VIGNAN Institute of Technology and Science

Department of Computer Science and Engineering

DATA STRUCTURES

B.Tech II year I Sem

by
S Akshara Reddy
Assistant Professor
Email id: akshara.revelly@gmail.com
Blog: http://aksharatechnicalstuff.blogspot.com/

Course Outcomes

1

Ability to select the data structures that efficiently model the information in a problem.

2

Ability to assess efficiency trade-offs among different data structure implementations or combinations.

3

Implement and know the application of algorithms for sorting and pattern matching.

4

Design programs using a variety of data structures

Course Syllabus

Unit-1

Syllabus

Introduction to Data Structures, Abstract Data Types, Linear list- Singly Linked List Implementation, Insertion, Deletion and searching operations on linear list, stack operations, array and linked representation of stacks, stack applications, queues operations, array and linked representations

Learning Objectives

UNIT - I

1. Classification of Data Structures
2. Array Versus Linked List
3. Linked List Representation & Classification
4. Singly Linked List & its Operations
5. Stack-Operations, Array and Linked List representation & Applications
6. Queue-Operations, Array and Linked List representation

Data Structure?

- Data Structure is basically a group of data elements that are put together under one name and which defines a particular way of storing and organizing in a computer memory so that it can used efficiently.

- Reema Thareja

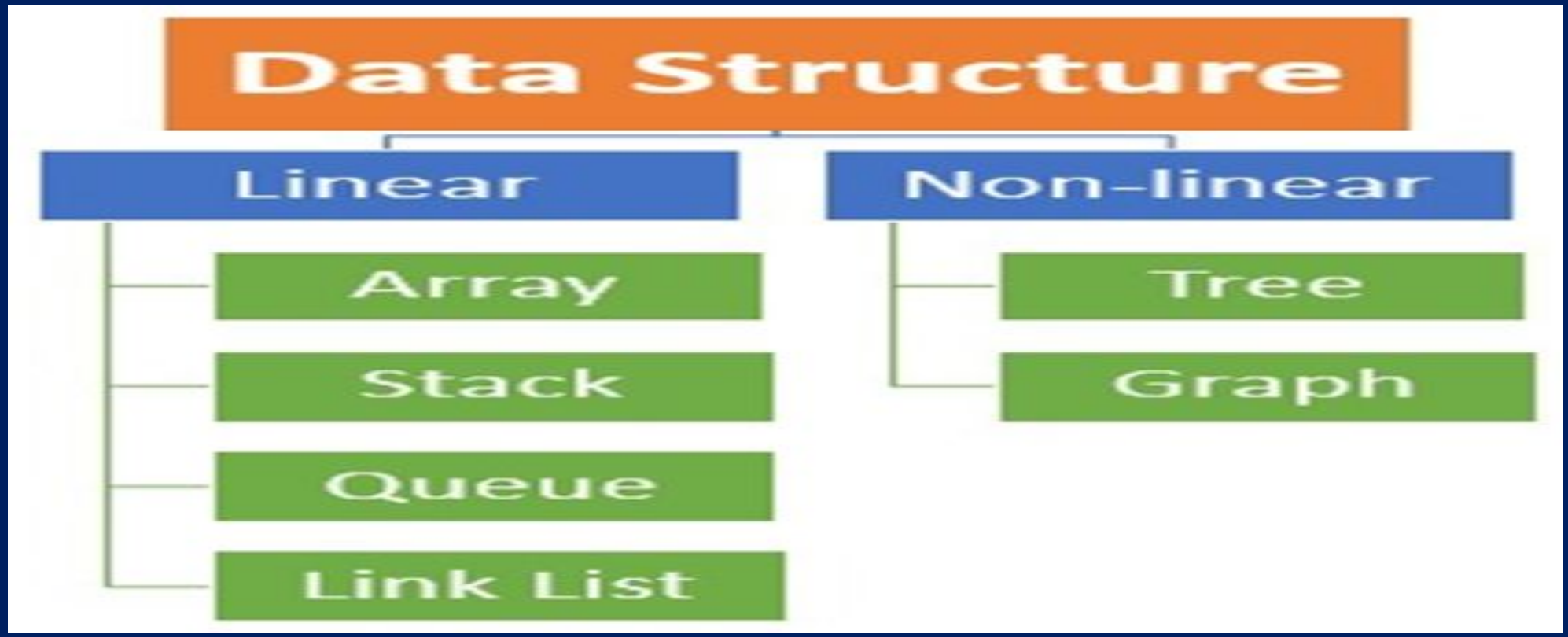
- Data Structure is a way of collecting and organizing data in such a way that we can perform operations on these data in an effective way.
- Data Structure is an arrangement of data in computer's memory.

Importance of Data Structures

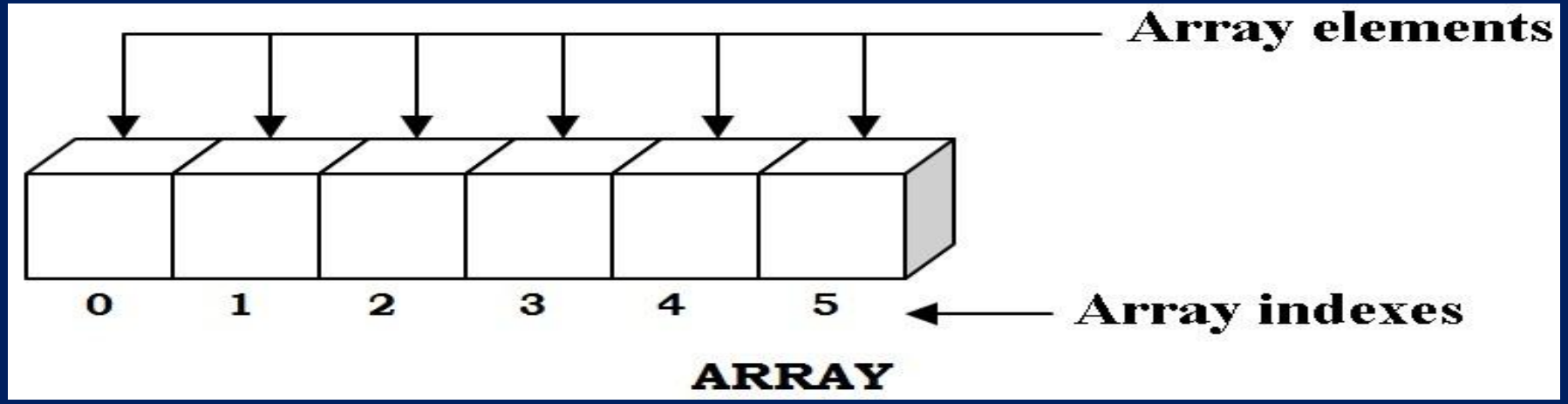
- Data structures are the building blocks of any program or the software. They are widely used in almost every aspect of Computer Science like:
  - Operating System
  - Compiler Design
  - Graphics
  - Database Management system and many more
- Data structures provide a means to manage large amounts of data efficiently for uses such as large databases and internet indexing services.
- Choosing the appropriate data structure for a program is the most difficult task for a programmer.

