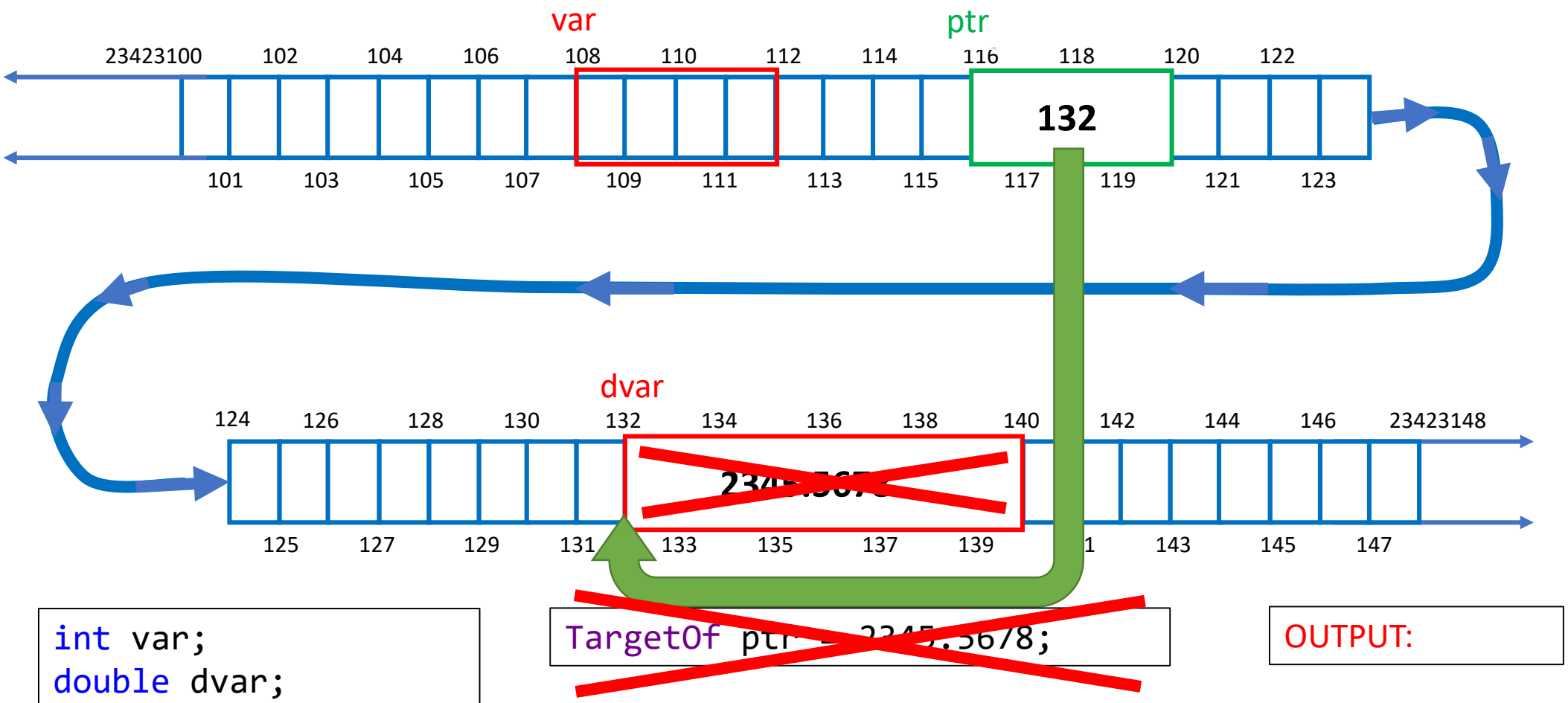
OUTPUT:




```
int var;
double dvar;
Pointer ptr;
ptr = AddressOf dvar;
```

```
TargetOf ptr = 2345.5678;
```

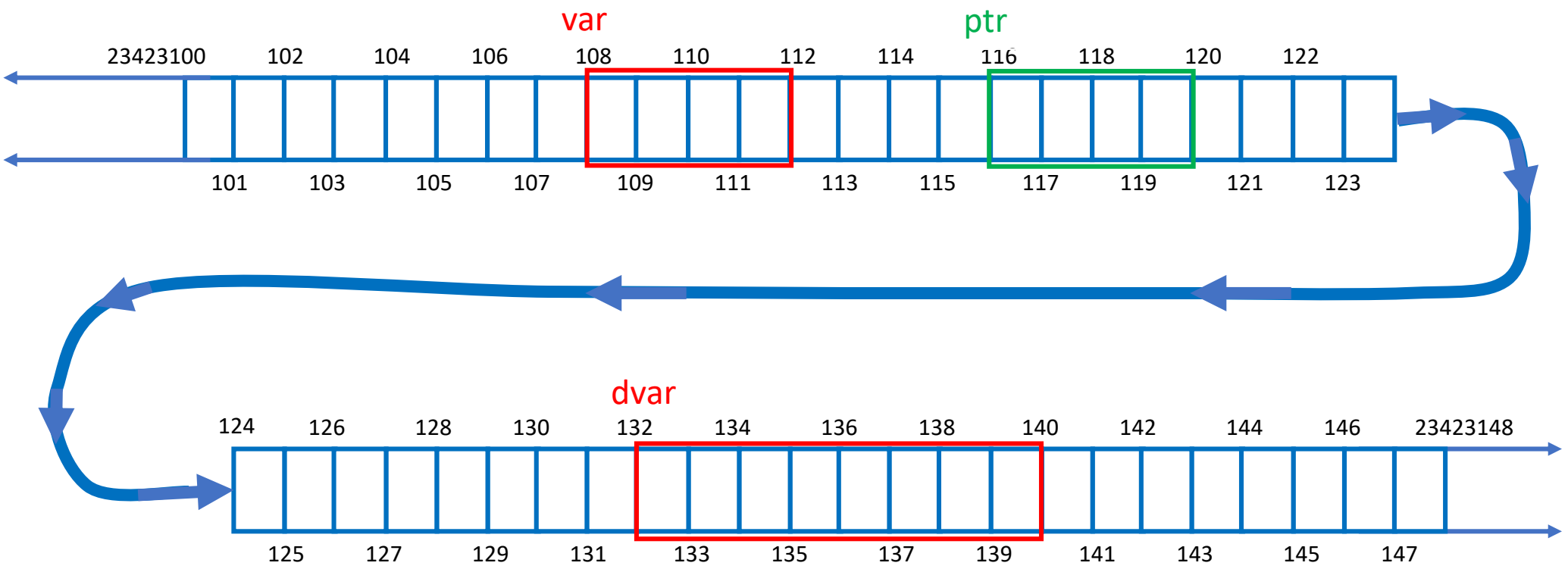
OUTPUT:



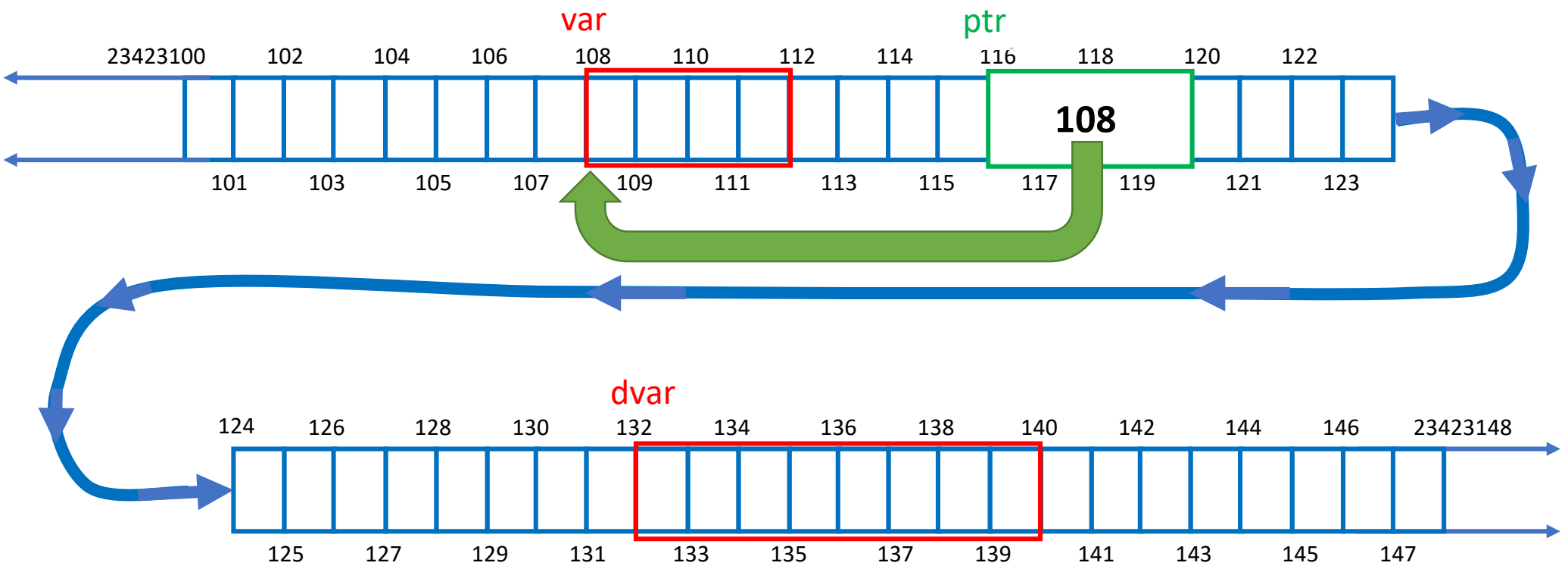

```
int var;
double dvar;
Pointer ptr;
ptr = AddressOf dvar;
```

```
TargetOf ptr
```

