Chapter 1

Basic Concepts

Objectives

Upon completion you will be able to:

- Use pseudocode in the development of algorithms
- Understand the need for Abstract Data Type (ADT)
- Understand the implementation of ADTs
- Use *void* pointers and pointer to functions
- Understand the role of Big-O notation

CS 1037
Computer Science Fundamentals II

Part Seven:
Basic Concepts

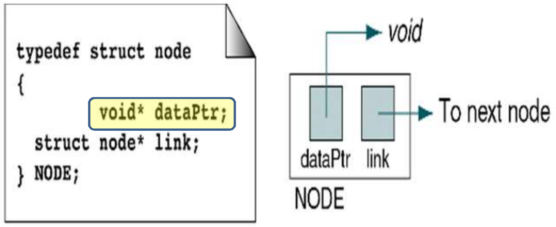

```
typedef struct node
{
        void* dataPtr;
  struct node* link;
} NODE;
```

FIGURE 1-8 Pointer to Node

```
typedef struct node
{
         void*           dataPtr;
         struct node*    link;
} NODE;

//      =================== createNode ====================

NODE* createNode (void* itemPtr)
{
        NODE* nodePtr;
        nodePtr = (NODE*) malloc (sizeof (NODE));
        nodePtr->dataPtr = itemPtr;
        nodePtr->link    = NULL;
        return nodePtr;
}       // createNode
```

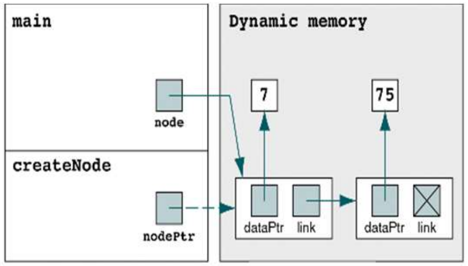
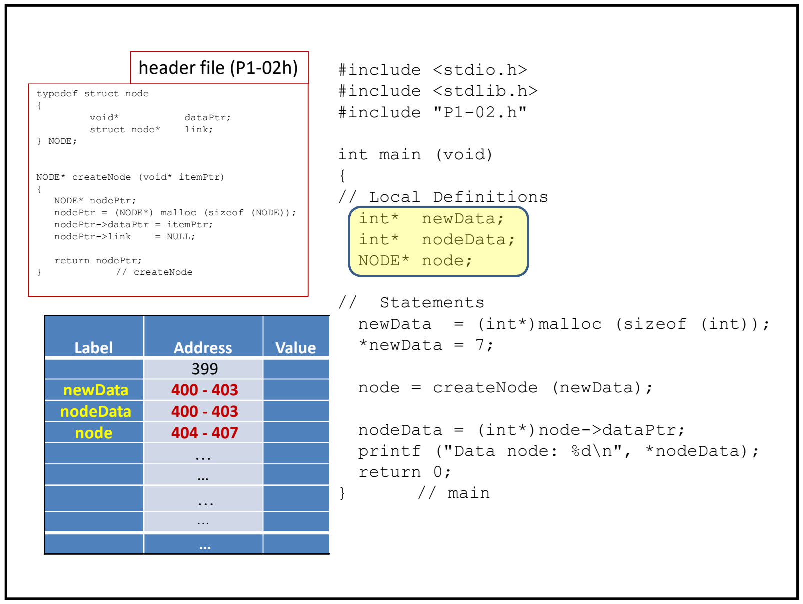


FIGURE 1-10 Structure for Two Linked Nodes

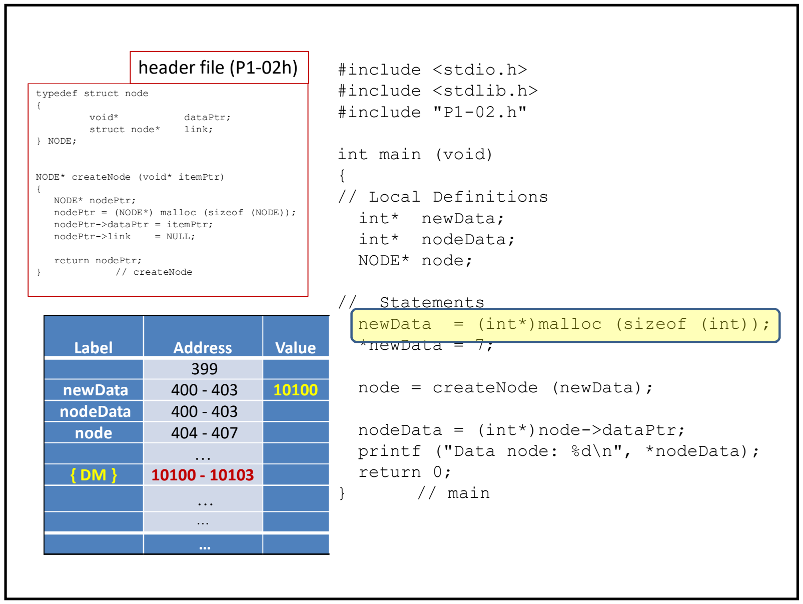


header file (P1-02h)

```
typedef struct node
{
        void*           dataPtr;
        struct node*    link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr = (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;

   return nodePtr;
}            // createNode
```

Label	Address	Value
	399	
newData	400 - 403	
nodeData	400 - 403	
node	404 - 407	
	…	
	…	
	…	
	…	
	…	

```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"

int main (void)
{
// Local Definitions
  int*   newData;
  int*   nodeData;
  NODE* node;

//  Statements
  newData  = (int*)malloc (sizeof (int));
  *newData = 7;

  node = createNode (newData);

  nodeData = (int*)node->dataPtr;
  printf ("Data node: %d\n", *nodeData);
  return 0;
}       // main
```

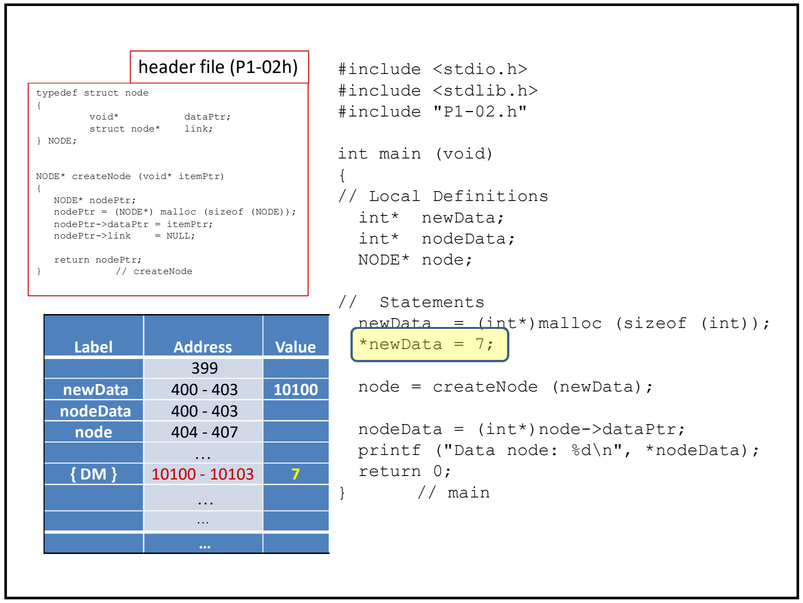


header file (P1-02h)

```
typedef struct node
{
        void*           dataPtr;
        struct node*    link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr = (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;

   return nodePtr;
}            // createNode
```

Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	
node	404 - 407	
	…	
{ DM }	10100 - 10103	
	…	
	…	
	…	

```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"

int main (void)
{
// Local Definitions
  int*   newData;
  int*   nodeData;
  NODE* node;

//  Statements
  newData  = (int*)malloc (sizeof (int));
  *newData = 7;

  node = createNode (newData);

  nodeData = (int*)node->dataPtr;
  printf ("Data node: %d\n", *nodeData);
  return 0;
}       // main
```

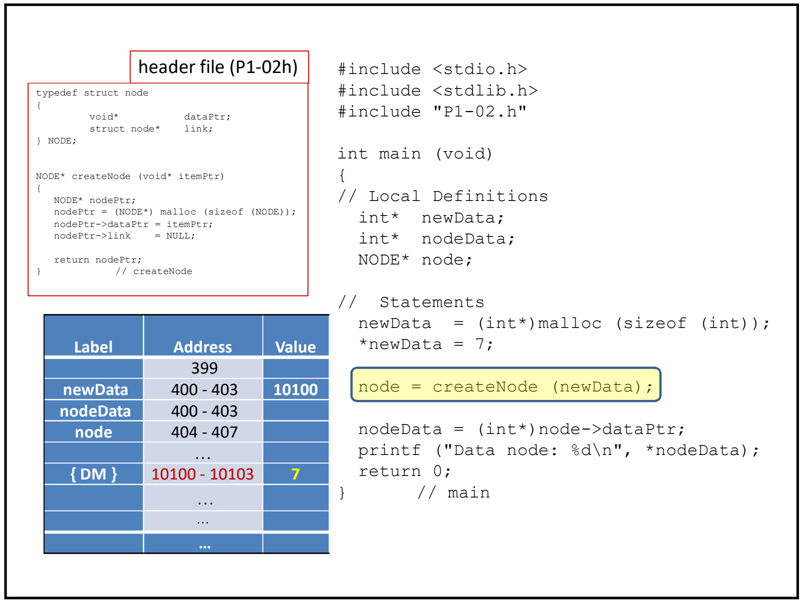


header file (P1-02h)

```
typedef struct node
{
        void*           dataPtr;
        struct node*    link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr = (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;

   return nodePtr;
}            // createNode
```

Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	
node	404 - 407	
	…	
{ DM }	10100 - 10103	7
	…	
	…	
	…	

```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"

int main (void)
{
// Local Definitions
  int*   newData;
  int*   nodeData;
  NODE* node;

//  Statements
  newData  = (int*)malloc (sizeof (int));
  *newData = 7;

  node = createNode (newData);

  nodeData = (int*)node->dataPtr;
  printf ("Data node: %d\n", *nodeData);
  return 0;
}       // main
```

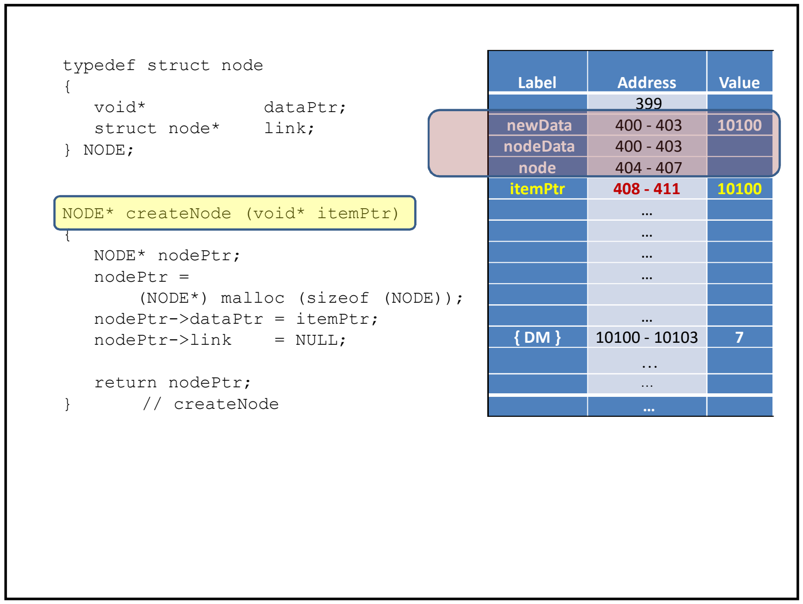


header file (P1-02h)

```
typedef struct node
{
        void*           dataPtr;
        struct node*    link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr = (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;

   return nodePtr;
}            // createNode
```