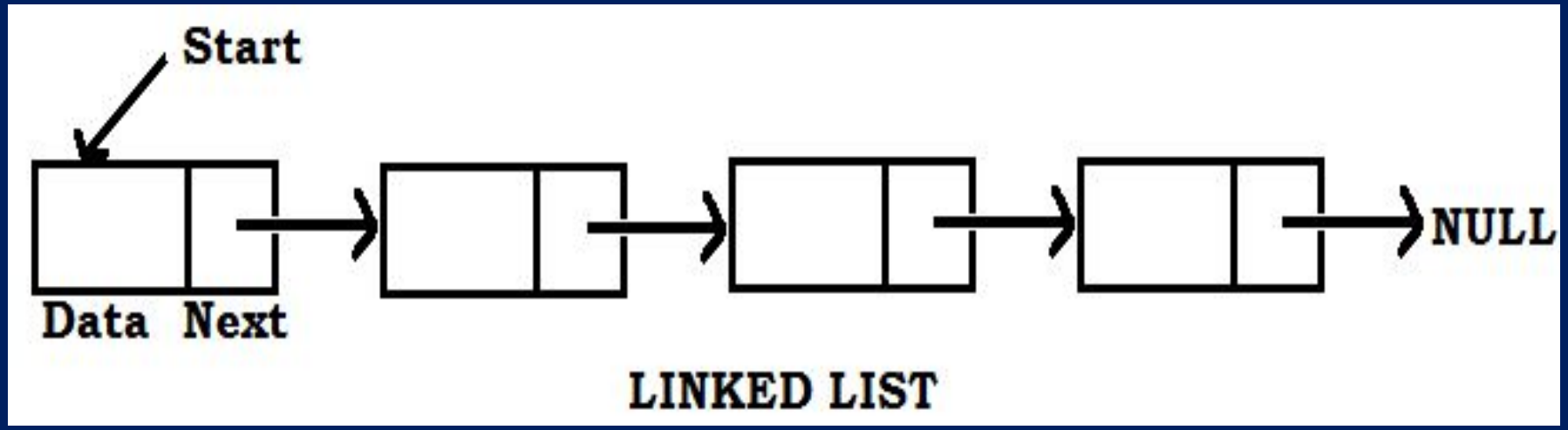
Classification of Data Structures

Data Structure is an arrangement of data in computer's memory. Depending upon the arrangement of data, data structures can be classified as:

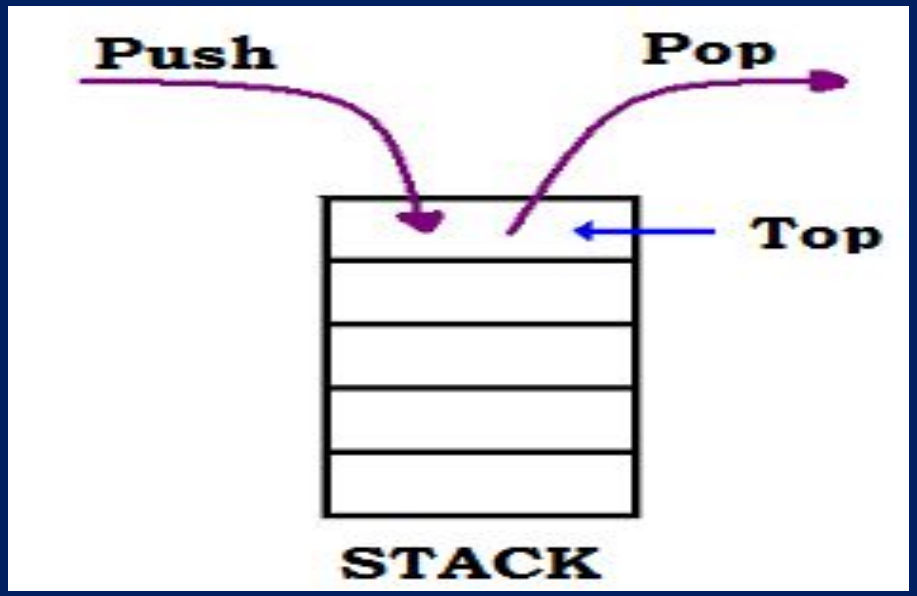


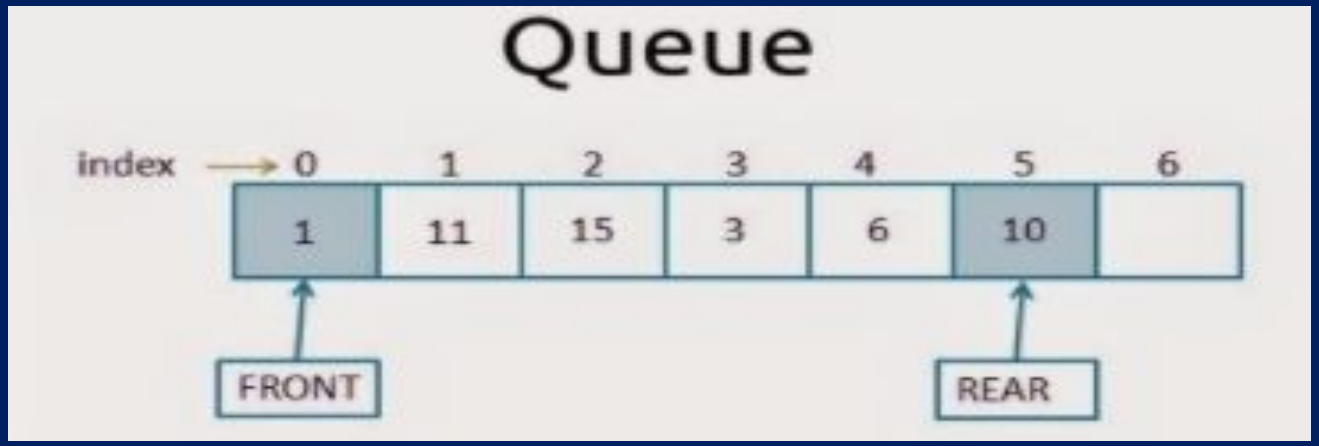
Linear Data Structures



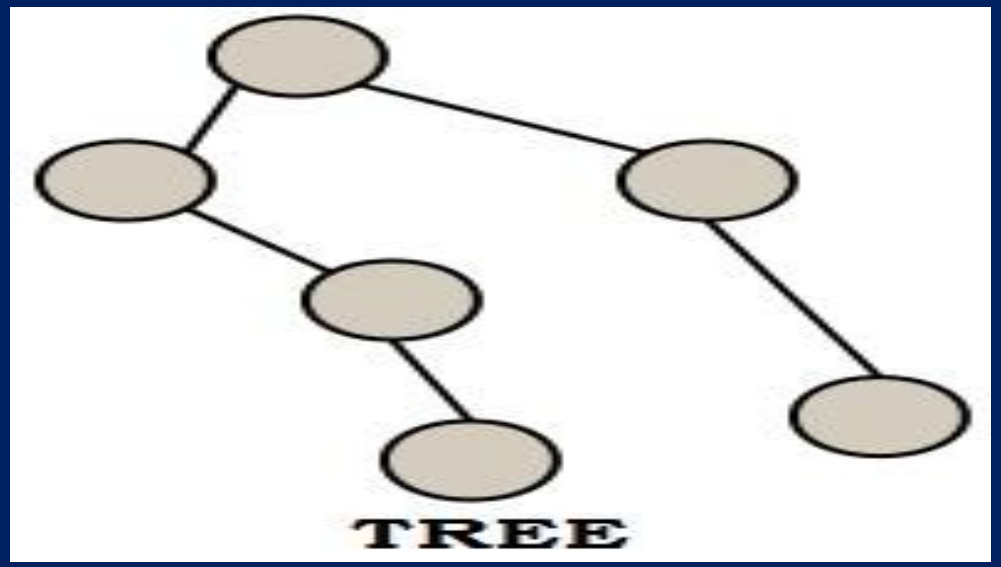


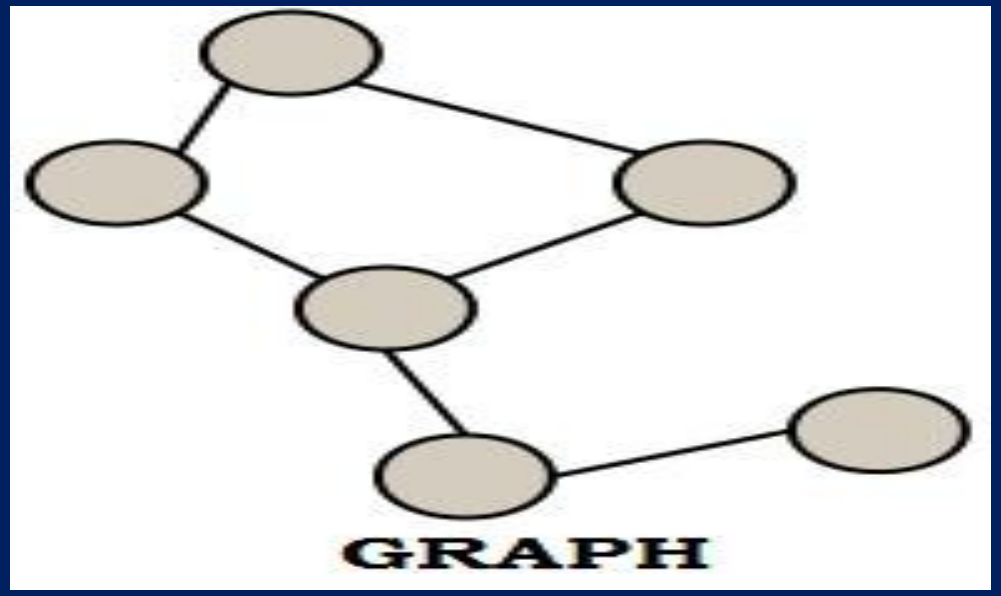
Linear Data Structures



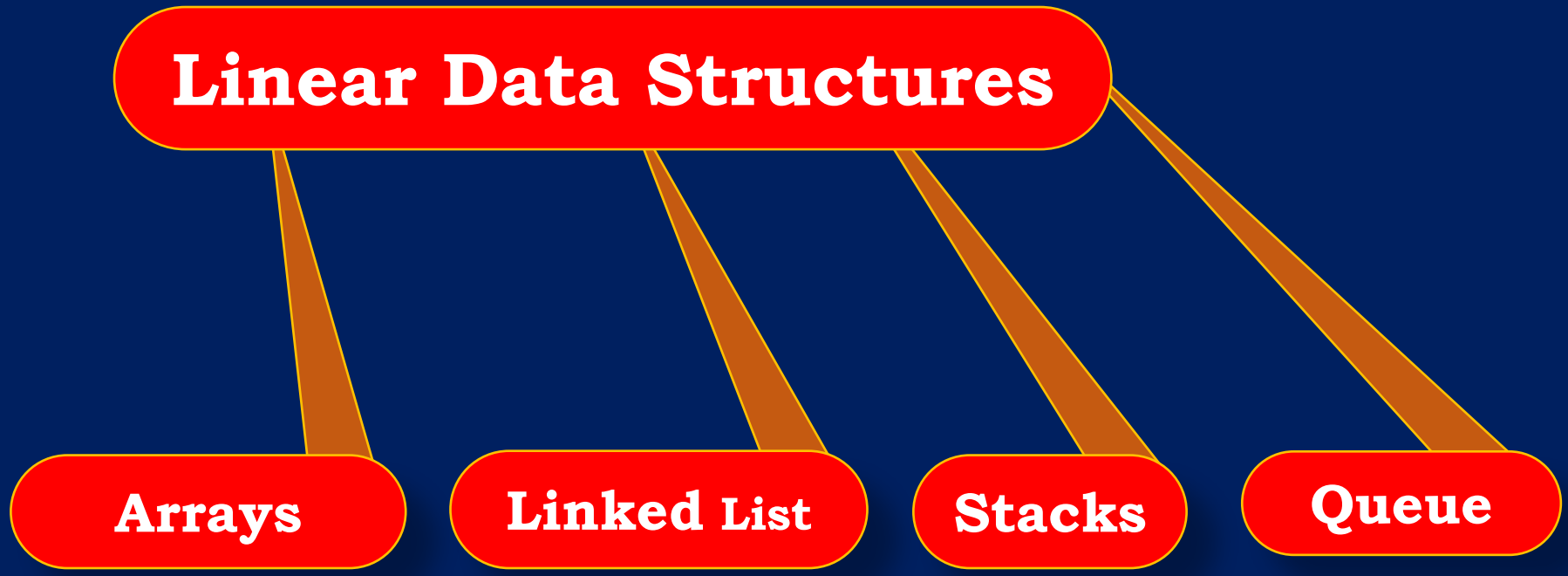


Non-Linear Data Structures





Types of Linear Data Structures



..	10	20	30	40	..	..	..

VIGNAN Institute of Technology and Science

Department of Computer Science and Engineering

Topic: Linked List

Data Structures
B.Tech II year I Sem

by
S Akshara Reddy
Assistant Professor
Email id: akshara.revelly@gmail.com
Blog: http://aksharatechnicalstuff.blogspot.com/