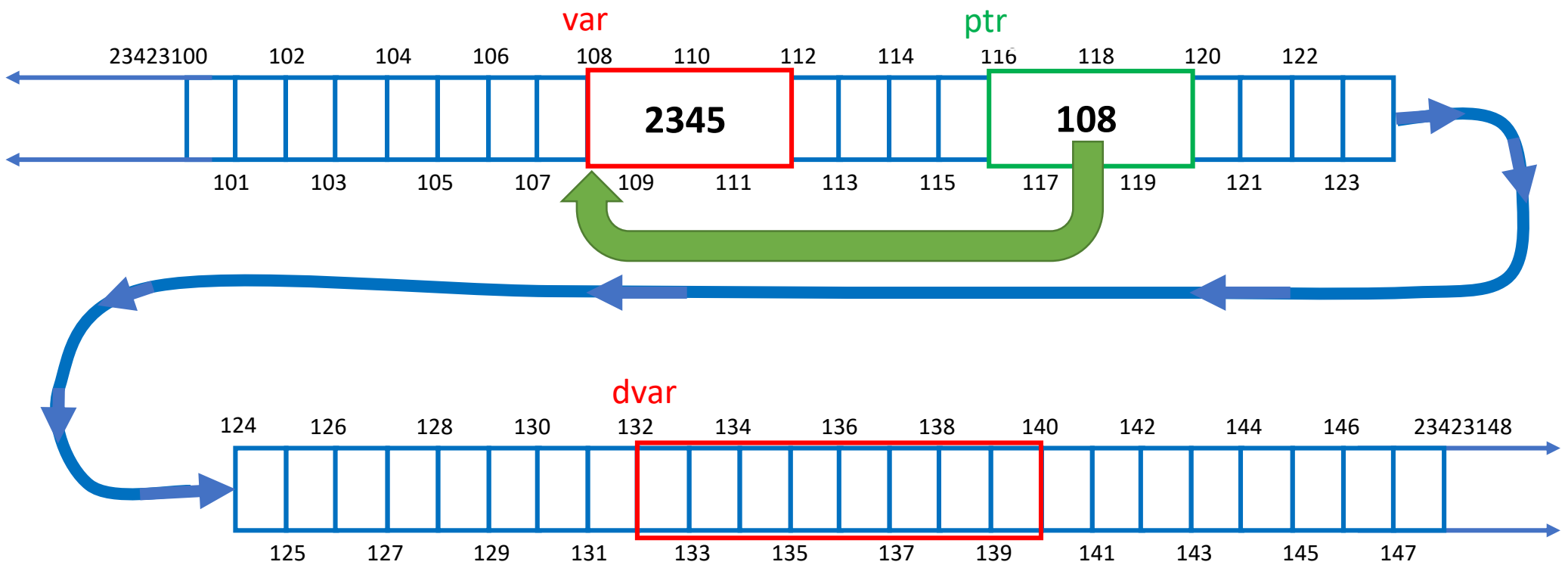
OUTPUT:




```
int var;
double dvar;
int Pointer ptr;
```

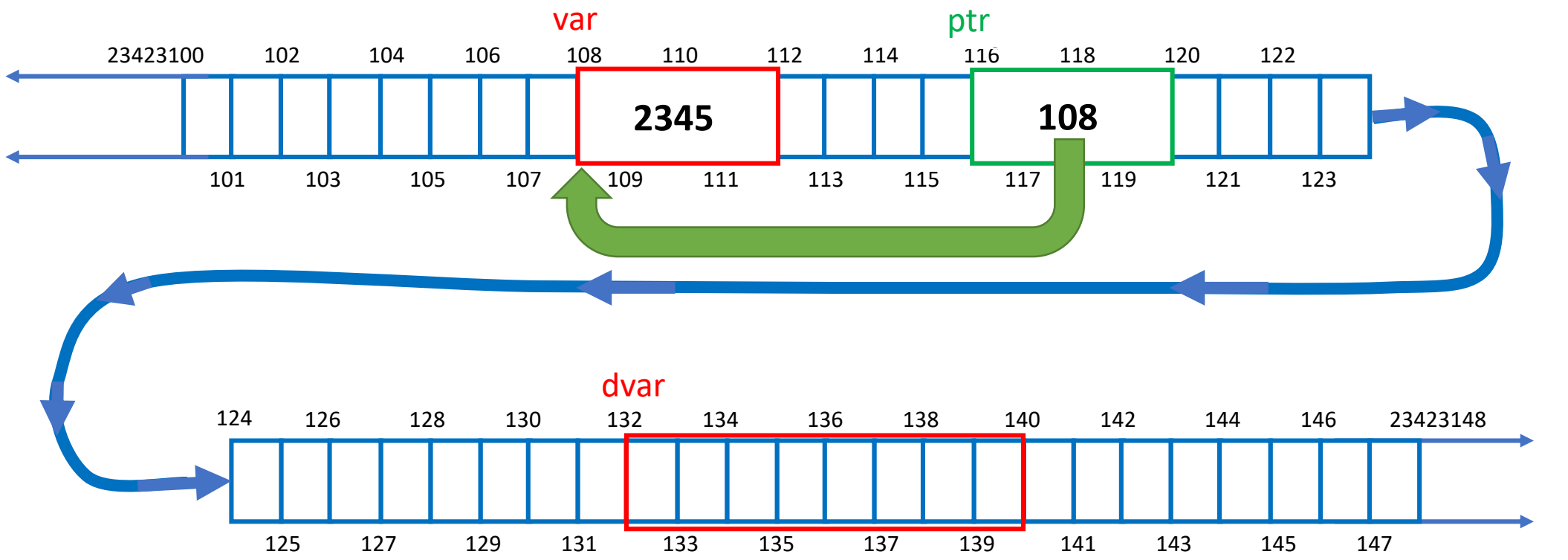



```
int var;
double dvar;
int Pointer ptr;
ptr = AddressOf var;
```




```
int var;
double dvar;
int Pointer ptr;
ptr = AddressOf var;
```

```
TargetOf ptr = 2345;
```




```
int var;
double dvar;
int Pointer ptr;
ptr = AddressOf var;
```