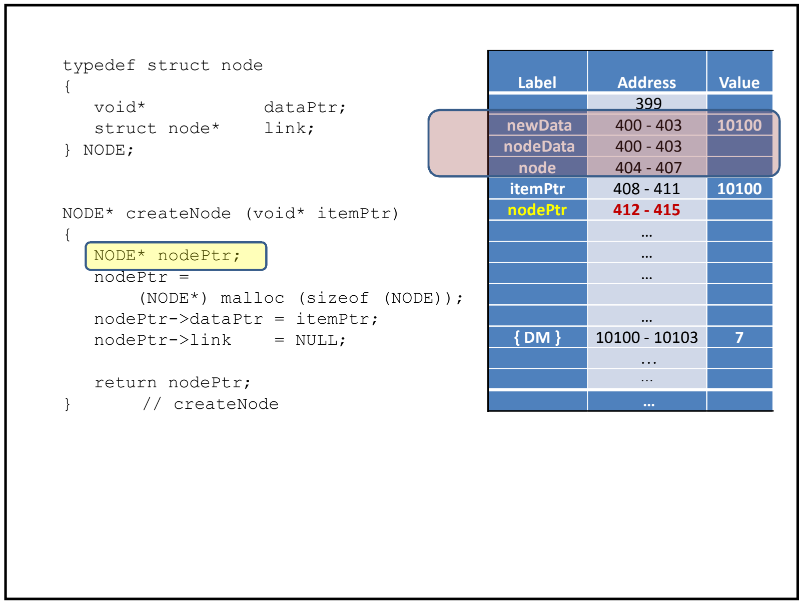
Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	
node	404 - 407	
	…	
{ DM }	10100 - 10103	7
	…	
	…	
	…	

```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"

int main (void)
{
// Local Definitions
  int*   newData;
  int*   nodeData;
  NODE* node;

//  Statements
  newData  = (int*)malloc (sizeof (int));
  *newData = 7;

  node = createNode (newData);

  nodeData = (int*)node->dataPtr;
  printf ("Data node: %d\n", *nodeData);
  return 0;
}       // main
```



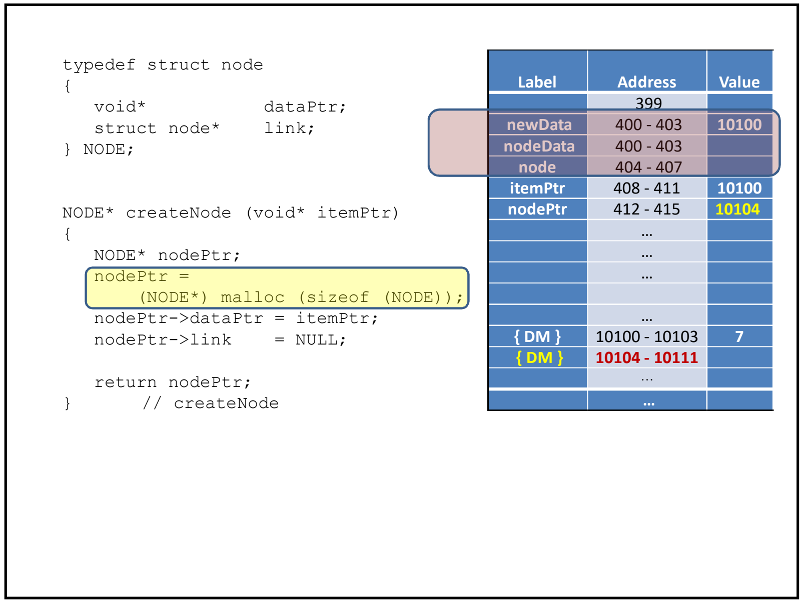
```
typedef struct node
{
   void*          dataPtr;
   struct node*   link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr =
       (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;

   return nodePtr;
}      // createNode
```

Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	
node	404 - 407	
itemPtr	408 - 411	10100
	…	
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
	…	
	…	
	…	



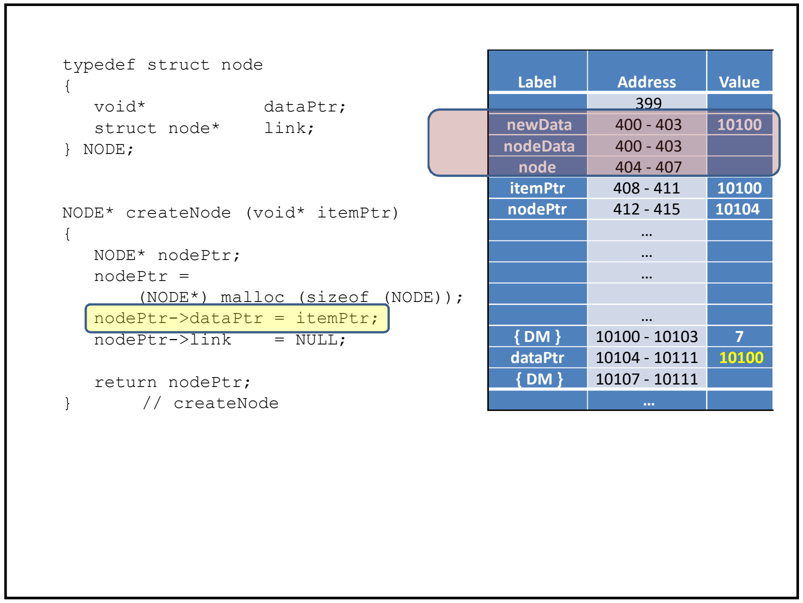
```
typedef struct node
{
   void*          dataPtr;
   struct node*   link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr =
       (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;

   return nodePtr;
}      // createNode
```

Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	
node	404 - 407	
itemPtr	408 - 411	10100
nodePtr	412 - 415	
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
	…	
	…	
	…	



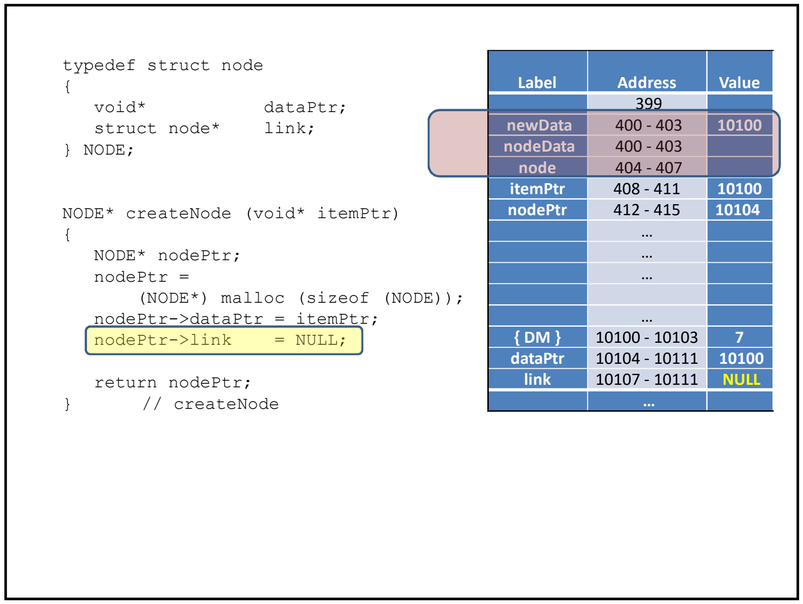
```
typedef struct node
{
   void*          dataPtr;
   struct node*   link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr =
       (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;

   return nodePtr;
}      // createNode
```

Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	
node	404 - 407	
itemPtr	408 - 411	10100
nodePtr	412 - 415	10104
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
{ DM }	10104 - 10111	
	…	
	…	



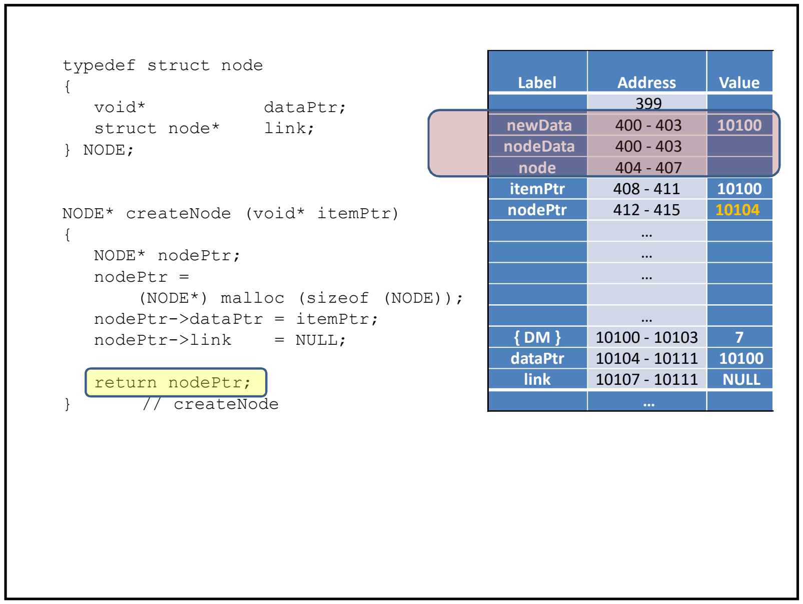
```
typedef struct node
{
   void*          dataPtr;
   struct node*   link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr =
       (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;

   return nodePtr;
}      // createNode
```

Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	
node	404 - 407	
itemPtr	408 - 411	10100
nodePtr	412 - 415	10104
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
{ DM }	10107 - 10111	
	…	



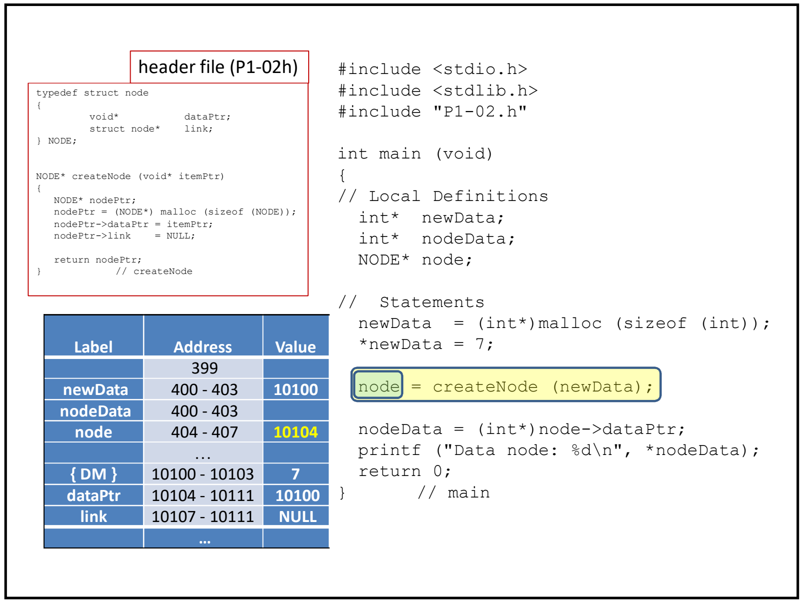
```
typedef struct node
{
   void*          dataPtr;
   struct node*   link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr =
       (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link     = NULL;

   return nodePtr;
}       // createNode
```

Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	
node	404 - 407	
itemPtr	408 - 411	10100
nodePtr	412 - 415	10104
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	NULL
	…	



```
typedef struct node
{
   void*          dataPtr;
   struct node*   link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr =
       (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link     = NULL;

   return nodePtr;
}       // createNode
```

Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	
node	404 - 407	
itemPtr	408 - 411	10100
nodePtr	412 - 415	10104
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	NULL
	…	

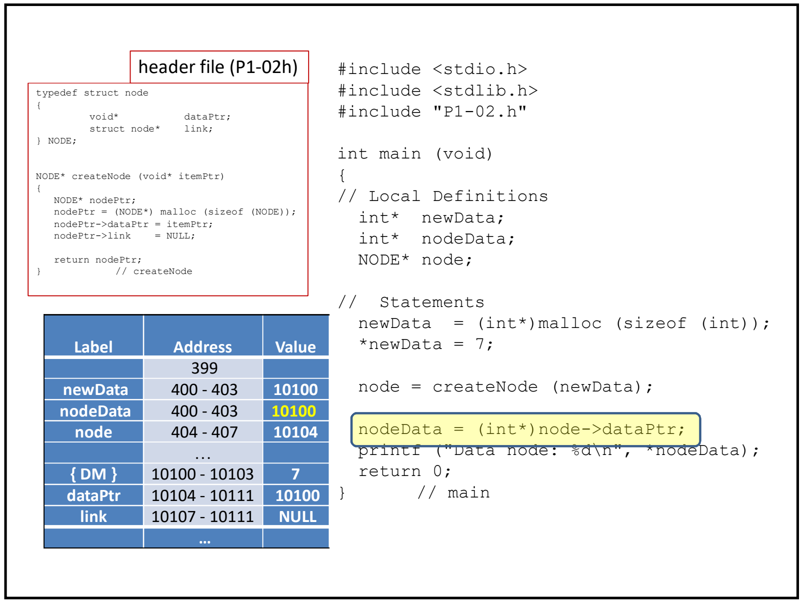


header file (P1-02h)

```
typedef struct node
{
        void*           dataPtr;
        struct node*    link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr = (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;

   return nodePtr;
}           // createNode
```

Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	
node	404 - 407	10104
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	NULL
	…	

```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"

int main (void)
{
// Local Definitions
   int*  newData;
   int*  nodeData;
   NODE* node;

//  Statements
   newData  = (int*)malloc (sizeof (int));
   *newData = 7;

   node = createNode (newData);

   nodeData = (int*)node->dataPtr;
   printf ("Data node: %d\n", *nodeData);
   return 0;
}       // main
```

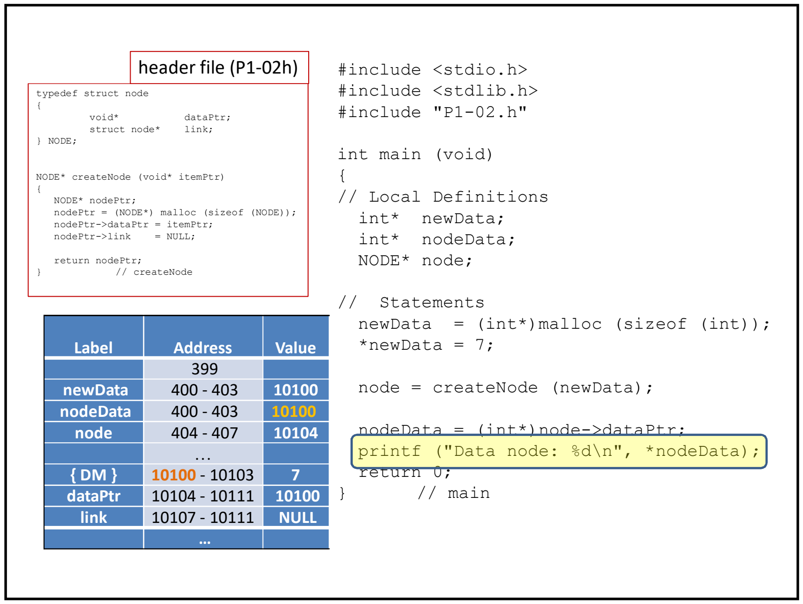


header file (P1-02h)

```
typedef struct node
{
        void*           dataPtr;
        struct node*    link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr = (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;

   return nodePtr;
}           // createNode
```

Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	10100
node	404 - 407	10104
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	NULL
	…	

```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"

int main (void)
{
// Local Definitions
   int*  newData;
   int*  nodeData;
   NODE* node;

//  Statements
   newData  = (int*)malloc (sizeof (int));
   *newData = 7;

   node = createNode (newData);

   nodeData = (int*)node->dataPtr;
   printf ("Data node: %d\n", *nodeData);
   return 0;
}       // main
```

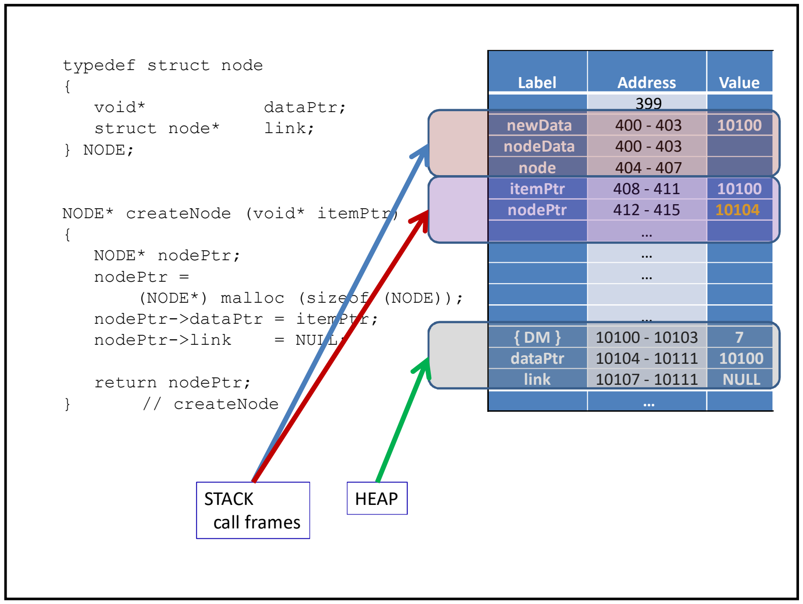


header file (P1-02h)

```
typedef struct node
{
         void*            dataPtr;
         struct node*     link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr = (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;

   return nodePtr;
}            // createNode
```

Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	10100
node	404 - 407	10104
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	NULL
	…	

```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"

int main (void)
{
// Local Definitions
  int*  newData;
  int*  nodeData;
  NODE* node;

//  Statements
  newData  = (int*)malloc (sizeof (int));
  *newData = 7;

  node = createNode (newData);

  nodeData = (int*)node->dataPtr;
  printf ("Data node: %d\n", *nodeData);
  return 0;
}       // main
```

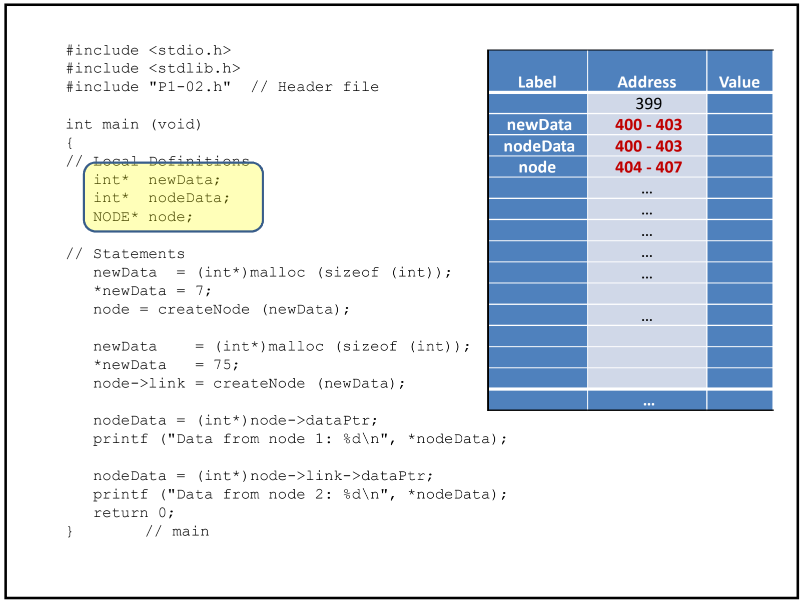


```
typedef struct node
{
   void*           dataPtr;
   struct node*    link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr =
      (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;

   return nodePtr;
}      // createNode
```

Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	
node	404 - 407	
itemPtr	408 - 411	10100
nodePtr	412 - 415	10104
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	NULL
	…	