Array Versus Linked List

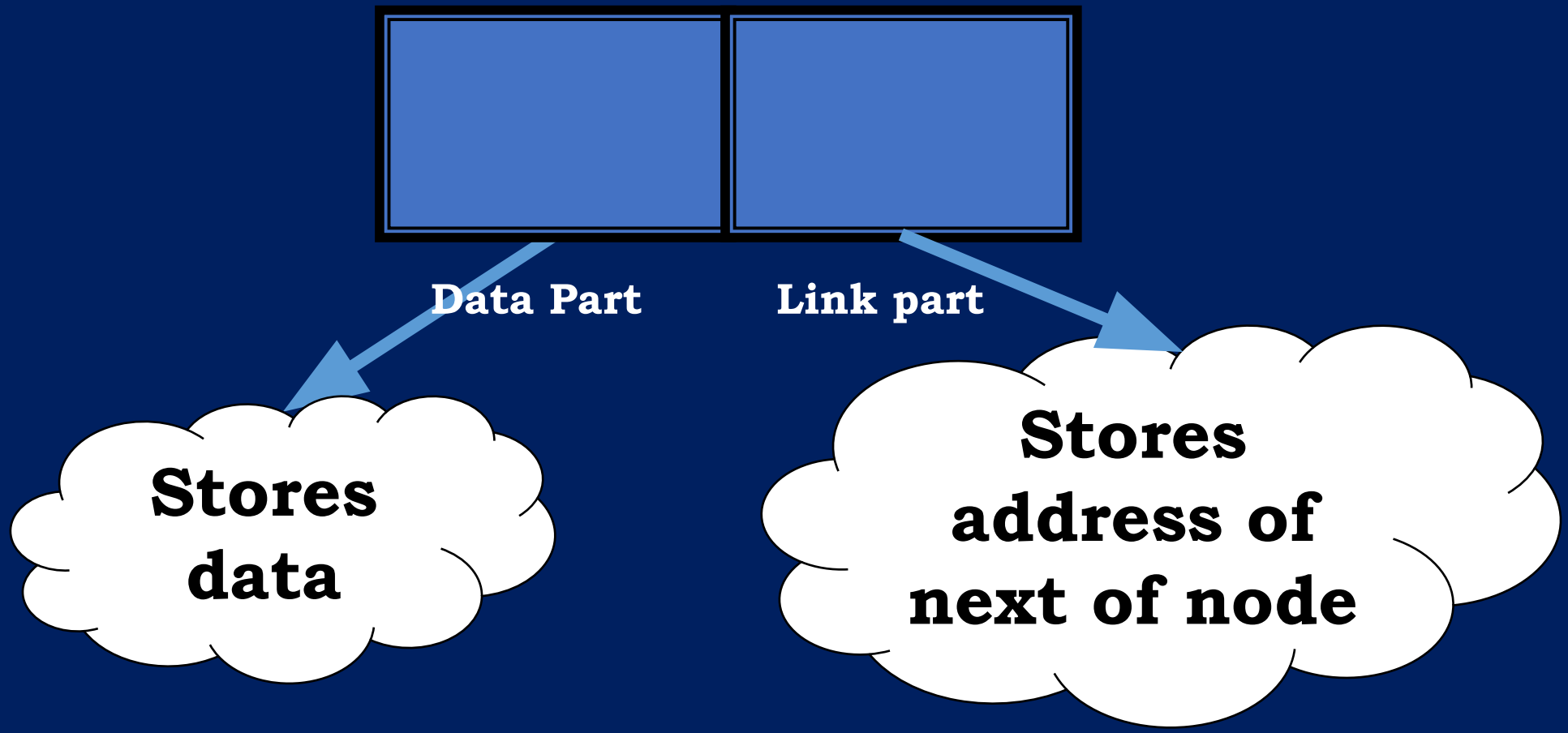
Array	Linked List
Fixed size	Dynamic size
Insertion and deletion are inefficient.	Insertion and deletion are efficient.
Random Access	No Random Access
Memory Wastage	No memory wastage
Memory Overflow	No memory overflow
Sequential access is fast	Sequential access is slow

Linked List

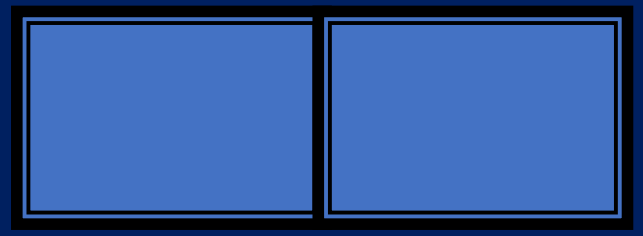
- A Linked list is a data structure that consist of a sequence of nodes such each node, contains two field.
  - ✔ one field that contains data item.
  - ✔ Another field that stores address of next node.
- Linked list provides easy implementation for other data structures such as stacks, queues and trees.

Linked List

- A Node in linked list represented as:



Node as Structure



Data Part

Link part

```
struct link
{
int data;
struct link *link;
}
```

Integer variable ‘data’ is used to integer value.

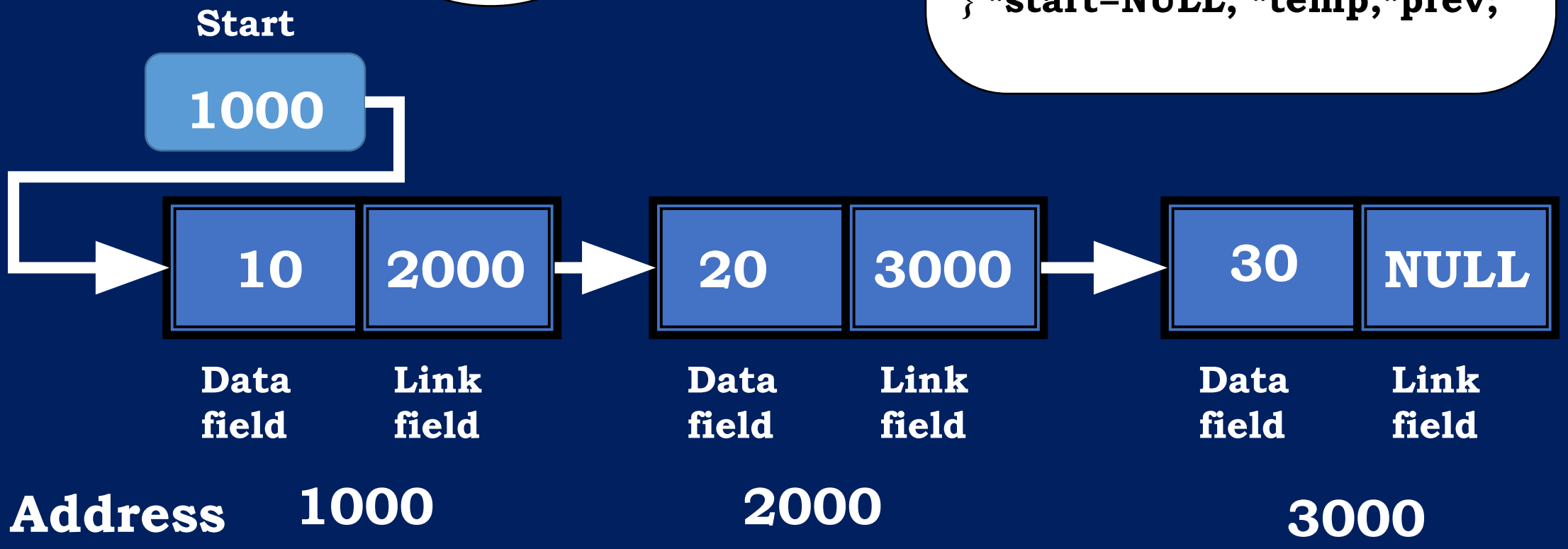
Structure pointer variable which stores address of another structure variable or node