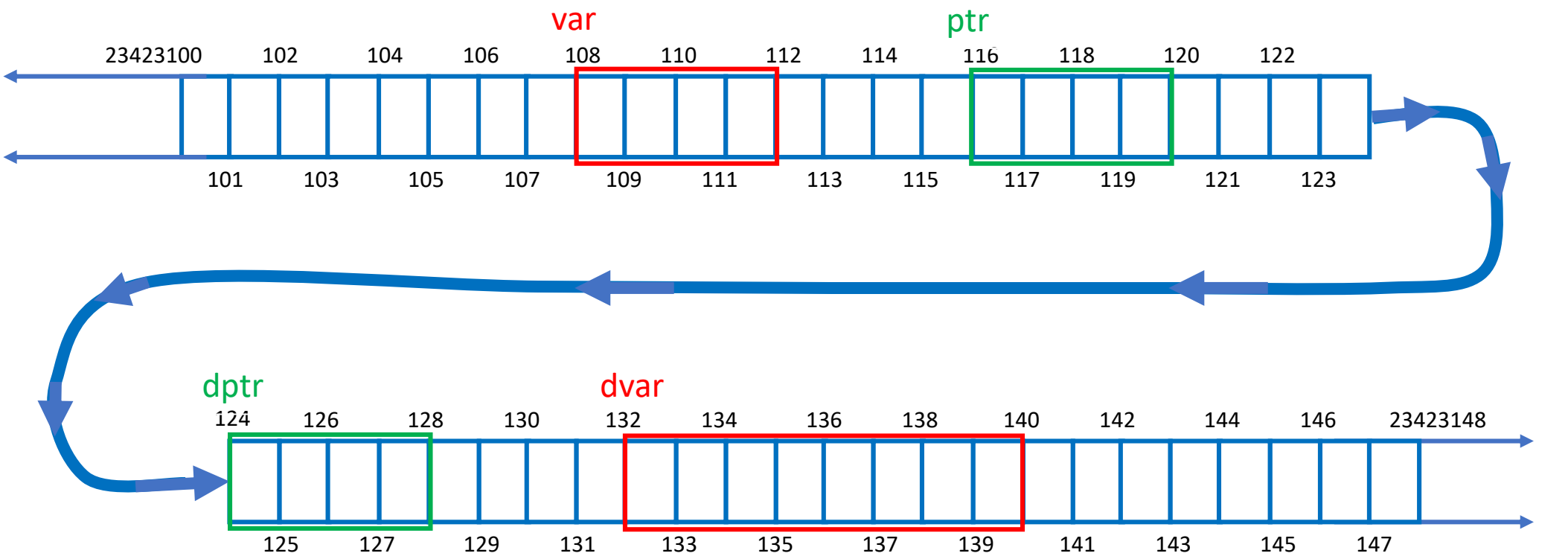
```
TargetOf ptr = 2345;
printf("%d\n", var);
printf("%d\n", TargetOf ptr );
printf("%u\n", ptr );
```

OUTPUT:

2345

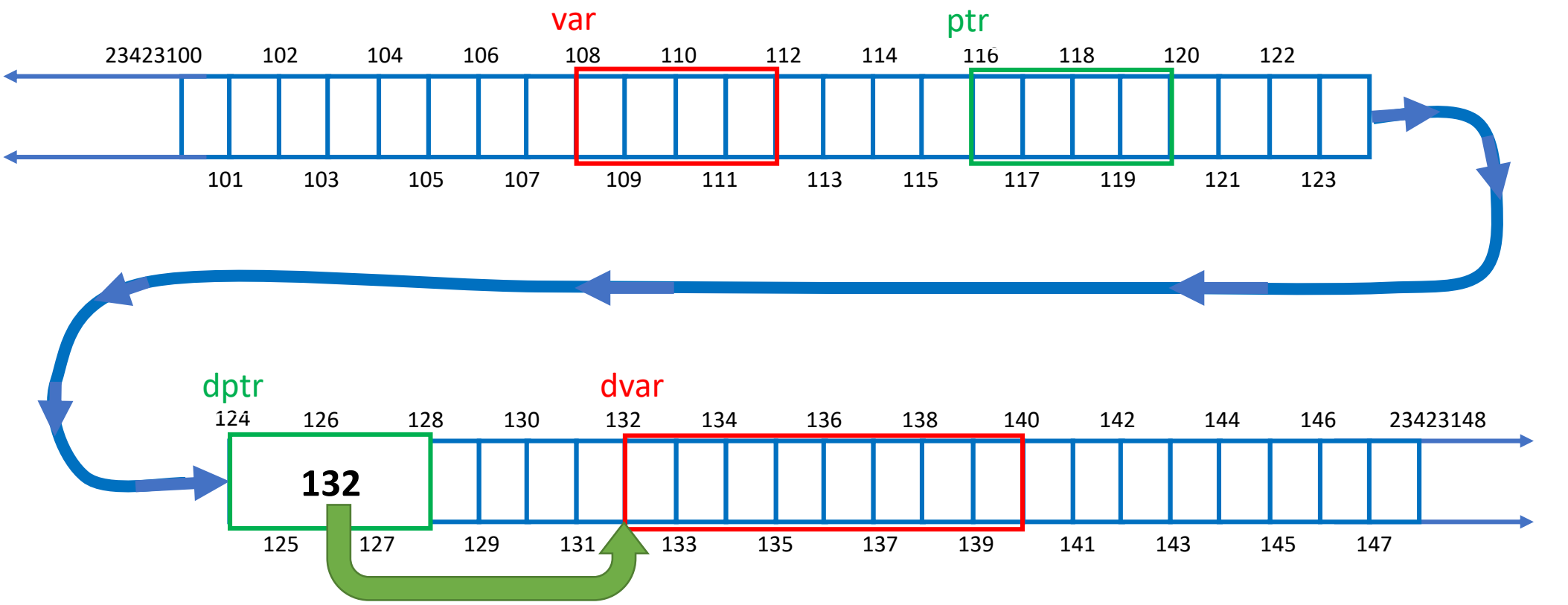
2345

23423**108**



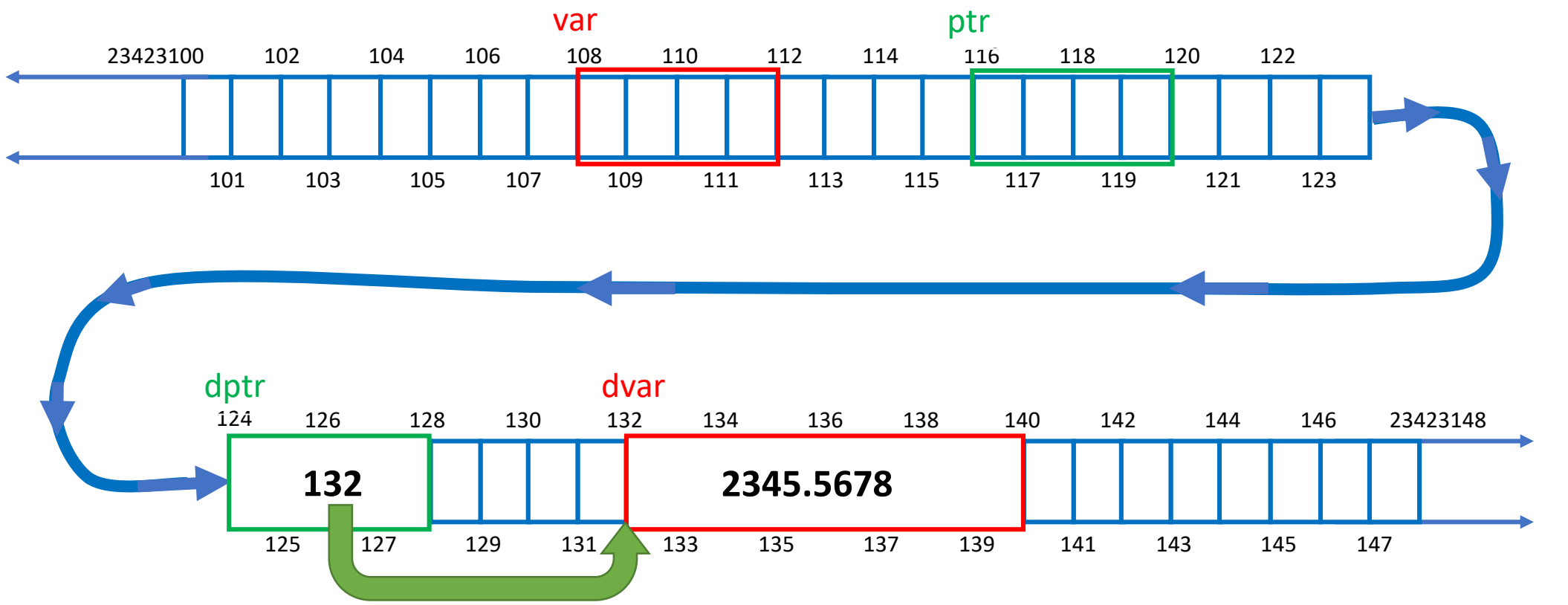

```
int var;
double dvar;
int Pointer ptr;
double Pointer dptr;
```

OUTPUT:




```
int var;
double dvar;
int Pointer ptr;
double Pointer dptr;
dptr = AddressOf dvar;
```