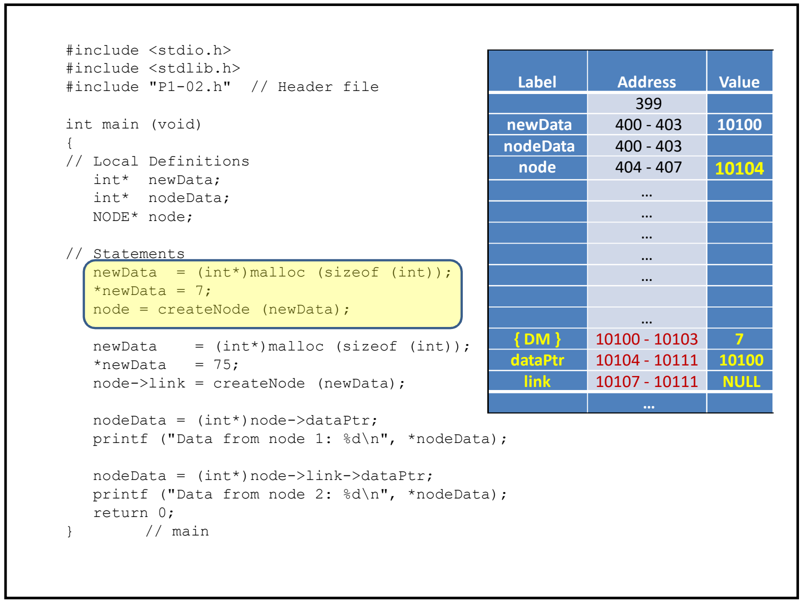
STACK
call frames

HEAP



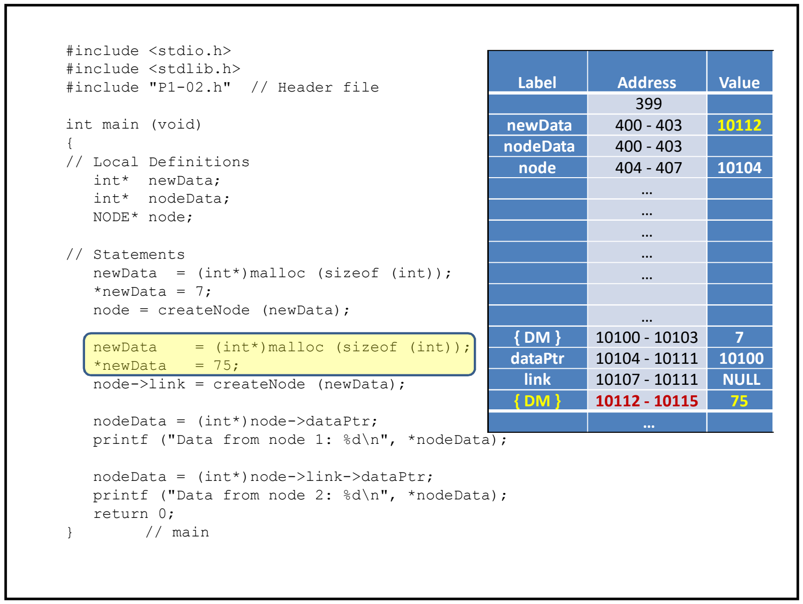
```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"  // Header file

int main (void)
{
// Local Definitions
   int*  newData;
   int*  nodeData;
   NODE* node;

// Statements
   newData  = (int*)malloc (sizeof (int));
   *newData = 7;
   node = createNode (newData);

   newData    = (int*)malloc (sizeof (int));
   *newData   = 75;
   node->link = createNode (newData);

   nodeData = (int*)node->dataPtr;
   printf ("Data from node 1: %d\n", *nodeData);

   nodeData = (int*)node->link->dataPtr;
   printf ("Data from node 2: %d\n", *nodeData);
   return 0;
}        // main
```

Label	Address	Value
	399	
newData	400 - 403	
nodeData	400 - 403	
node	404 - 407	
	…	
	…	
	…	
	…	
	…	
	…	
	…	



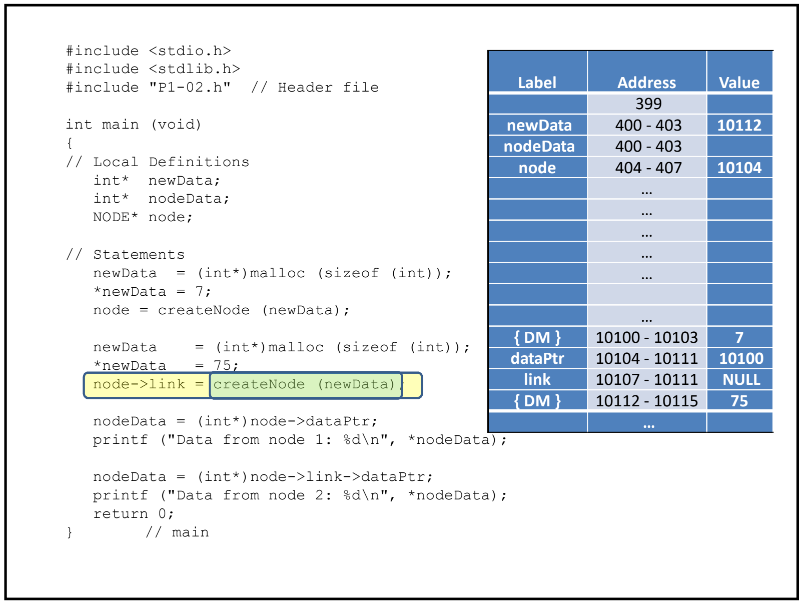
```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"  // Header file

int main (void)
{
// Local Definitions
   int*  newData;
   int*  nodeData;
   NODE* node;

// Statements
   newData  = (int*)malloc (sizeof (int));
   *newData = 7;
   node = createNode (newData);

   newData    = (int*)malloc (sizeof (int));
   *newData   = 75;
   node->link = createNode (newData);

   nodeData = (int*)node->dataPtr;
   printf ("Data from node 1: %d\n", *nodeData);

   nodeData = (int*)node->link->dataPtr;
   printf ("Data from node 2: %d\n", *nodeData);
   return 0;
}        // main
```

Label	Address	Value
	399	
newData	400 - 403	10100
nodeData	400 - 403	
node	404 - 407	10104
	…	
	…	
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	NULL
	…	



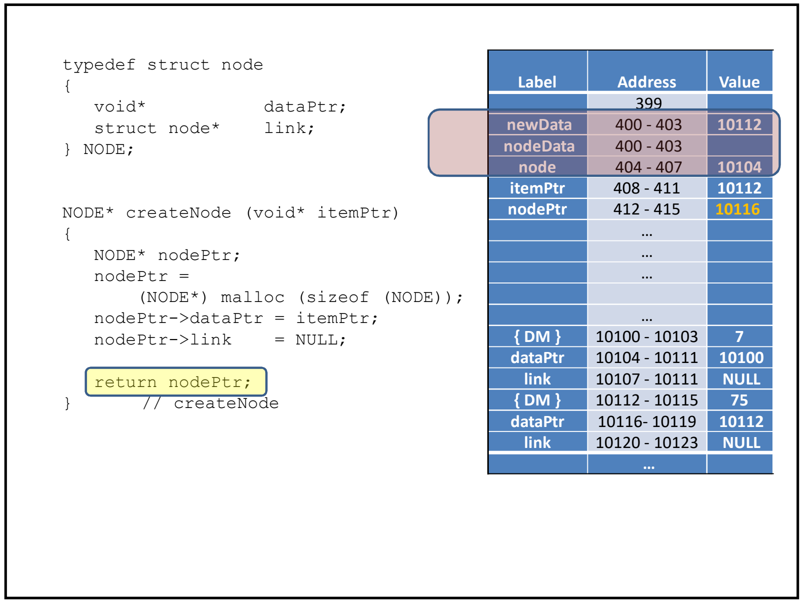
```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"  // Header file

int main (void)
{
// Local Definitions
   int*  newData;
   int*  nodeData;
   NODE* node;

// Statements
   newData  = (int*)malloc (sizeof (int));
   *newData = 7;
   node = createNode (newData);

   newData    = (int*)malloc (sizeof (int));
   *newData   = 75;
   node->link = createNode (newData);

   nodeData = (int*)node->dataPtr;
   printf ("Data from node 1: %d\n", *nodeData);

   nodeData = (int*)node->link->dataPtr;
   printf ("Data from node 2: %d\n", *nodeData);
   return 0;
}        // main
```

Label	Address	Value
	399	
newData	400 - 403	10112
nodeData	400 - 403	
node	404 - 407	10104
	…	
	…	
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	NULL
{ DM }	10112 - 10115	75
	…	



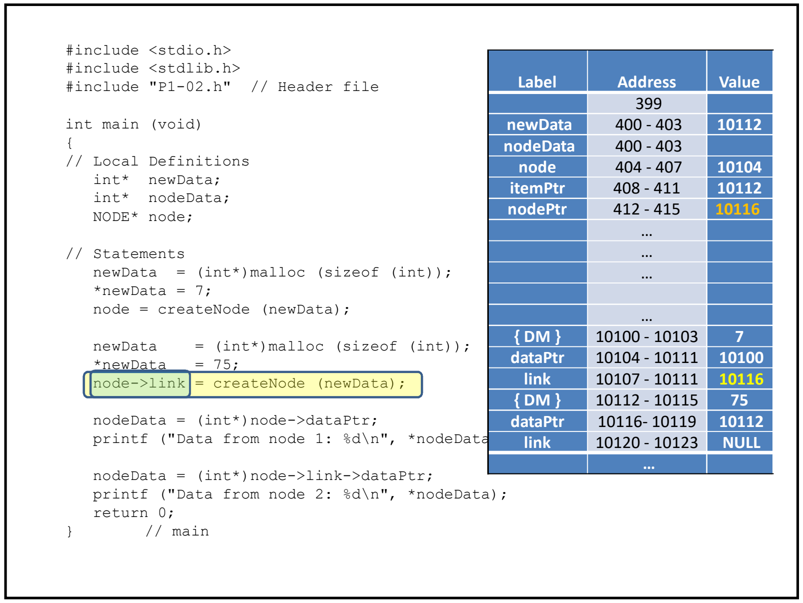
```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"  // Header file

int main (void)
{
// Local Definitions
   int*  newData;
   int*  nodeData;
   NODE* node;

// Statements
   newData  = (int*)malloc (sizeof (int));
   *newData = 7;
   node = createNode (newData);

   newData    = (int*)malloc (sizeof (int));
   *newData   = 75;
   node->link = createNode (newData);

   nodeData = (int*)node->dataPtr;
   printf ("Data from node 1: %d\n", *nodeData);

   nodeData = (int*)node->link->dataPtr;
   printf ("Data from node 2: %d\n", *nodeData);
   return 0;
}        // main
```

Label	Address	Value
	399	
newData	400 - 403	10112
nodeData	400 - 403	
node	404 - 407	10104
	…	
	…	
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	NULL
{ DM }	10112 - 10115	75
	…	



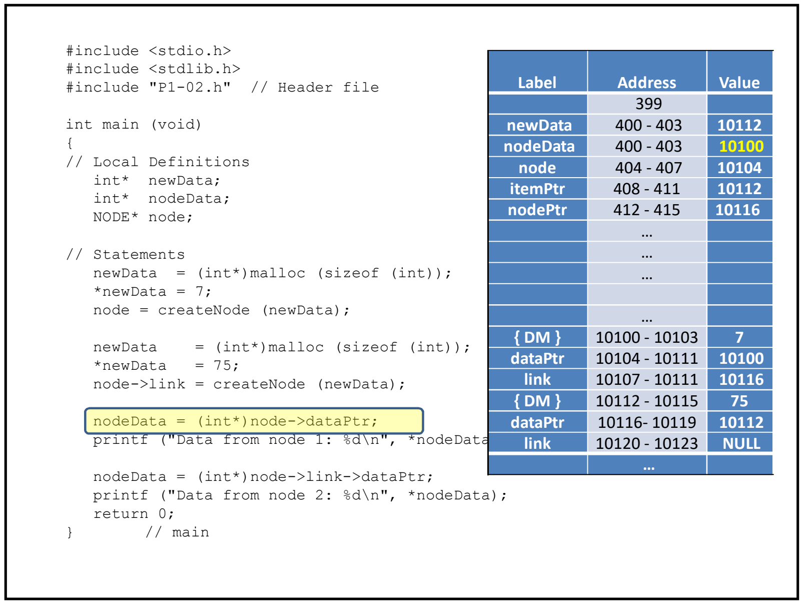
```
typedef struct node
{
   void*          dataPtr;
   struct node*   link;
} NODE;


NODE* createNode (void* itemPtr)
{
   NODE* nodePtr;
   nodePtr =
       (NODE*) malloc (sizeof (NODE));
   nodePtr->dataPtr = itemPtr;
   nodePtr->link    = NULL;

   return nodePtr;
}        // createNode
```