Self Referential Structures

Linked List

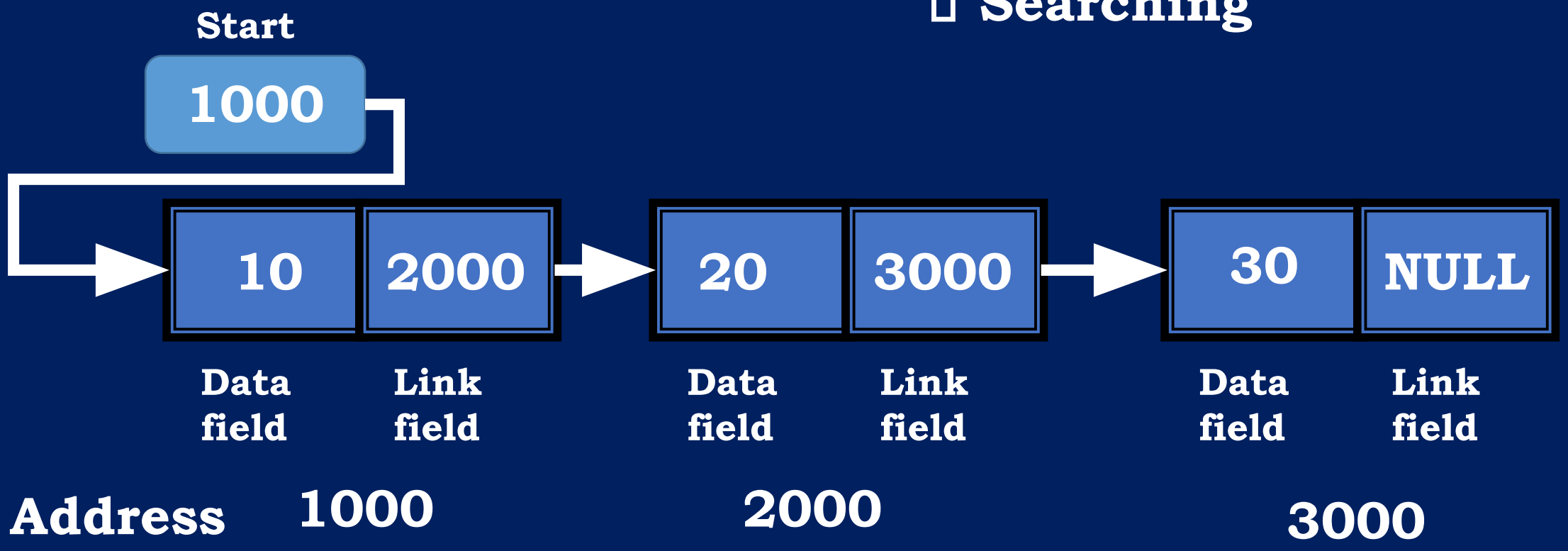
Start is structure pointer variable always stores first node address in the linked list

```
struct link
{
int data;
struct link *link;
} *start=NULL, *temp,*prev;
```



Operations On Linked List

- Insertion
- Deletion
- Searching



Linked List

- Types of Linked List
  - ✔ Singly Linked List
  - ✔ Doubly Linked List
  - ✔ Circular Linked List
    - ❑ Circular singly Linked list
    - ❑ Circular Doubly Linked list

VIGNAN Institute of Technology and Science

Department of Computer Science and Engineering

Singly Linked List

Data Structures
B.Tech II year I Sem

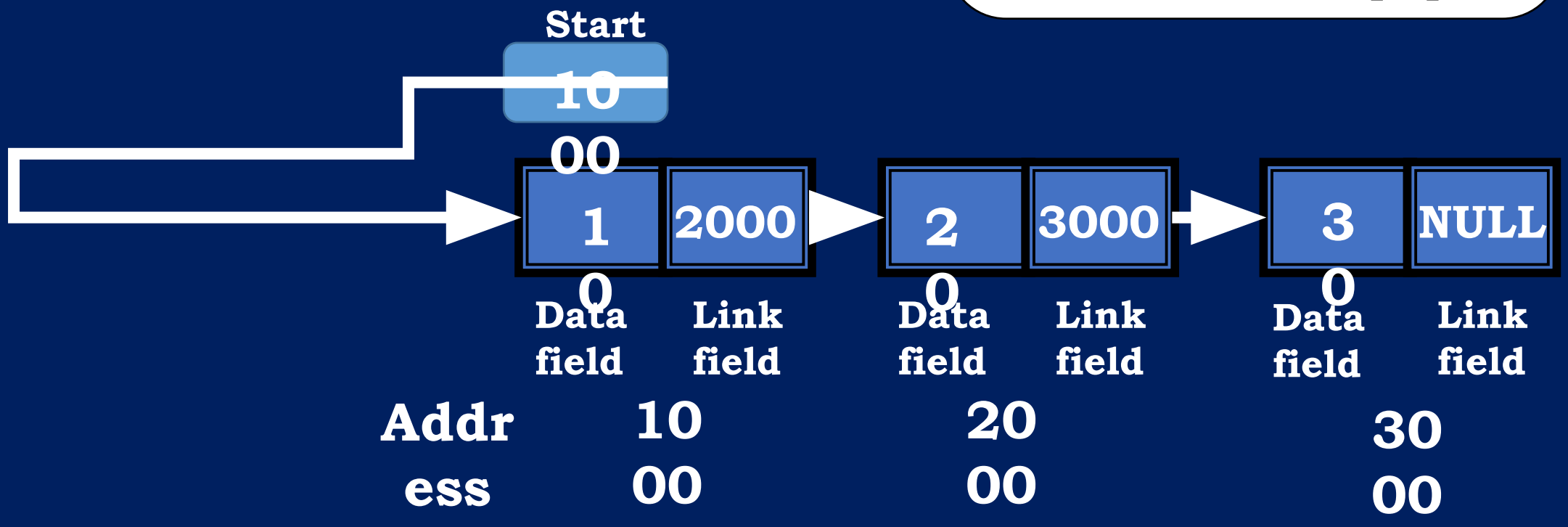
by
R Akshara
Assistant Professor
Email id: akshara.revelly@gmail.com

Singly Linked List

- A Singly Linked list is the simplest type of linked list in which every node contains data and a pointer to the next node.
- A singly linked list contains only one link field pointing to the next node in the list so that a singly linked list allows traversal of data only in one way.

Singly Linked List

```
struct slink
{
int data;
struct slink *link;
} *start=NULL, *temp,*prev;
```



Operations on Singly Linked List

- Creating a Linked List
- Traversing a Linked List
- Insertion of a node in to a Singly Linked List
- Deletion of a node from a Singly Linked List
- Searching for an element in a singly linked list.

Creating a Linked List

- **Steps to be followed while creating a list**

1. **Allocate block of memory for node & Divide it into data part and link part**
2. **Accept data part and assign Null to link**
3. **Establish links to newly created node.**
4. **Repeat Step1 to step5 for creating nodes.**

Creating a Linked List

Allocate block of memory for node &

Divide it into data part and link part

Creating a Linked List

Allocate block of memory for node &

Divide it into data part and link part

```
temp=(struct slink *) malloc(sizeof(struct slink));
```

Creating a Linked List

Allocate block of memory for node &

Divide it into data part and link part

```
temp=(struct slink *) malloc(sizeof(struct slink));
```

Allocates Block of memory of size 4 or 8 bytes

Creating a Linked List

Allocate block of memory for node &

Divide it into data part and link part

```
temp=(struct slink *) malloc(sizeof(struct slink));
```