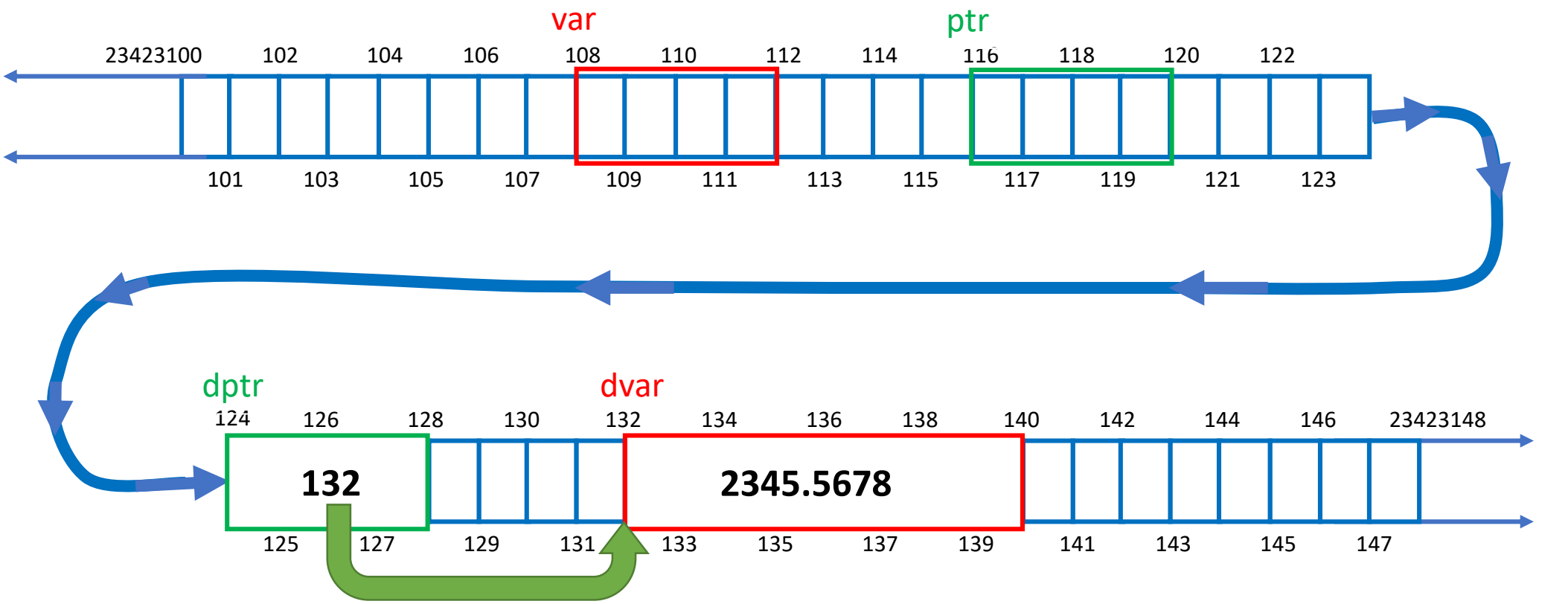
OUTPUT:




```
int var;
double dvar;
int Pointer ptr;
double Pointer dptr;
dptr = AddressOf dvar;
```

```
TargetOf dptr = 2345.5678;
```

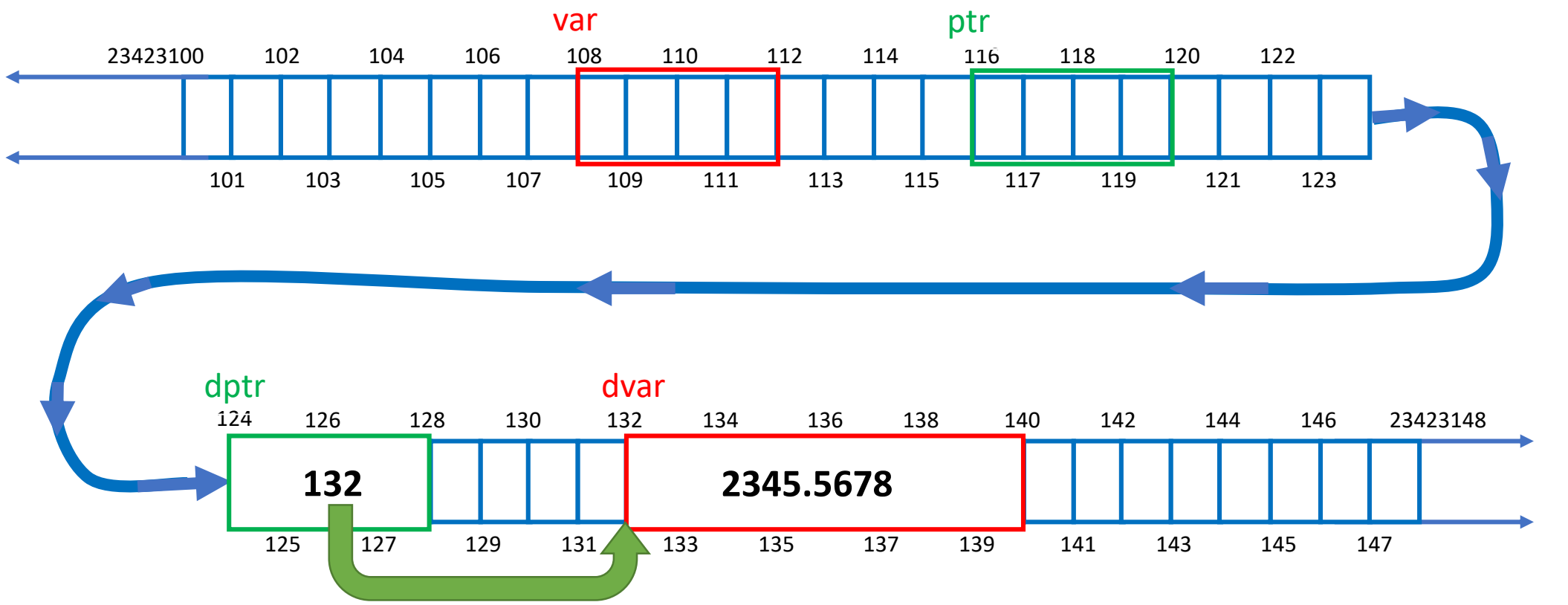
OUTPUT:




```
int var;
double dvar;
int Pointer ptr;
double Pointer dptr;
dptr = AddressOf dvar;
```

```
TargetOf dptr = 2345.5678;
printf("%.2lf\n", dvar);
```

OUTPUT:
2345.57

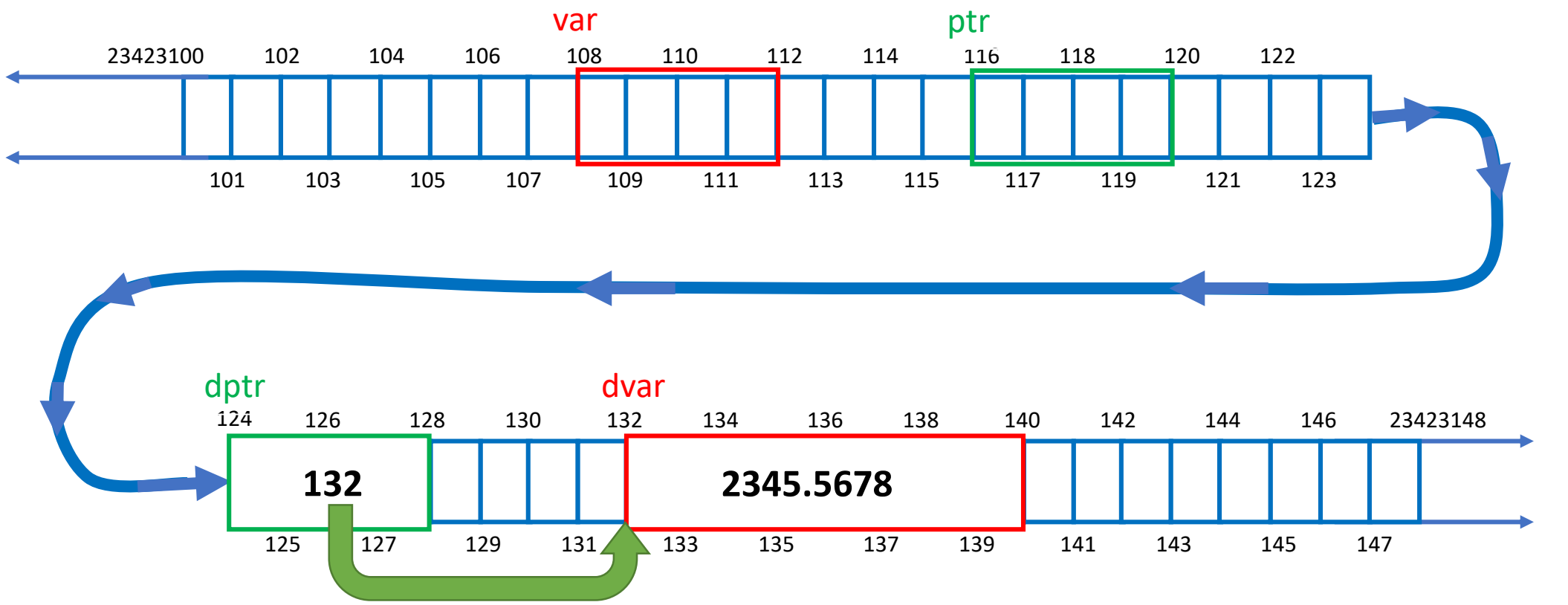



```
int var;
double dvar;
int Pointer ptr;
double Pointer dptr;
dptr = AddressOf dvar;
```

```
TargetOf dptr = 2345.5678;
printf("%.2lf\n", dvar);
printf("% ("%.2lf \n",
                TargetOf dptr );
```