Label	Address	Value
	399	
newData	400 - 403	10112
nodeData	400 - 403	
node	404 - 407	10104
itemPtr	408 - 411	10112
nodePtr	412 - 415	10116
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	NULL
{ DM }	10112 - 10115	75
dataPtr	10116- 10119	10112
link	10120 - 10123	NULL
	…	



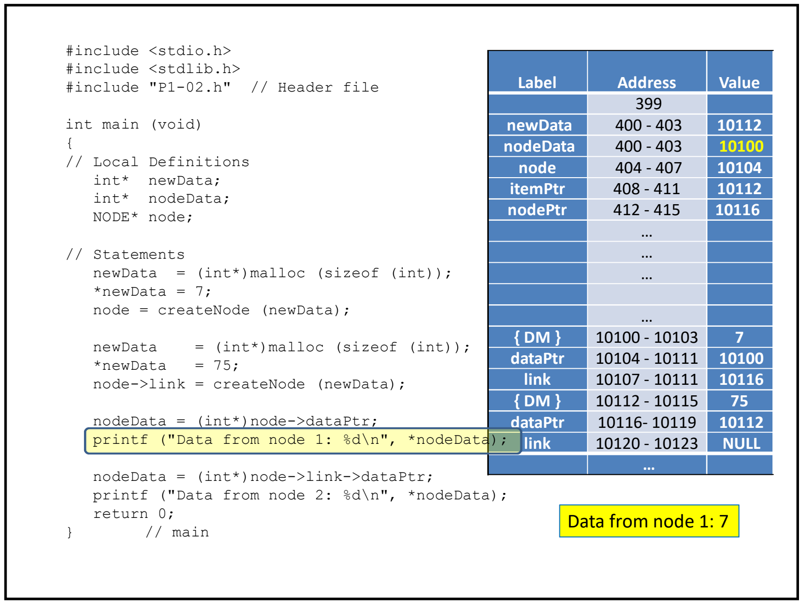
```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"  // Header file

int main (void)
{
// Local Definitions
   int*  newData;
   int*  nodeData;
   NODE* node;

// Statements
   newData  = (int*)malloc (sizeof (int));
   *newData = 7;
   node = createNode (newData);

   newData    = (int*)malloc (sizeof (int));
   *newData   = 75;
   node->link = createNode (newData);

   nodeData = (int*)node->dataPtr;
   printf ("Data from node 1: %d\n", *nodeData

   nodeData = (int*)node->link->dataPtr;
   printf ("Data from node 2: %d\n", *nodeData);
   return 0;
}        // main
```

Label	Address	Value
	399	
newData	400 - 403	10112
nodeData	400 - 403	
node	404 - 407	10104
itemPtr	408 - 411	10112
nodePtr	412 - 415	10116
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	10116
{ DM }	10112 - 10115	75
dataPtr	10116- 10119	10112
link	10120 - 10123	NULL
	…	



```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"  // Header file

int main (void)
{
// Local Definitions
   int*  newData;
   int*  nodeData;
   NODE* node;

// Statements
   newData  = (int*)malloc (sizeof (int));
   *newData = 7;
   node = createNode (newData);

   newData    = (int*)malloc (sizeof (int));
   *newData   = 75;
   node->link = createNode (newData);

   nodeData = (int*)node->dataPtr;
   printf ("Data from node 1: %d\n", *nodeData

   nodeData = (int*)node->link->dataPtr;
   printf ("Data from node 2: %d\n", *nodeData);
   return 0;
}       // main
```

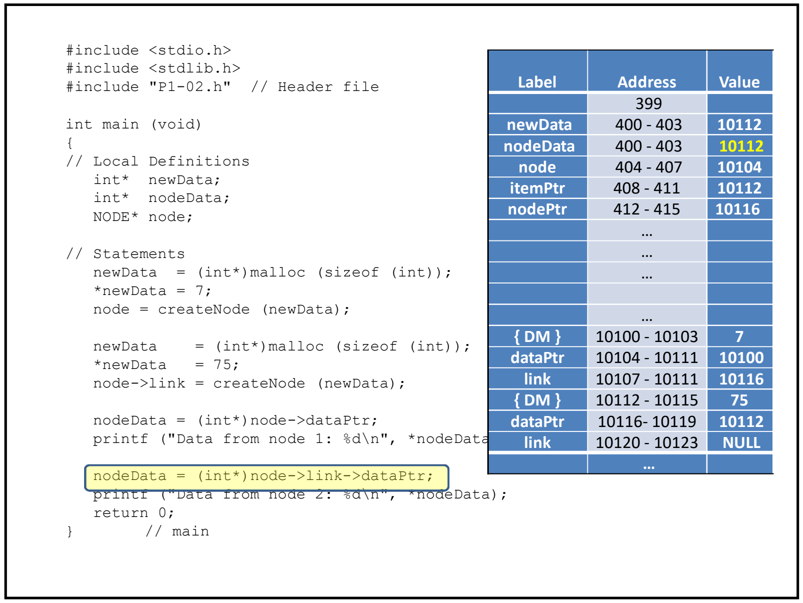
Label	Address	Value
	399	
newData	400 - 403	10112
nodeData	400 - 403	10100
node	404 - 407	10104
itemPtr	408 - 411	10112
nodePtr	412 - 415	10116
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	10116
{ DM }	10112 - 10115	75
dataPtr	10116- 10119	10112
link	10120 - 10123	NULL
	…	



```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"  // Header file

int main (void)
{
// Local Definitions
   int*  newData;
   int*  nodeData;
   NODE* node;

// Statements
   newData  = (int*)malloc (sizeof (int));
   *newData = 7;
   node = createNode (newData);

   newData    = (int*)malloc (sizeof (int));
   *newData   = 75;
   node->link = createNode (newData);

   nodeData = (int*)node->dataPtr;
   printf ("Data from node 1: %d\n", *nodeData);

   nodeData = (int*)node->link->dataPtr;
   printf ("Data from node 2: %d\n", *nodeData);
   return 0;
}       // main
```

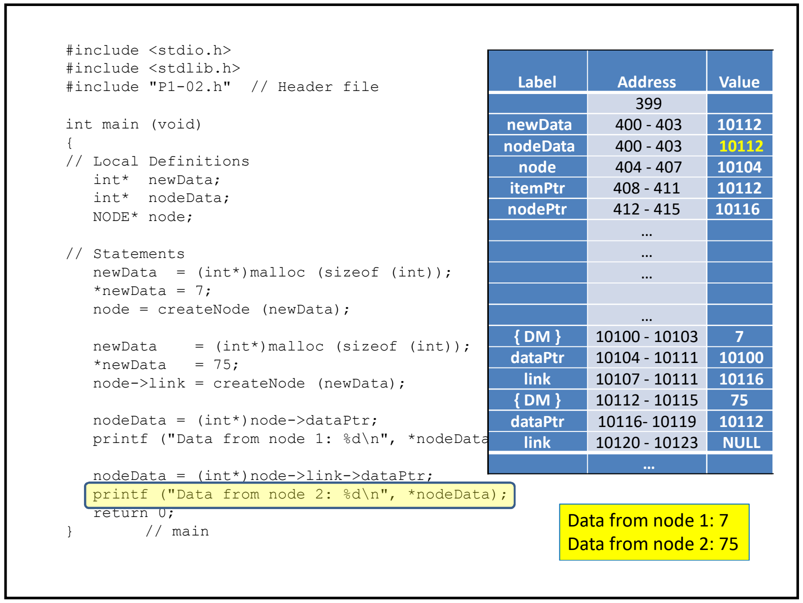
Label	Address	Value
	399	
newData	400 - 403	10112
nodeData	400 - 403	10100
node	404 - 407	10104
itemPtr	408 - 411	10112
nodePtr	412 - 415	10116
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	10116
{ DM }	10112 - 10115	75
dataPtr	10116- 10119	10112
link	10120 - 10123	NULL
	…	

```
Data from node 1: 7
```



```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"  // Header file

int main (void)
{
// Local Definitions
   int*  newData;
   int*  nodeData;
   NODE* node;

// Statements
   newData  = (int*)malloc (sizeof (int));
   *newData = 7;
   node = createNode (newData);

   newData    = (int*)malloc (sizeof (int));
   *newData   = 75;
   node->link = createNode (newData);

   nodeData = (int*)node->dataPtr;
   printf ("Data from node 1: %d\n", *nodeData

   nodeData = (int*)node->link->dataPtr;
   printf ("Data from node 2: %d\n", *nodeData);
   return 0;
}       // main
```

Label	Address	Value
	399	
newData	400 - 403	10112
nodeData	400 - 403	10112
node	404 - 407	10104
itemPtr	408 - 411	10112
nodePtr	412 - 415	10116
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	10116
{ DM }	10112 - 10115	75
dataPtr	10116- 10119	10112
link	10120 - 10123	NULL
	…	



```
#include <stdio.h>
#include <stdlib.h>
#include "P1-02.h"  // Header file

int main (void)
{
// Local Definitions
   int*  newData;
   int*  nodeData;
   NODE* node;

// Statements
   newData  = (int*)malloc (sizeof (int));
   *newData = 7;
   node = createNode (newData);

   newData    = (int*)malloc (sizeof (int));
   *newData   = 75;
   node->link = createNode (newData);

   nodeData = (int*)node->dataPtr;
   printf ("Data from node 1: %d\n", *nodeData

   nodeData = (int*)node->link->dataPtr;
   printf ("Data from node 2: %d\n", *nodeData);
   return 0;
}       // main
```

Label	Address	Value
	399	
newData	400 - 403	10112
nodeData	400 - 403	10112
node	404 - 407	10104
itemPtr	408 - 411	10112
nodePtr	412 - 415	10116
	…	
	…	
	…	
	…	
{ DM }	10100 - 10103	7
dataPtr	10104 - 10111	10100
link	10107 - 10111	10116
{ DM }	10112 - 10115	75
dataPtr	10116- 10119	10112
link	10120 - 10123	NULL
	…	

```
Data from node 1: 7
Data from node 2: 75
```

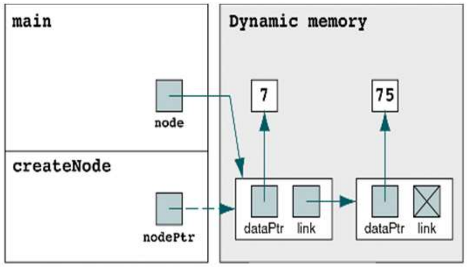
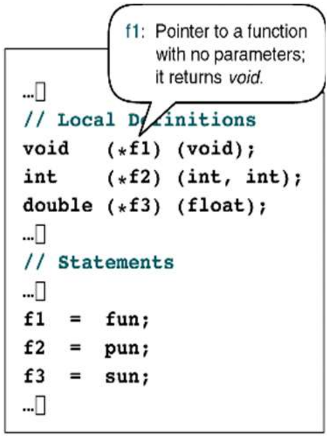