1000

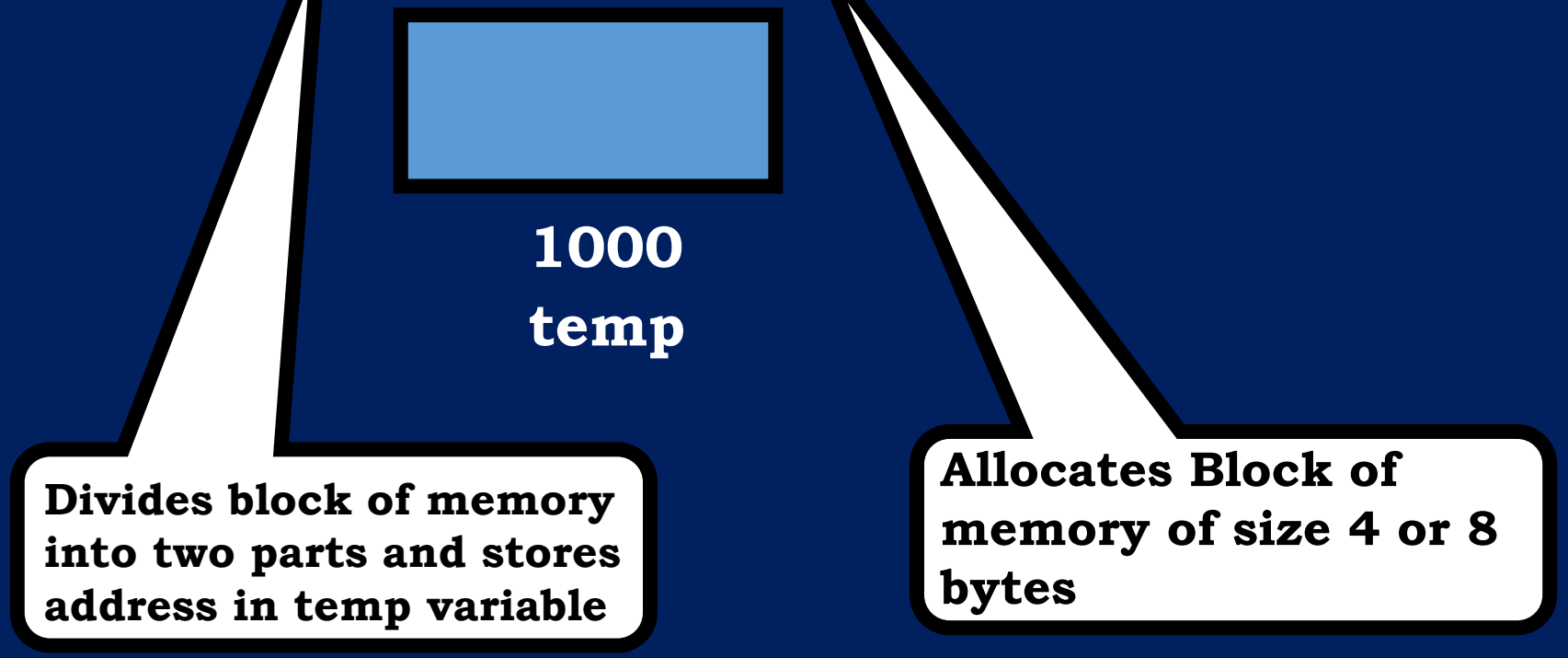
Allocates Block of memory of size 4 or 8 bytes

Creating a Linked List

Allocate block of memory for node &

Divide it into data part and link part

```
temp=(struct slink *) malloc(sizeof(struct slink));
```

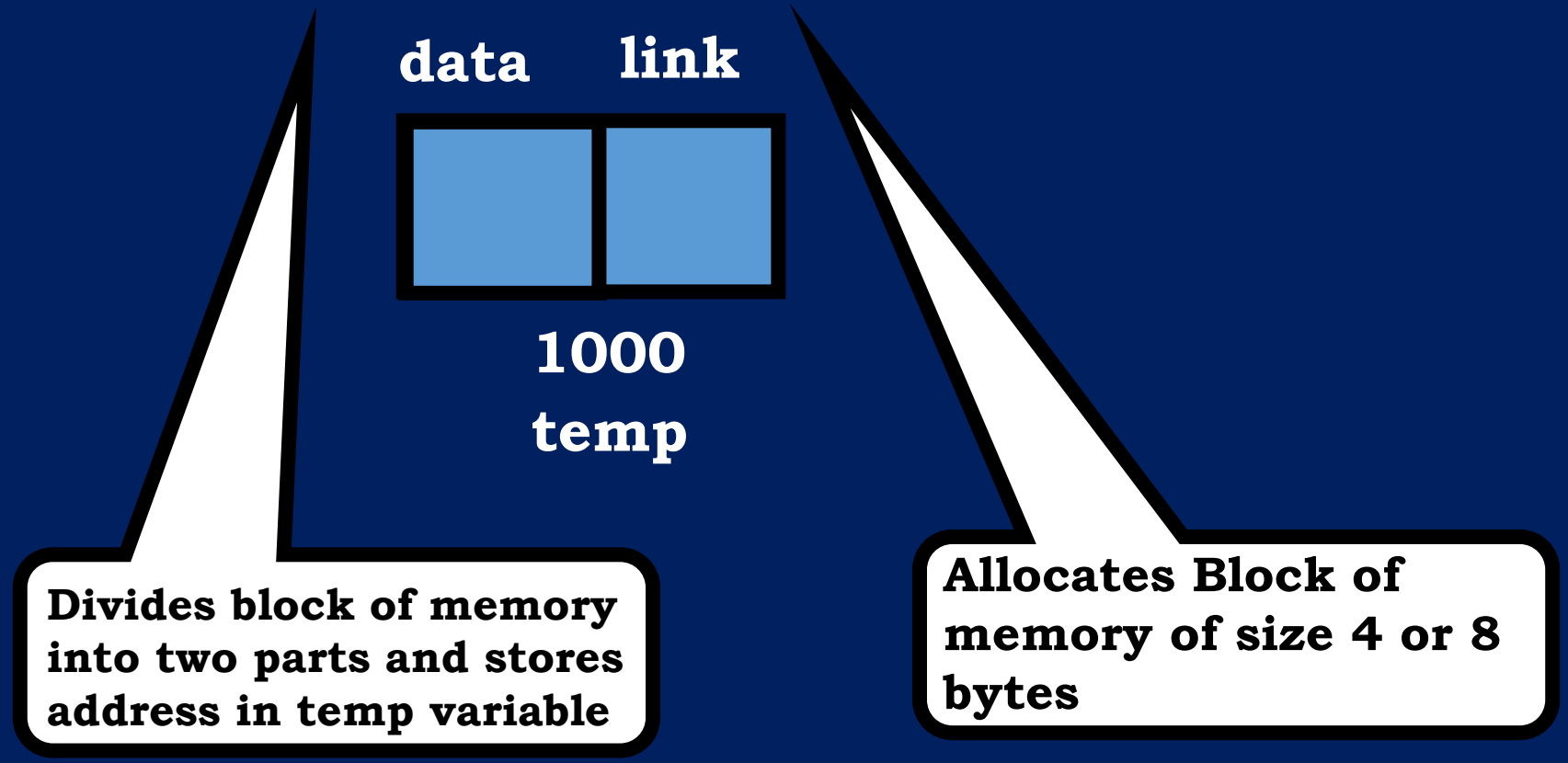


Creating a Linked List

Allocate block of memory for node &

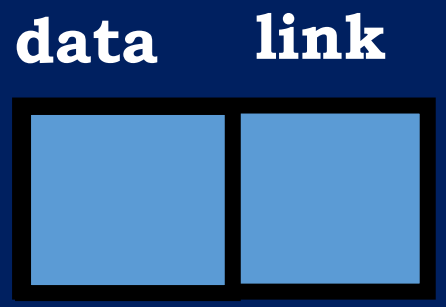
Divide it into data part and link part

```
temp=(struct slink *) malloc(sizeof(struct slink));
```