OUTPUT:

```
2345.57
2345.57
```




```
int var;
double dvar;
int Pointer ptr;
double Pointer dptr;
dptr = AddressOf dvar;
```

```
TargetOf dptr = 2345.5678;
printf("%.2lf\n", dvar);
printf("% ("%.2lf \n",
                TargetOf dptr );
printf("%u\n", dptr );
```

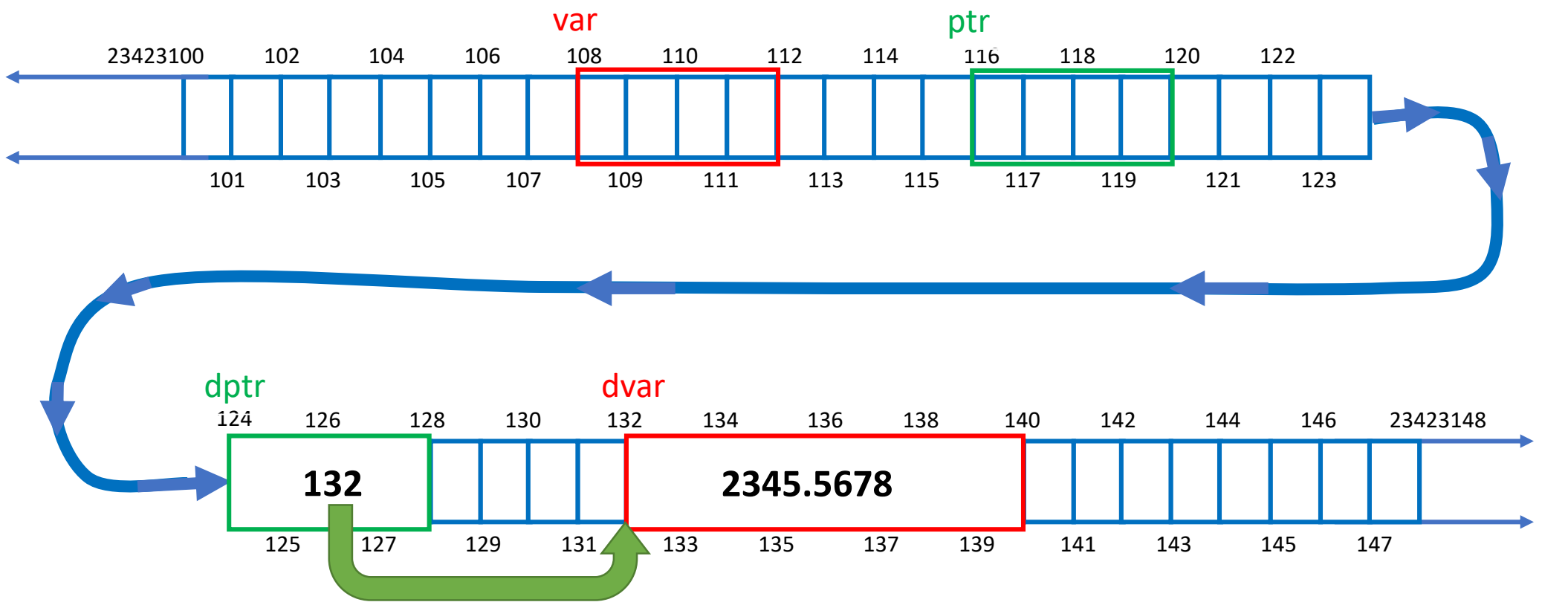
OUTPUT:

2345.57

2345.57

23423**132**

“Pointer” in C is presented by *




```
int var;
double dvar;
int Pointer ptr;
double Pointer dptr;
dptr = AddressOf dvar;
```

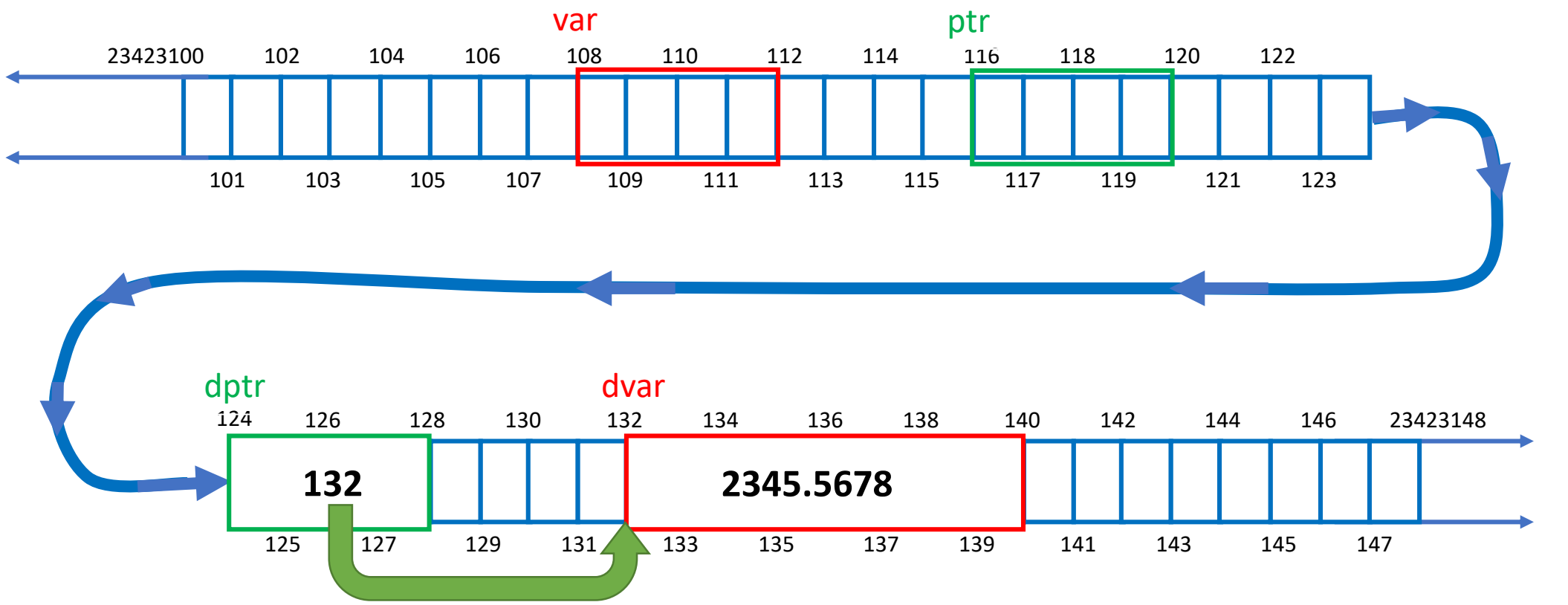
```
TargetOf dptr = 2345.5678;
printf("%.2lf\n", dvar);
printf("% ("%.2lf \n",
                TargetOf dptr );
printf("%u\n", dptr );
```

OUTPUT:

2345.57

2345.57

23423**132**




```
int var;
double dvar;
int * ptr;
double Pointer dptr;
dptr = AddressOf dvar;
```