FIGURE 1-10 Structure for Two Linked Nodes


f1: Pointer to a function with no parameters; it returns *void*.

```
…
// Local Definitions
void   (*f1) (void);
int    (*f2) (int, int);
double (*f3) (float);
…
// Statements
…
f1  =  fun;
f2  =  pun;
f3  =  sun;
…
```

FIGURE 1-12 Pointers to Functions

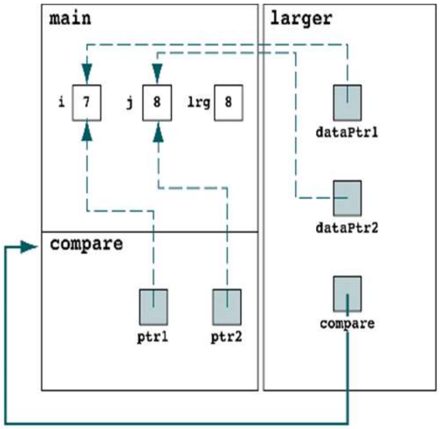
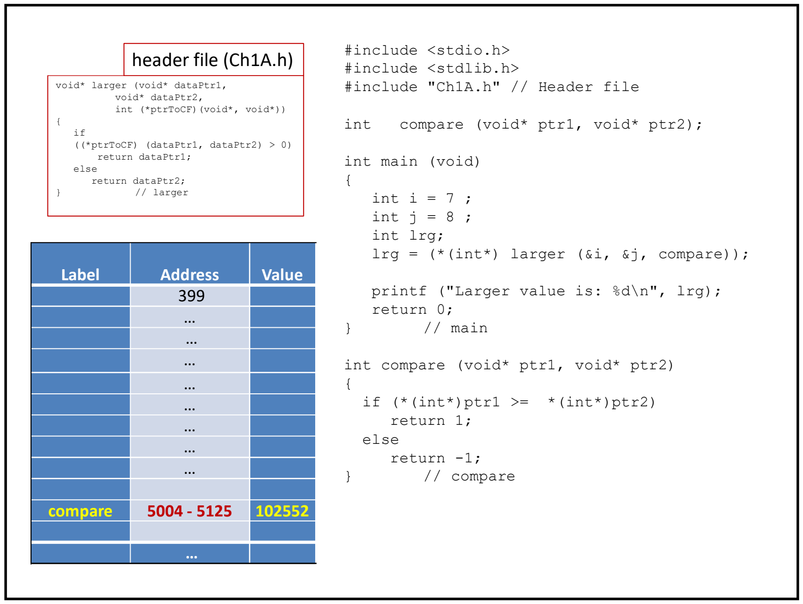


FIGURE 1-13 Design of Larger Function

```
void* larger (void* dataPtr1,    void* dataPtr2,
              int (*ptrToCF)(void*, void*))
{
   if ((*ptrToCF) (dataPtr1, dataPtr2) > 0)
       return dataPtr1;
   else
      return dataPtr2;
}       // larger
```

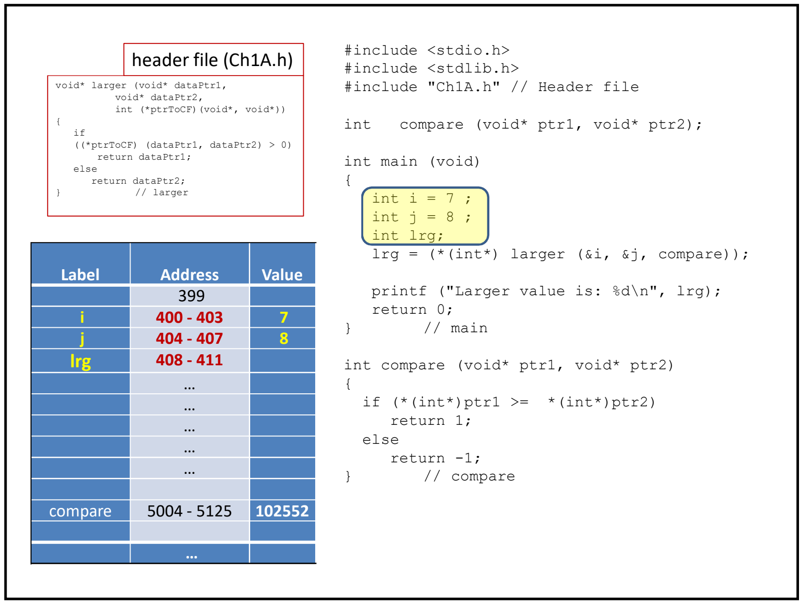


header file (Ch1A.h)

```
void* larger (void* dataPtr1,
         void* dataPtr2,
         int (*ptrToCF)(void*, void*))
{
   if
   ((*ptrToCF) (dataPtr1, dataPtr2) > 0)
       return dataPtr1;
   else
      return dataPtr2;
}            // larger
```

Label	Address	Value
	399	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
compare	5004 - 5125	102552
	…	

```
#include <stdio.h>
#include <stdlib.h>
#include "Ch1A.h" // Header file

int   compare (void* ptr1, void* ptr2);

int main (void)
{
   int i = 7 ;
   int j = 8 ;
   int lrg;
   lrg = (*(int*) larger (&i, &j, compare));

   printf ("Larger value is: %d\n", lrg);
   return 0;
}        // main

int compare (void* ptr1, void* ptr2)
{
  if (*(int*)ptr1 >=  *(int*)ptr2)
     return 1;
  else
     return -1;
}        // compare
```

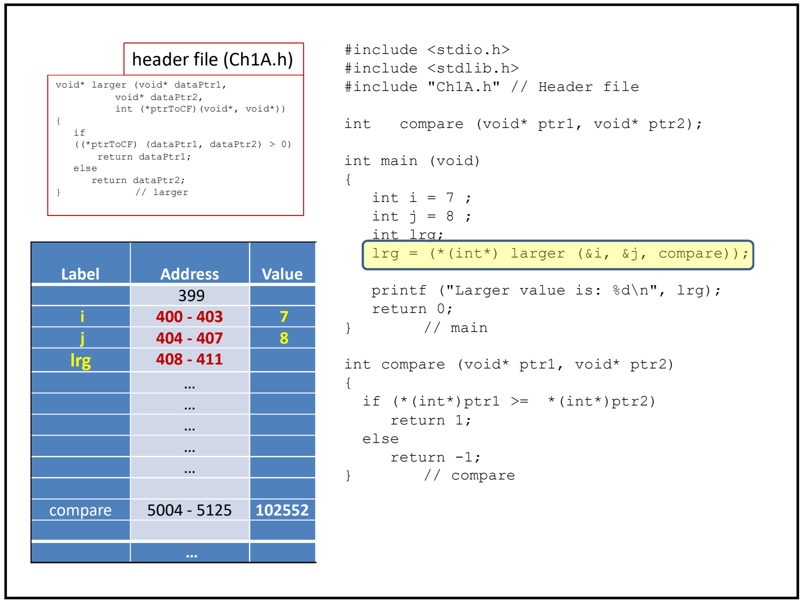


header file (Ch1A.h)

```
void* larger (void* dataPtr1,
         void* dataPtr2,
         int (*ptrToCF)(void*, void*))
{
   if
   ((*ptrToCF) (dataPtr1, dataPtr2) > 0)
       return dataPtr1;
   else
      return dataPtr2;
}            // larger
```

Label	Address	Value
	399	
i	400 - 403	7
j	404 - 407	8
lrg	408 - 411	
	…	
	…	
	…	
	…	
	…	
compare	5004 - 5125	102552
	…	

```
#include <stdio.h>
#include <stdlib.h>
#include "Ch1A.h" // Header file

int   compare (void* ptr1, void* ptr2);

int main (void)
{
   int i = 7 ;
   int j = 8 ;
   int lrg;
   lrg = (*(int*) larger (&i, &j, compare));

   printf ("Larger value is: %d\n", lrg);
   return 0;
}        // main

int compare (void* ptr1, void* ptr2)
{
  if (*(int*)ptr1 >=  *(int*)ptr2)
     return 1;
  else
     return -1;
}        // compare
```

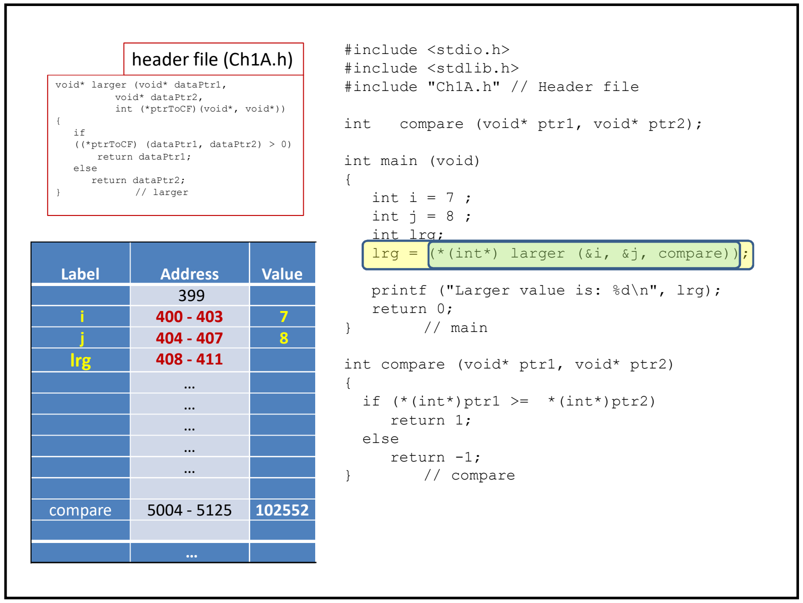


header file (Ch1A.h)

```
void* larger (void* dataPtr1,
         void* dataPtr2,
         int (*ptrToCF)(void*, void*))
{
   if
   ((*ptrToCF) (dataPtr1, dataPtr2) > 0)
       return dataPtr1;
   else
      return dataPtr2;
}            // larger
```

Label	Address	Value
	399	
i	400 - 403	7
j	404 - 407	8
lrg	408 - 411	
	…	
	…	
	…	
	…	
	…	
compare	5004 - 5125	102552
	…	

```
#include <stdio.h>
#include <stdlib.h>
#include "Ch1A.h" // Header file

int   compare (void* ptr1, void* ptr2);

int main (void)
{
   int i = 7 ;
   int j = 8 ;
   int lrg;
   lrg = (*(int*) larger (&i, &j, compare));

   printf ("Larger value is: %d\n", lrg);
   return 0;
}        // main

int compare (void* ptr1, void* ptr2)
{
  if (*(int*)ptr1 >=  *(int*)ptr2)
     return 1;
  else
     return -1;
}        // compare
```