Creating a Linked List

Accept data part and assign Null to link

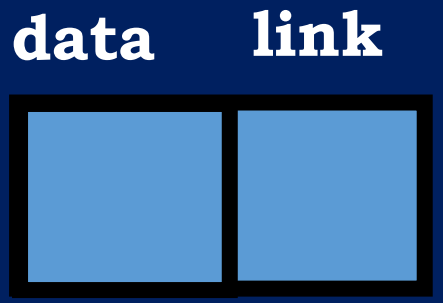


1000

temp

Creating a Linked List

Accept data part and assign Null to link



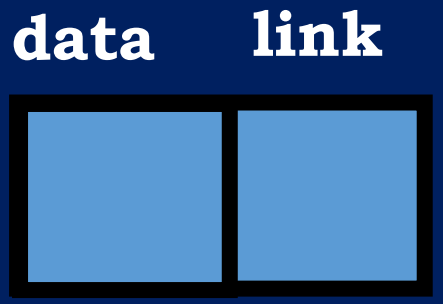
1000

temp

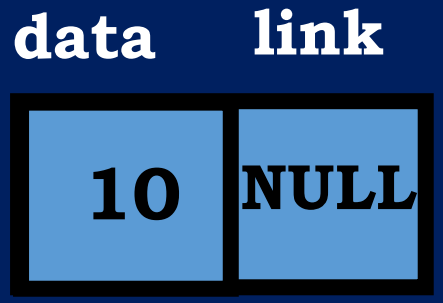
```
scanf("%d",&temp->data);
temp->link=NULL;
```

Creating a Linked List

Accept data part and assign Null to link

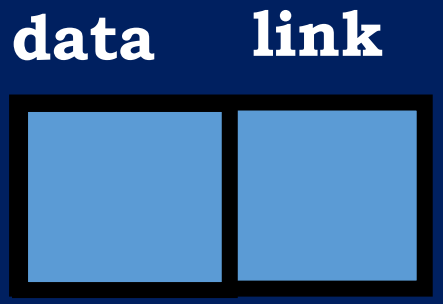



```
scanf("%d",&temp->data);
temp->link=NULL;
```



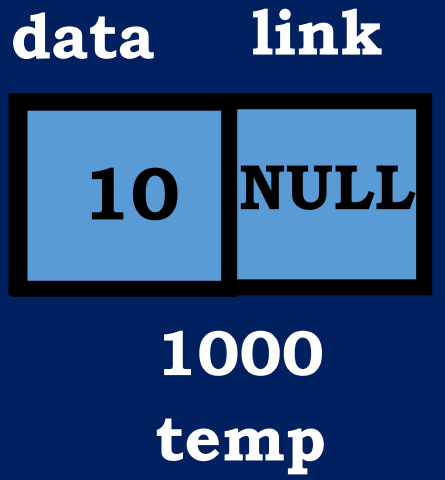
Creating a Linked List

Accept data part and assign Null to link



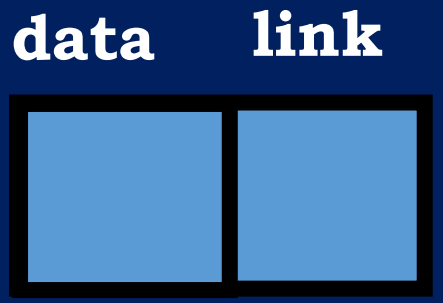

```
scanf("%d",&temp->data);
temp->link=NULL;
```

If it is first node in the list then assign address of that node to start pointer



Creating a Linked List

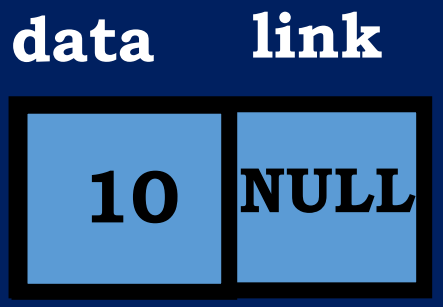
Accept data part and assign Null to link



1000

temp

```
scanf("%d",&temp->data);
temp->link=NULL;
```



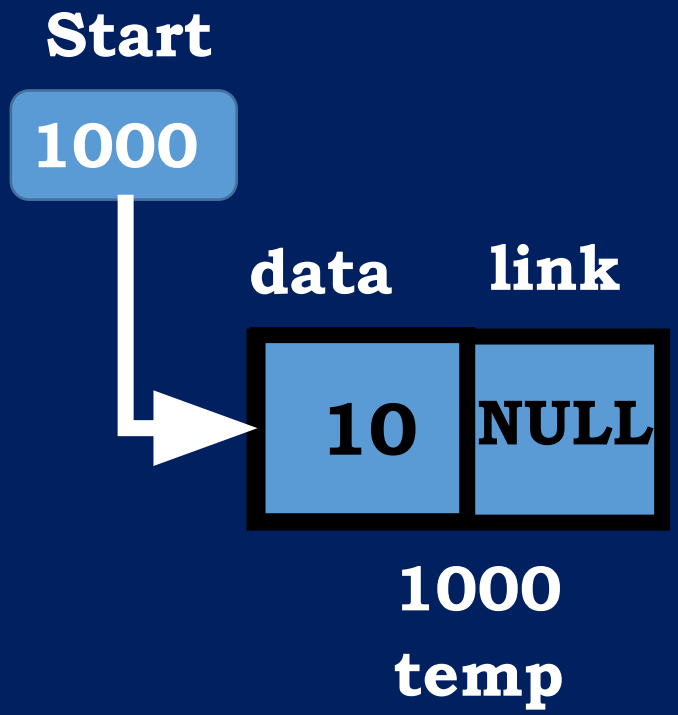
1000

temp

If it is first node in the list then assign address of that node to start pointer