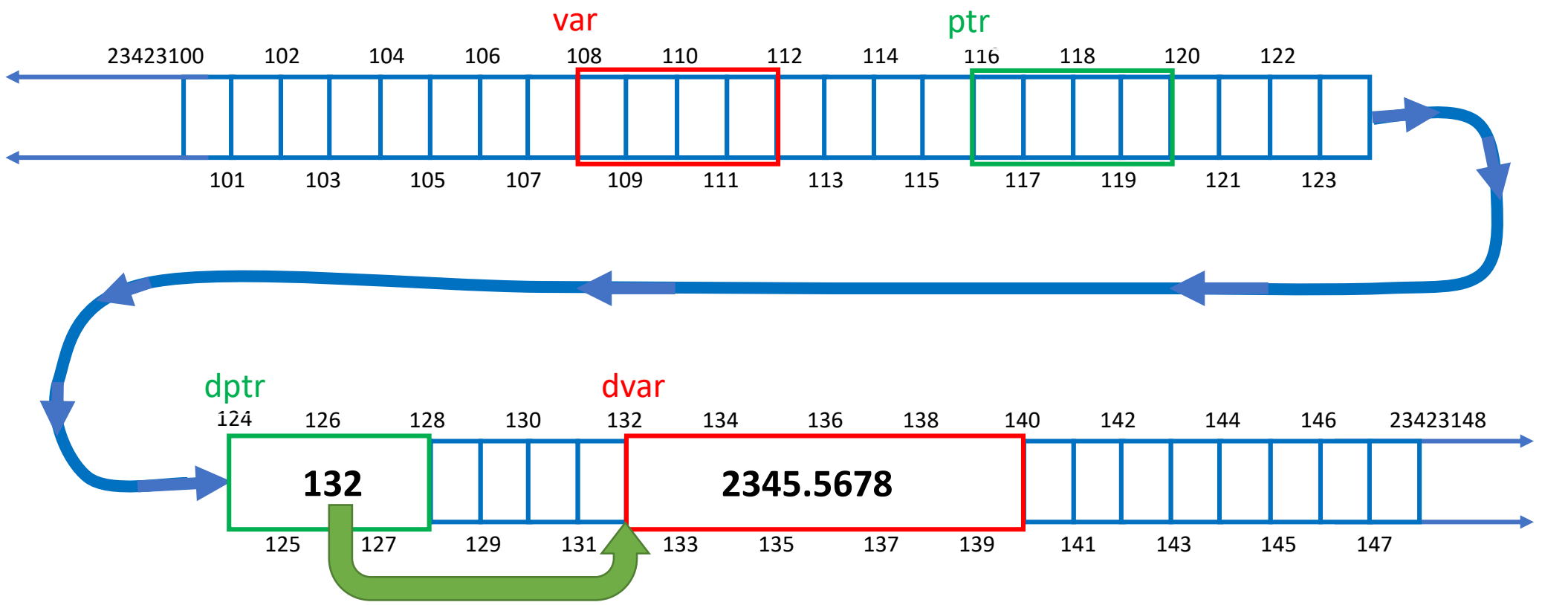
```
TargetOf dptr = 2345.5678;
printf("%.2lf\n", dvar);
printf("% ("%.2lf \n",
                TargetOf dptr );
printf("%u\n", dptr );
```

OUTPUT:

2345.57

2345.57

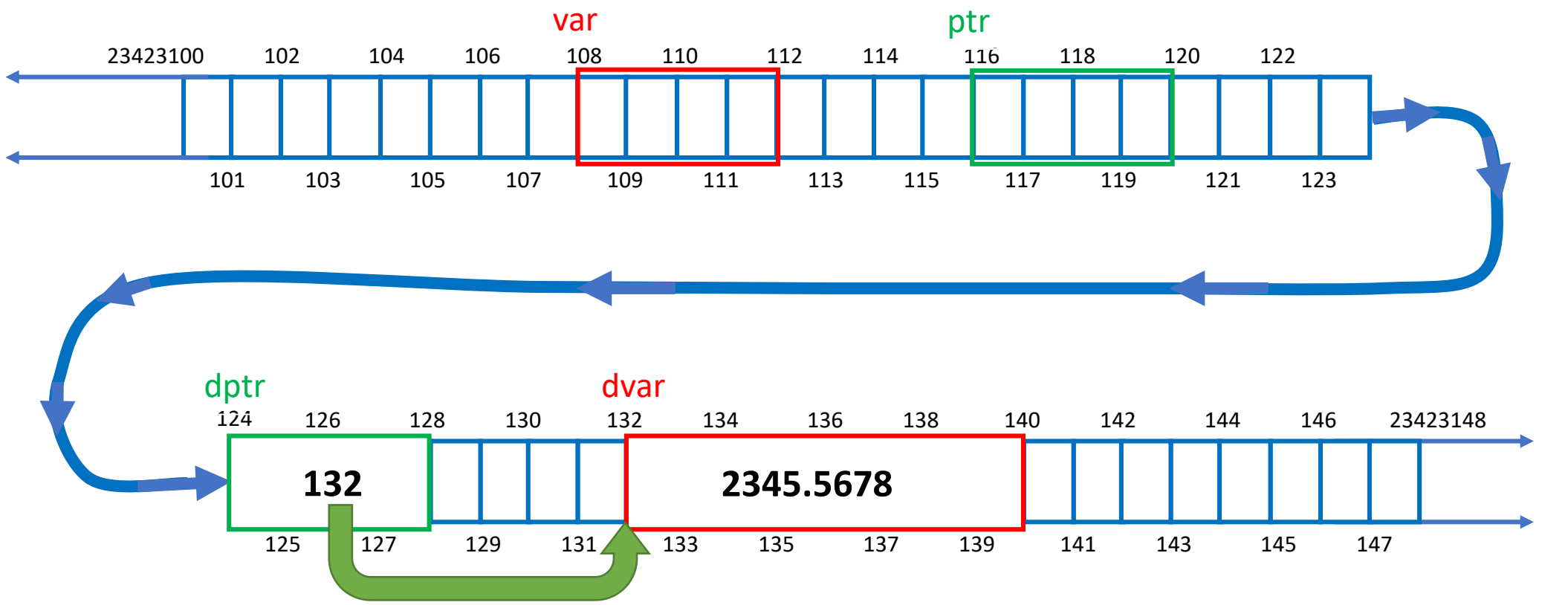
23423**132**




```
int var;
double dvar;
int * ptr;
double * dptr;
dptr = AddressOf dvar;
```

```
TargetOf dptr = 2345.5678;
printf("%.2lf\n", dvar);
printf("% ("%.2lf \n",
                TargetOf dptr );
printf("%u\n", dptr );
```

OUTPUT:
2345.57
2345.57
23423**132**




```
int var;
double dvar;
int* ptr;
double* dptr;
dptr = AddressOf dvar;
```

```
TargetOf dptr = 2345.5678;
printf("%.2lf\n", dvar);
printf("% ("%.2lf \n",
                TargetOf dptr );
printf("%u\n", dptr );
```

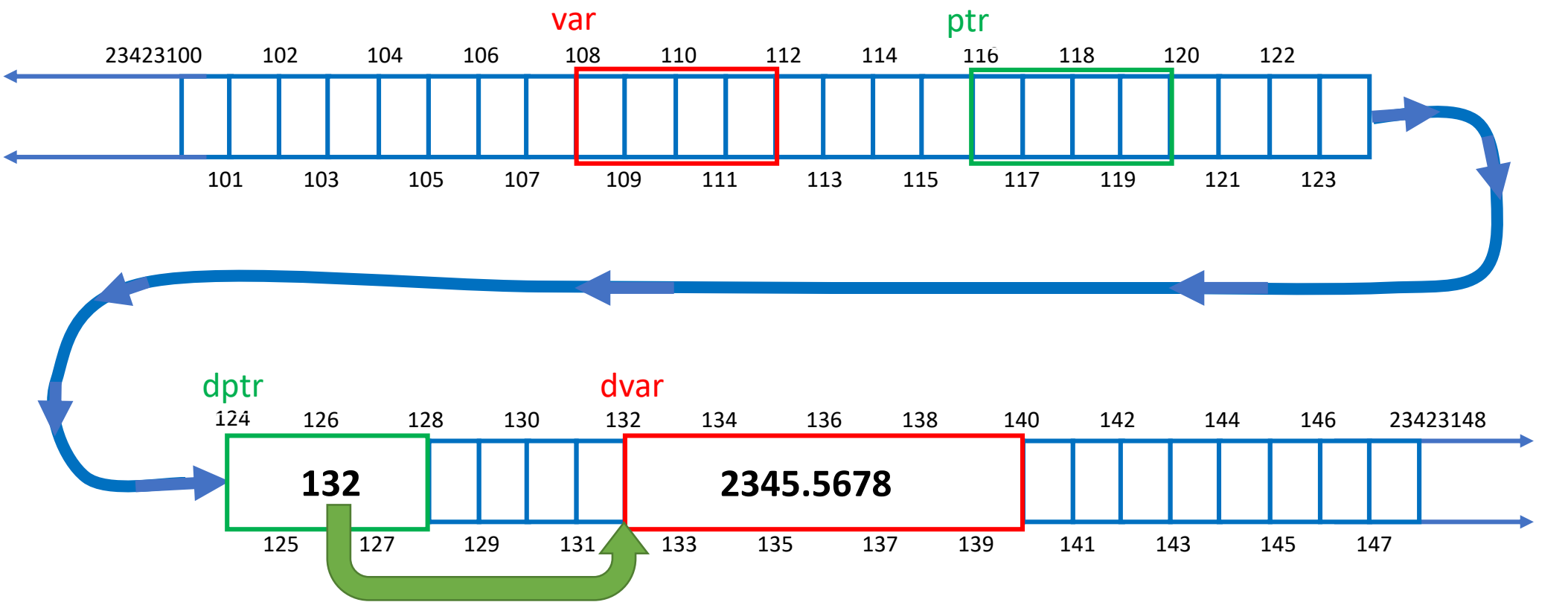
OUTPUT:

2345.57

2345.57

23423**132**

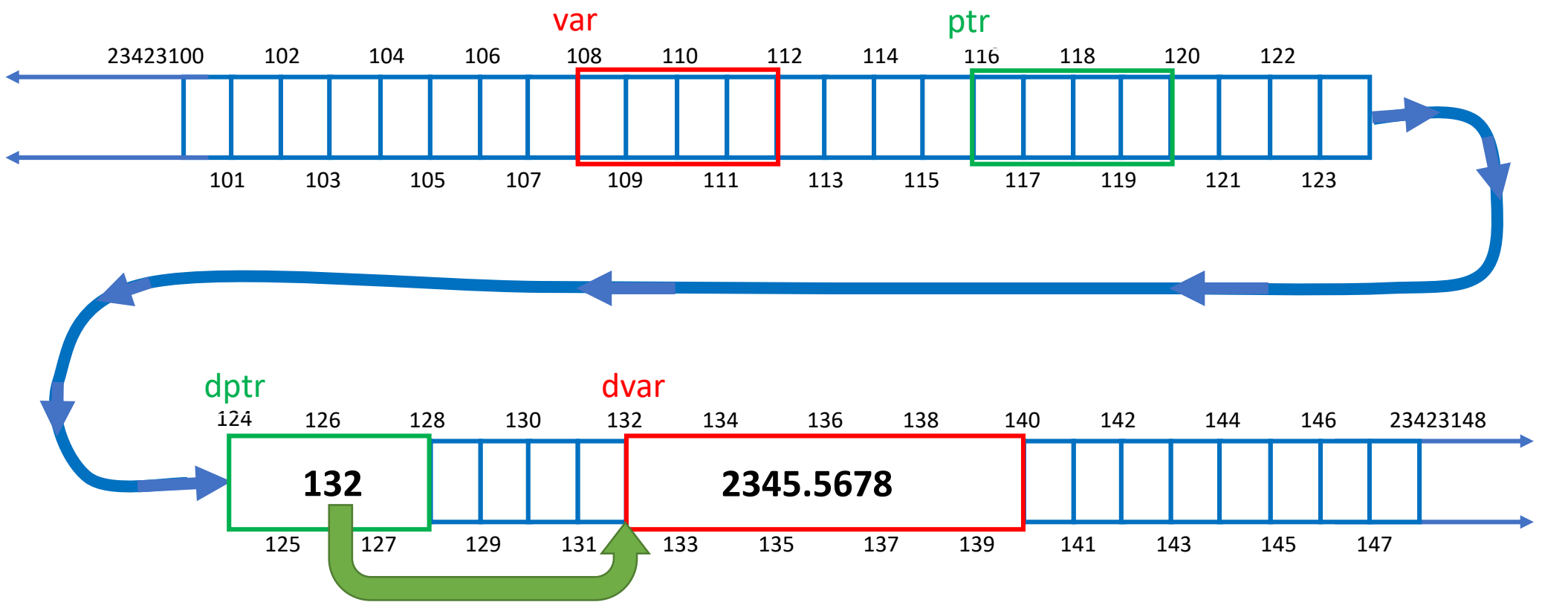
“AddressOf” in C is presented by &




```
int var;
double dvar;
int* ptr;
double* dptr;
dptr = & dvar;
```

```
TargetOf dptr = 2345.5678;
printf("%.2lf\n", dvar);
printf("% ("%.2lf \n",
                TargetOf dptr );
printf("%u\n", dptr );
```

OUTPUT:
2345.57
2345.57
23423**132**

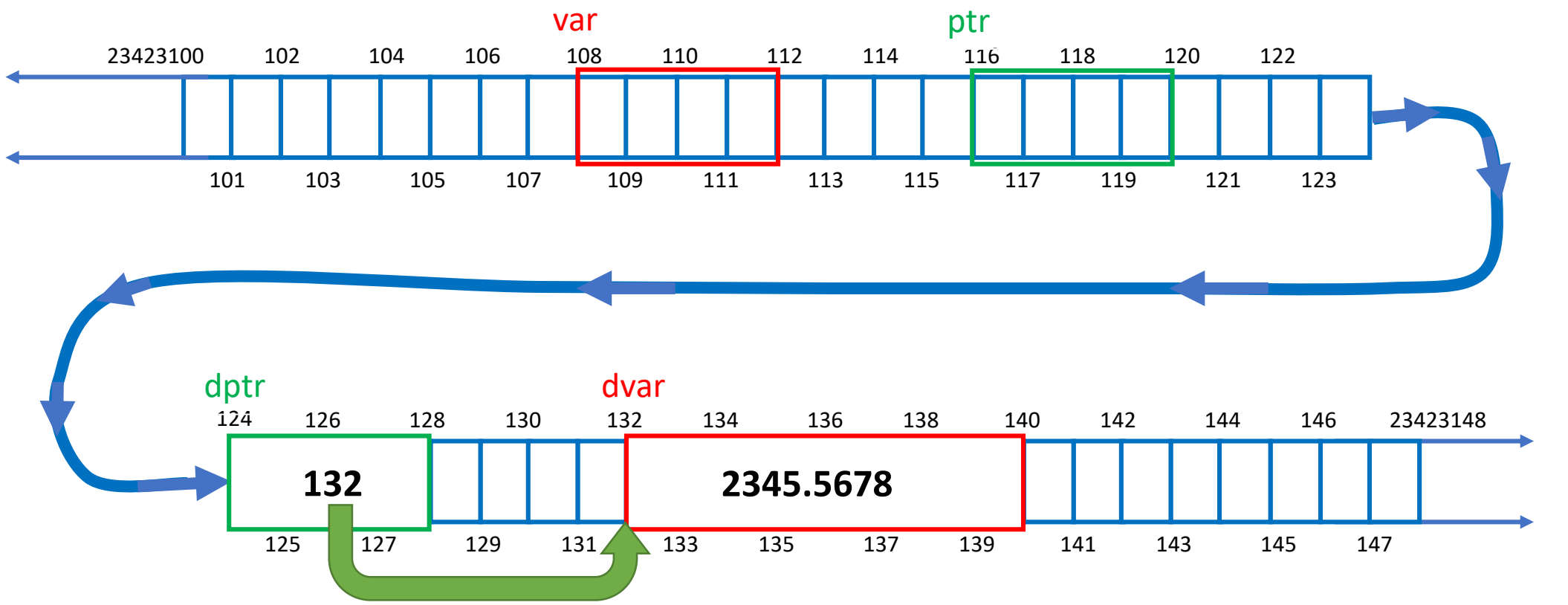



```
int var;
double dvar;
int* ptr;
double* dptr;
dptr = &dvar;
```

```
TargetOf dptr = 2345.5678;
printf("%.2lf\n", dvar);
printf("% ("%.2lf \n",
                TargetOf dptr );
printf("%u\n", dptr );
```

OUTPUT:
2345.57
2345.57
23423**132**

“TargetOf” in C is presented by *




```
int var;
double dvar;
int* ptr;
double* dptr;
dptr = &dvar;
```