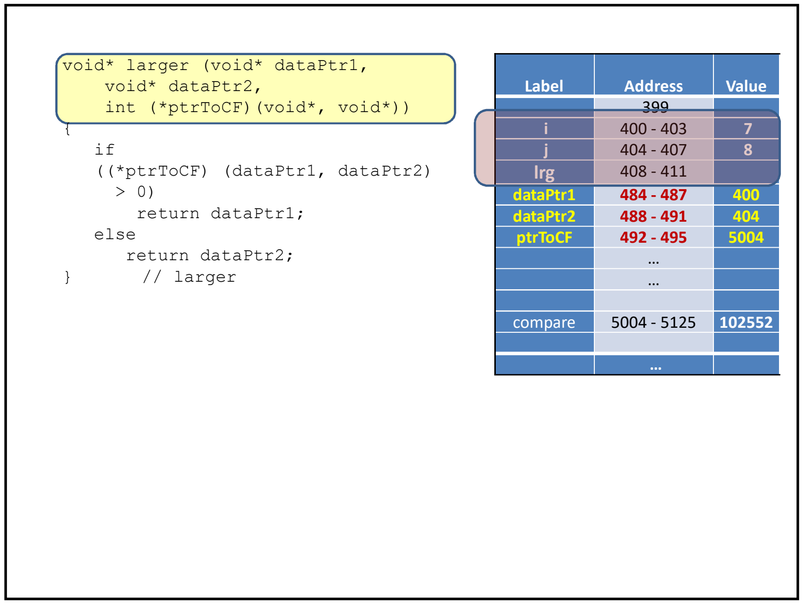


header file (Ch1A.h)

```
void* larger (void* dataPtr1,
         void* dataPtr2,
         int (*ptrToCF)(void*, void*))
{
   if
   ((*ptrToCF) (dataPtr1, dataPtr2) > 0)
       return dataPtr1;
   else
      return dataPtr2;
}            // larger
```

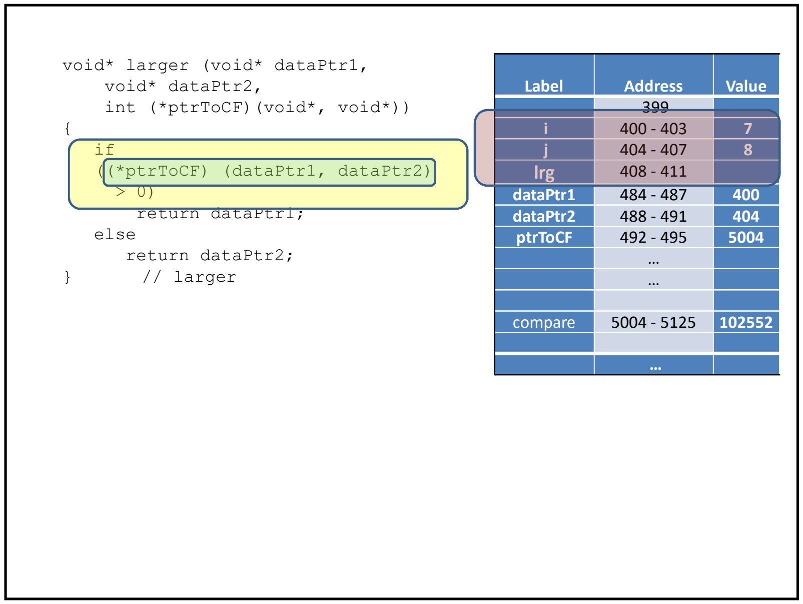
Label	Address	Value
	399	
i	400 - 403	7
j	404 - 407	8
lrg	408 - 411	
	…	
	…	
	…	
	…	
	…	
compare	5004 - 5125	102552
	…	

```
#include <stdio.h>
#include <stdlib.h>
#include "Ch1A.h" // Header file

int   compare (void* ptr1, void* ptr2);

int main (void)
{
   int i = 7 ;
   int j = 8 ;
   int lrg;
   lrg = (*(int*) larger (&i, &j, compare));

   printf ("Larger value is: %d\n", lrg);
   return 0;
}        // main

int compare (void* ptr1, void* ptr2)
{
  if (*(int*)ptr1 >=  *(int*)ptr2)
     return 1;
  else
     return -1;
}        // compare
```



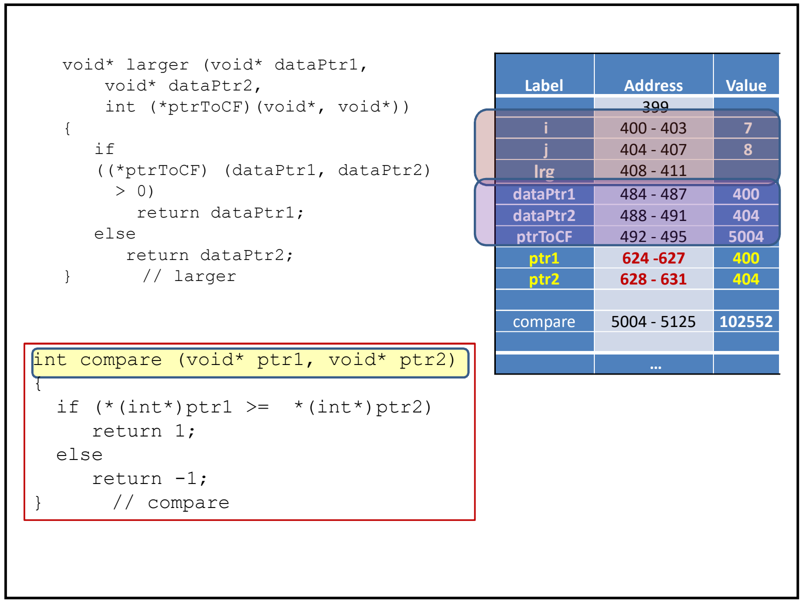
```
void* larger (void* dataPtr1,
    void* dataPtr2,
    int (*ptrToCF)(void*, void*))
{
   if
   ((*ptrToCF) (dataPtr1, dataPtr2)
     > 0)
       return dataPtr1;
   else
      return dataPtr2;
}       // larger
```

Label	Address	Value
	399	
i	400 - 403	7
j	404 - 407	8
lrg	408 - 411	
dataPtr1	484 - 487	400
dataPtr2	488 - 491	404
ptrToCF	492 - 495	5004
	…	
	…	
compare	5004 - 5125	102552
	…	



```
void* larger (void* dataPtr1,
    void* dataPtr2,
    int (*ptrToCF)(void*, void*))
{
   if
   ((*ptrToCF) (dataPtr1, dataPtr2)
     > 0)
       return dataPtr1;
   else
      return dataPtr2;
}       // larger
```

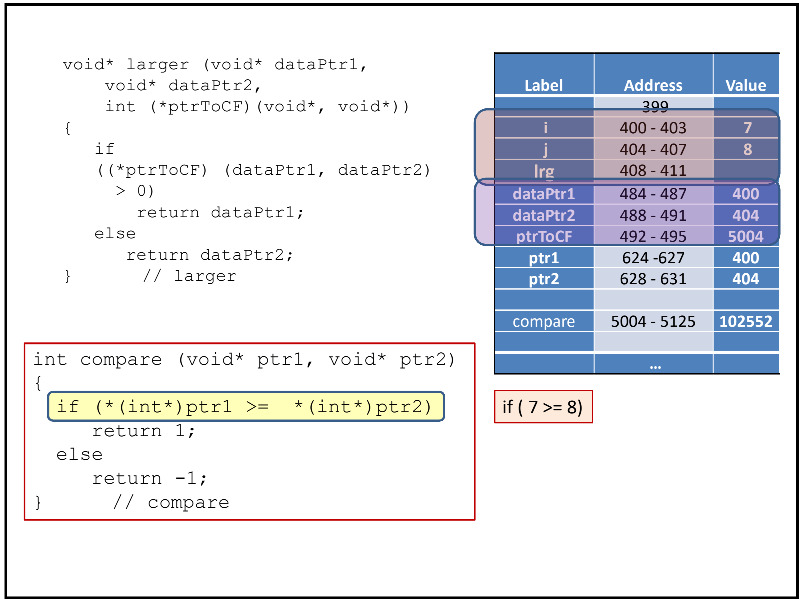
Label	Address	Value
	399	
i	400 - 403	7
j	404 - 407	8
lrg	408 - 411	
dataPtr1	484 - 487	400
dataPtr2	488 - 491	404
ptrToCF	492 - 495	5004
	…	
	…	
compare	5004 - 5125	102552
	…	



```
void* larger (void* dataPtr1,
    void* dataPtr2,
    int (*ptrToCF)(void*, void*))
{
   if
   ((*ptrToCF) (dataPtr1, dataPtr2)
     > 0)
       return dataPtr1;
   else
      return dataPtr2;
}       // larger
```

Label	Address	Value
	399	
i	400 - 403	7
j	404 - 407	8
lrg	408 - 411	
dataPtr1	484 - 487	400
dataPtr2	488 - 491	404
ptrToCF	492 - 495	5004
ptr1	624 -627	400
ptr2	628 - 631	404
compare	5004 - 5125	102552
	…	

```
int compare (void* ptr1, void* ptr2)
{
  if (*(int*)ptr1 >=  *(int*)ptr2)
     return 1;
  else
     return -1;
}      // compare
```

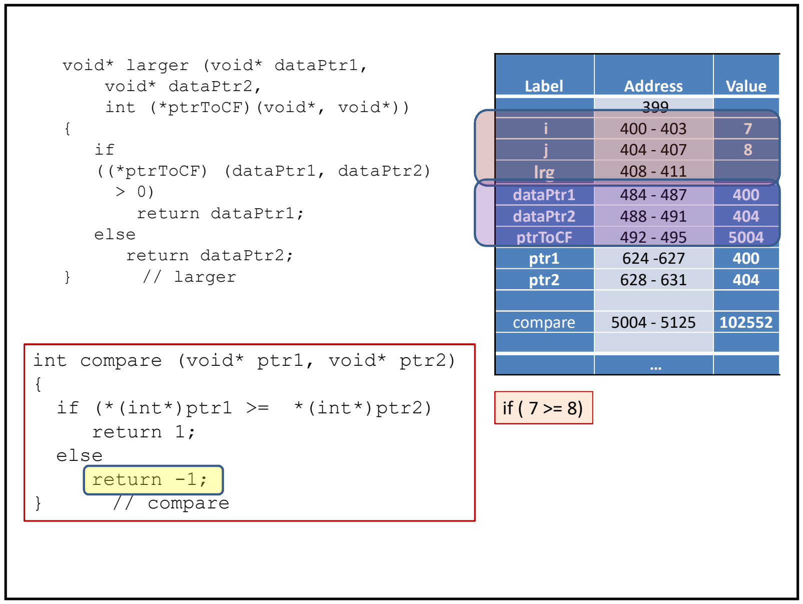


```
void* larger (void* dataPtr1,
    void* dataPtr2,
    int (*ptrToCF)(void*, void*))
{
   if
   ((*ptrToCF) (dataPtr1, dataPtr2)
     > 0)
       return dataPtr1;
   else
      return dataPtr2;
}       // larger
```

Label	Address	Value
	399	
i	400 - 403	7
j	404 - 407	8
lrg	408 - 411	
dataPtr1	484 - 487	400
dataPtr2	488 - 491	404
ptrToCF	492 - 495	5004
ptr1	624 -627	400
ptr2	628 - 631	404
compare	5004 - 5125	102552
	…	

if (7 >= 8)

```
int compare (void* ptr1, void* ptr2)
{
  if (*(int*)ptr1 >=  *(int*)ptr2)
     return 1;
  else
     return -1;
}      // compare
```

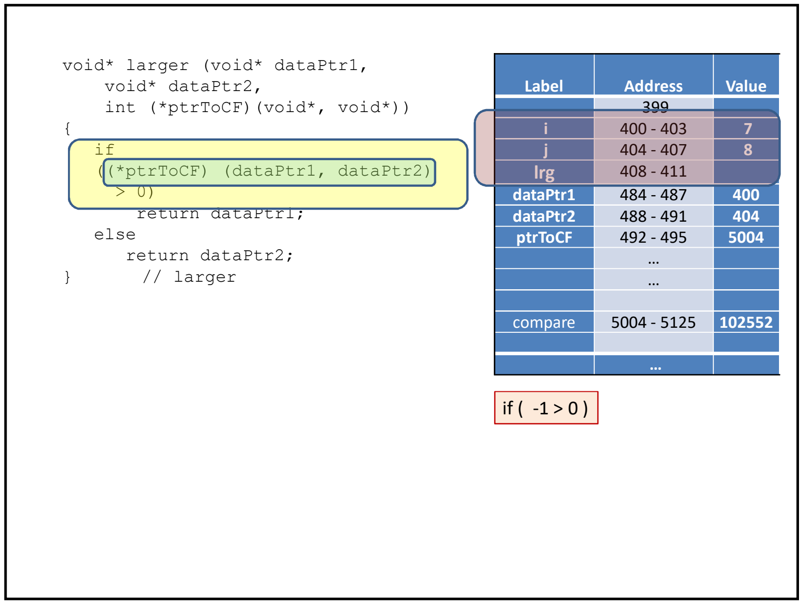


```
void* larger (void* dataPtr1,
     void* dataPtr2,
     int (*ptrToCF)(void*, void*))
{
    if
    ((*ptrToCF) (dataPtr1, dataPtr2)
      > 0)
         return dataPtr1;
    else
        return dataPtr2;
}         // larger
```

Label	Address	Value
	399	
i	400 - 403	7
j	404 - 407	8
lrg	408 - 411	
dataPtr1	484 - 487	400
dataPtr2	488 - 491	404
ptrToCF	492 - 495	5004
ptr1	624 -627	400
ptr2	628 - 631	404
compare	5004 - 5125	102552
	...	

```
int compare (void* ptr1, void* ptr2)
{
  if (*(int*)ptr1 >=  *(int*)ptr2)
     return 1;
  else
     return -1;
}         // compare
```

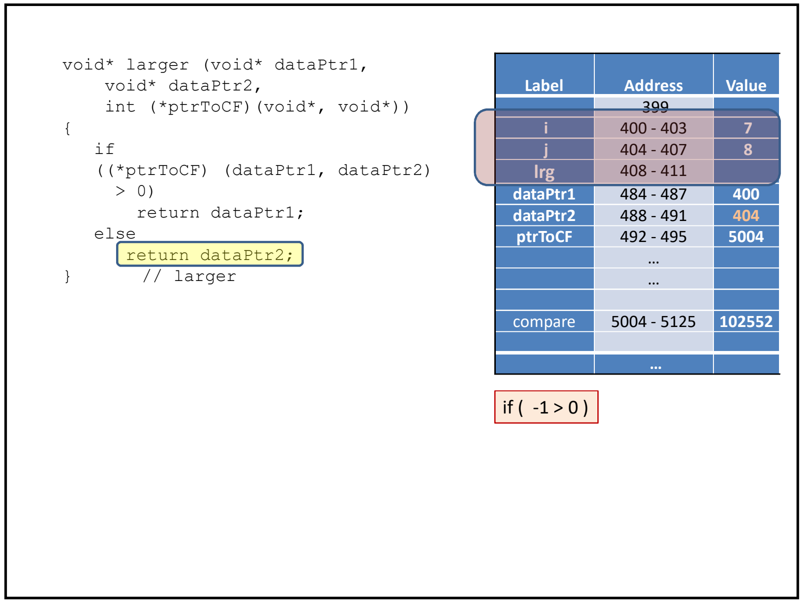
if (7 >= 8)



```
void* larger (void* dataPtr1,
     void* dataPtr2,
     int (*ptrToCF)(void*, void*))
{
    if
    ((*ptrToCF) (dataPtr1, dataPtr2)
      > 0)
         return dataPtr1;
    else
        return dataPtr2;
}         // larger
```

Label	Address	Value
	399	
i	400 - 403	7
j	404 - 407	8
lrg	408 - 411	
dataPtr1	484 - 487	400
dataPtr2	488 - 491	404
ptrToCF	492 - 495	5004
	...	
	...	
compare	5004 - 5125	102552
	...	

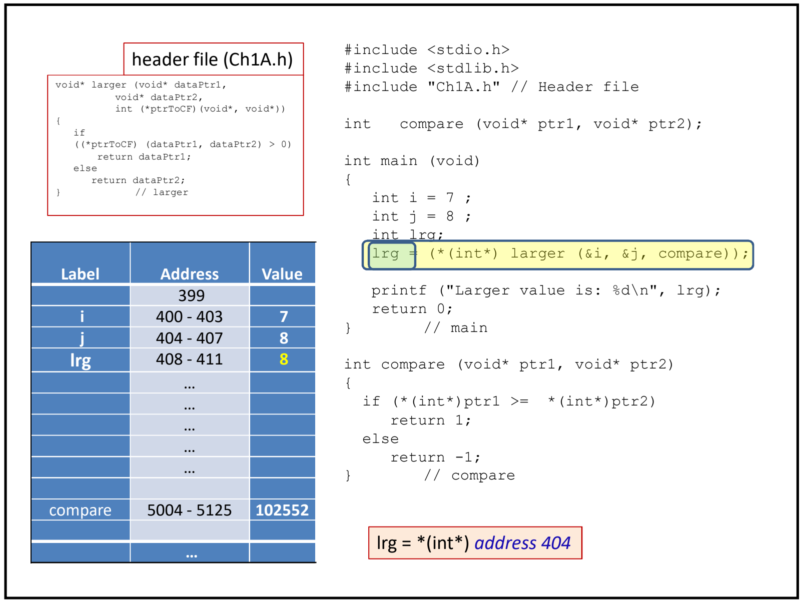
if (-1 > 0)



```
void* larger (void* dataPtr1,
     void* dataPtr2,
     int (*ptrToCF)(void*, void*))
{
    if
    ((*ptrToCF) (dataPtr1, dataPtr2)
      > 0)
         return dataPtr1;
    else
        return dataPtr2;
}         // larger
```

Label	Address	Value
	399	
i	400 - 403	7
j	404 - 407	8
lrg	408 - 411	
dataPtr1	484 - 487	400
dataPtr2	488 - 491	404
ptrToCF	492 - 495	5004
	...	
	...	
compare	5004 - 5125	102552
	...	

if (-1 > 0)



header file (Ch1A.h)

```
void* larger (void* dataPtr1,
          void* dataPtr2,
          int (*ptrToCF)(void*, void*))
{
   if
   ((*ptrToCF) (dataPtr1, dataPtr2) > 0)
       return dataPtr1;
   else
      return dataPtr2;
}             // larger
```

Label	Address	Value
	399	
i	400 - 403	7
j	404 - 407	8
lrg	408 - 411	8
	...	
	...	
	...	
	...	
	...	
compare	5004 - 5125	102552
	...	

```
#include <stdio.h>
#include <stdlib.h>
#include "Ch1A.h" // Header file

int   compare (void* ptr1, void* ptr2);

int main (void)
{
   int i = 7 ;
   int j = 8 ;
   int lrg;
   lrg = (*(int*) larger (&i, &j, compare));

   printf ("Larger value is: %d\n", lrg);
   return 0;
}        // main

int compare (void* ptr1, void* ptr2)
{
  if (*(int*)ptr1 >=  *(int*)ptr2)
     return 1;
  else
     return -1;
}        // compare
```

lrg = *(int*) *address 404*

header file (Ch1A.h)

```
void* larger (void* dataPtr1,
              void* dataPtr2,
              int (*ptrToCF)(void*, void*))
{
   if
   ((*ptrToCF) (dataPtr1, dataPtr2) > 0)
       return dataPtr1;
   else
      return dataPtr2;
}            // larger
```

Label	Address	Value
	399	
i	400 - 403	7
j	404 - 407	8
lrg	408 - 411	8
	…	
	…	
	…	
	…	
	…	
compare	5004 - 5125	102552
	…	

```
#include <stdio.h>
#include <stdlib.h>
#include "Ch1A.h" // Header file

int   compare (void* ptr1, void* ptr2);

int main (void)
{
   int i = 7 ;
   int j = 8 ;
   int lrg;
   lrg = (*(int*) larger (&i, &j, compare));

   printf ("Larger value is: %d\n", lrg);
   return 0;
}        // main

int compare (void* ptr1, void* ptr2)
{
  if (*(int*)ptr1 >=  *(int*)ptr2)
     return 1;
  else
     return -1;
}        // compare
```

```
Larger value is: 8
```

header file (Ch1A.h)

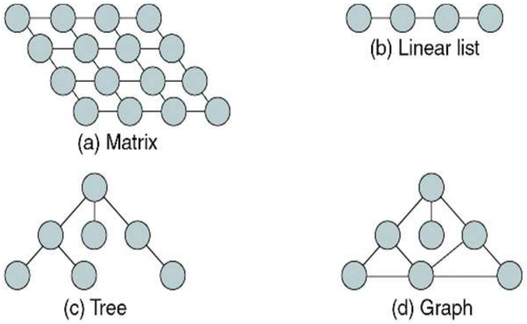
```
void* larger (void* dataPtr1,
              void* dataPtr2,
              int (*ptrToCF)(void*, void*))
{
   if
   ((*ptrToCF) (dataPtr1, dataPtr2) > 0)
       return dataPtr1;
   else
      return dataPtr2;
}            // larger
```

Label	Address	Value
	399	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

```
#include <stdio.h>
#include <stdlib.h>
#include "Ch1A.h" // Header file

int   compare (void* ptr1, void* ptr2);

int main (void)
{
   float f1 = 73.4;
   float f2 = 81.7;
   float lrg;
   lrg = (*(float*) larger(&f1,&f2, compare));

   printf ("Larger value is: %d\n", lrg);
   return 0;
}        // main

int compare (void* ptr1, void* ptr2)
{
  if (*(float*)ptr1 >=  *(float*)ptr2)
     return 1;
  else
     return -1;
}        // compare
```

changes required to make this compare two floating point values (notice func. `larger` does NOT change.
 - passing func. pointer makes `larger` generic

GENERIC STRUCTURE TYPES



WE NOW HAVE THE TOOL SET TO BUILD THESE STRUCTURES