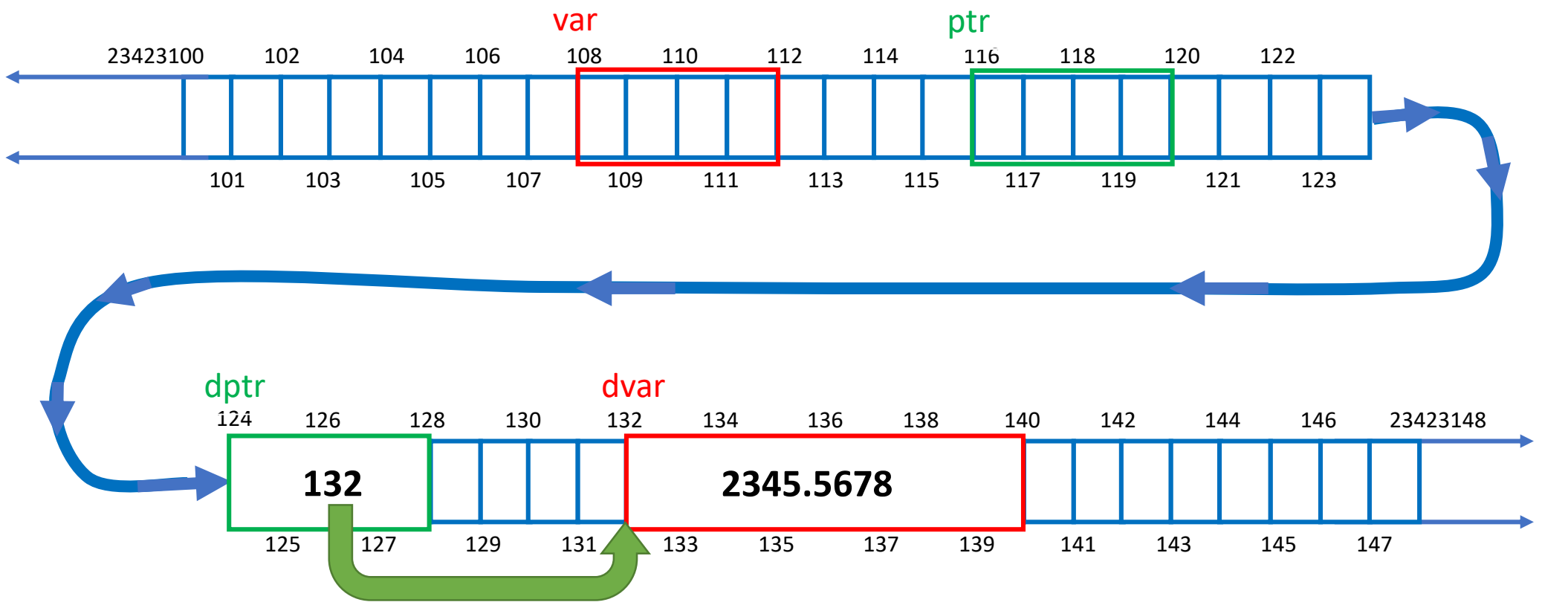
```
* dptr = 2345.5678;
printf("%.2lf\n", dvar);
printf("% ("%.2lf \n",
                TargetOf dptr );
printf("%u\n", dptr );
```

OUTPUT:

2345.57

2345.57

23423**132**

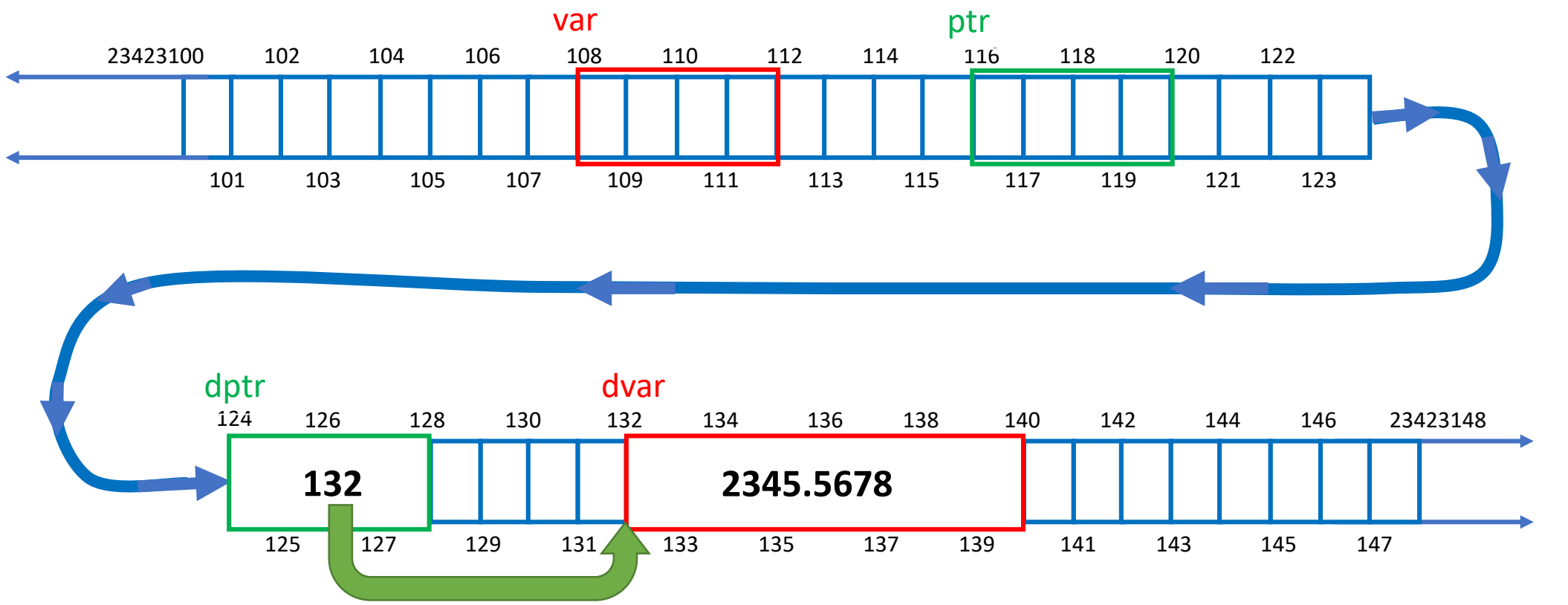



```
int var;
double dvar;
int* ptr;
double* dptr;
dptr = &dvar;
```

```
* dptr = 2345.5678;
printf("%.2lf\n", dvar);
printf("% ("%.2lf \n",* dptr );
printf("%u\n", dptr );
```

OUTPUT:

```
2345.57
2345.57
23423132
```




```
int var;
double dvar;
int* ptr;
double* dptr;
dptr = &dvar;
```

```
*dptr = 2345.5678;
printf("%.2lf\n", dvar);
printf("% ("%.2lf \n",*dptr );
printf("%u\n", dptr );
```

OUTPUT:

2345.57

2345.57

23423**132**

“TargetOf” and “Pointer” are both presented by * ???

“TargetOf” and “Pointer” are both presented by *

If * comes after a type, it means “type pointer”:

“TargetOf” and “Pointer” are both presented by *

If * comes after a type, it means “type pointer”:

```
int* ptr;  // integer pointer ptr
```

“TargetOf” and “Pointer” are both presented by *

If * comes after a type, it means “type pointer”:

```
int* ptr;  // integer pointer ptr
double* dptr; // double pointer dptr
```

“TargetOf” and “Pointer” are both presented by *

If * comes after a type, it means “type pointer”:

```
int* ptr;  // integer pointer ptr
double* dptr; // double pointer dptr
struct Employee* eptr; // Employee pointer eptr
```

“TargetOf” and “Pointer” are both presented by *

If * comes after a type, it means “type pointer”:

```
int* ptr;  // integer pointer ptr
double* dptr; // double pointer dptr
struct Employee* eptr; // Employee pointer eptr
```

If * comes in front of a variable as a unary operator, it means “Target of”:

“TargetOf” and “Pointer” are both presented by *

If * comes after a type, it means “type pointer”:

```
int* ptr;  // integer pointer ptr
double* dptr; // double pointer dptr
struct Employee* eptr; // Employee pointer eptr
```

If * comes in front of a variable as a unary operator, it means “Target of”:

```
a = *p; // a is set to target of p; (p is a pointer, a is a variable)
```

“TargetOf” and “Pointer” are both presented by *

If * comes after a type, it means “type pointer”:

```
int* ptr;  // integer pointer ptr
double* dptr; // double pointer dptr
struct Employee* eptr; // Employee pointer eptr
```

If * comes in front of a variable as a unary operator, it means “Target of”:

```
a = *p; // a is set to target of p; (p is a pointer, a is a variable)
*t = x; // target of t is set to x; (t is a pointer, x is a variable)
```

“TargetOf” and “Pointer” are both presented by *

If * comes after a type, it means “type pointer”:

```
int* ptr;  // integer pointer ptr
double* dptr; // double pointer dptr
struct Employee* eptr; // Employee pointer eptr
```

If * comes in front of a variable as a unary operator, it means “Target of”:

```
a = *p; // a is set to target of p; (p is a pointer, a is a variable)
*t = x; // target of t is set to x;
A = B * *C;     // A is set to B multiply by target of C
                // A is B are variables but C is a pointer
```

“TargetOf” and “Pointer” are both presented by *

If * comes after a type, it means “type pointer”:

```
int* ptr;  // integer pointer ptr
double* dptr; // double pointer dptr
struct Employee* eptr; // Employee pointer eptr
```

If * comes in front of a variable as a unary operator, it means “Target of”:

```
a = *p; // a is set to target of p; (p is a pointer, a is a variable)
*t = x; // target of t is set to x;
A = B * *C;      // A is set to B multiply by target of C
                 // A is B are variables but C is a pointer
E = *M * C * C; // E is set to target of M multiply by C multiply by C
                 // E and C are variables but M is a pointer
```