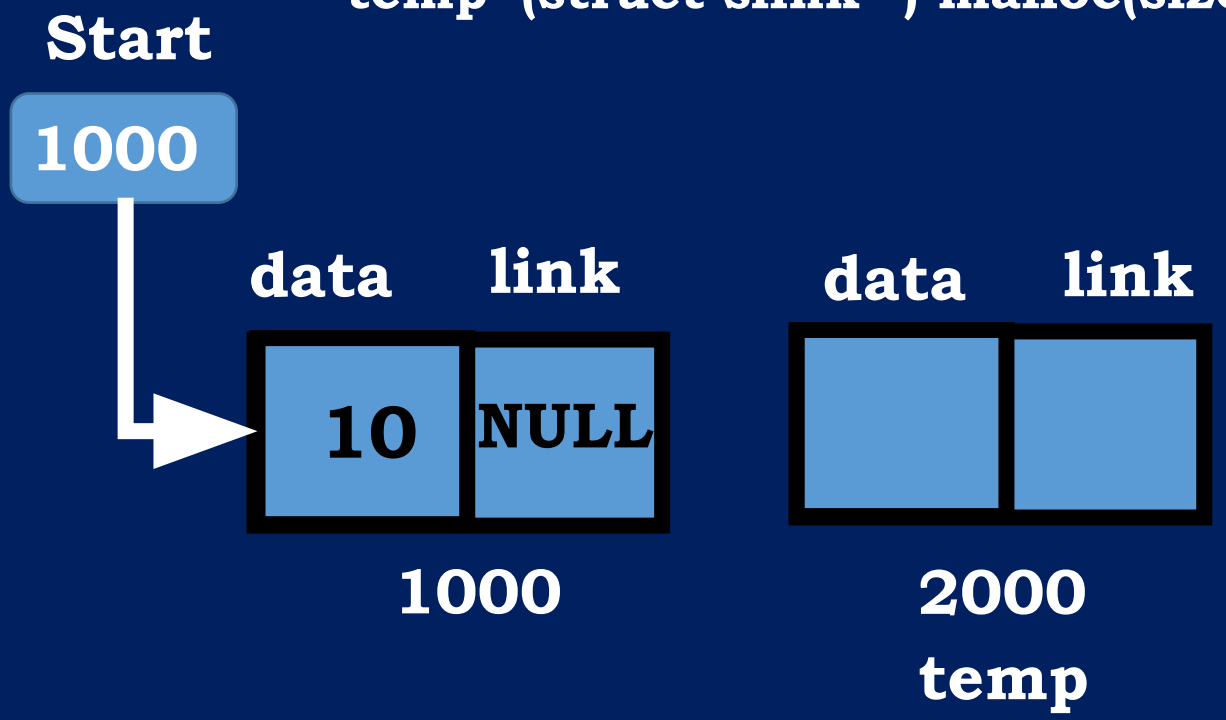
start = temp

Creating a Linked List



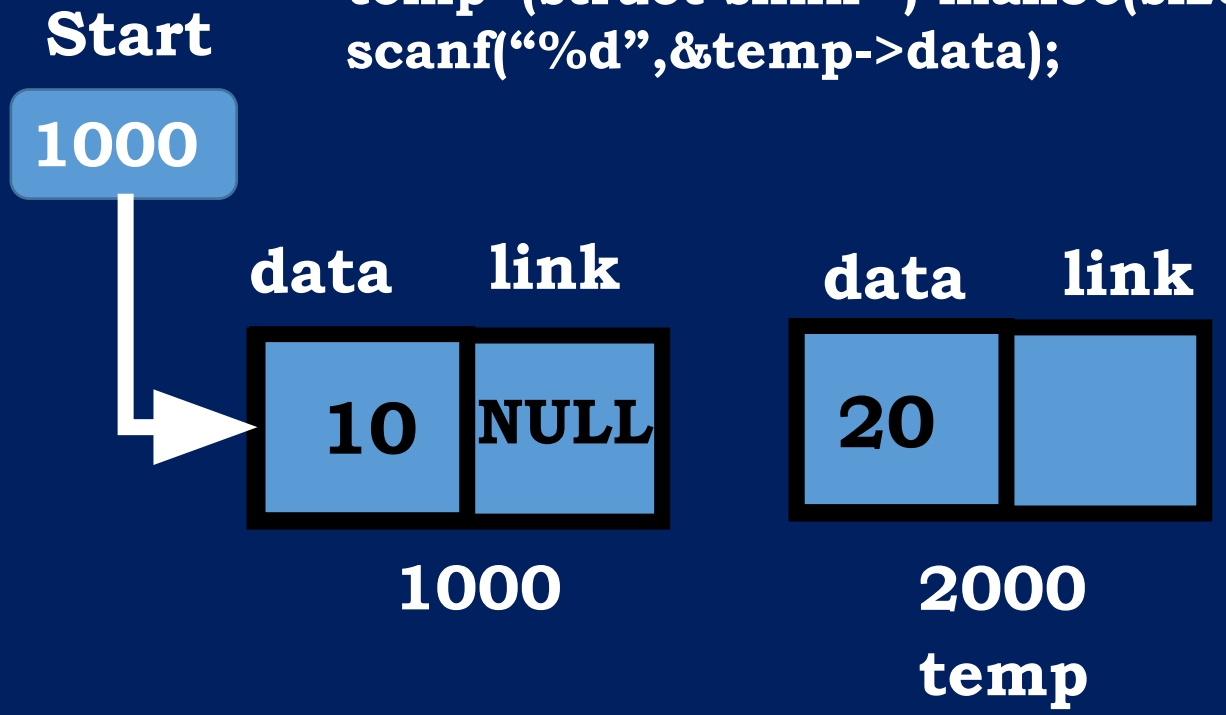
Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
```



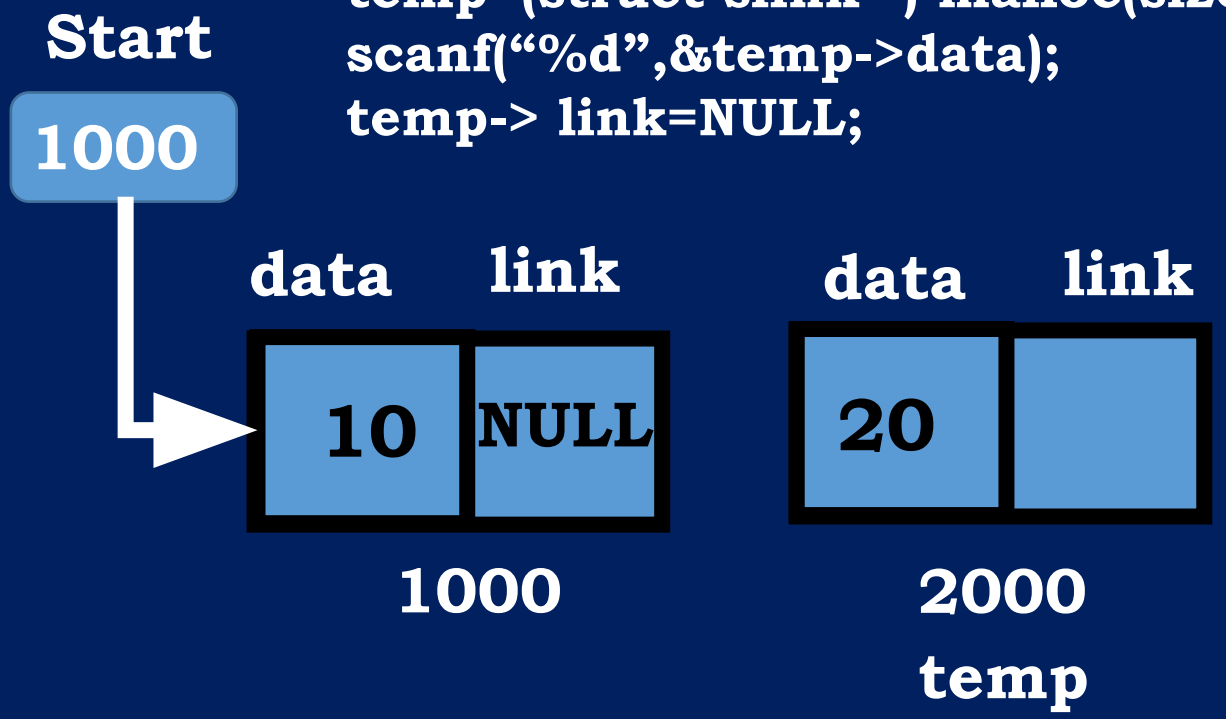
Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
```



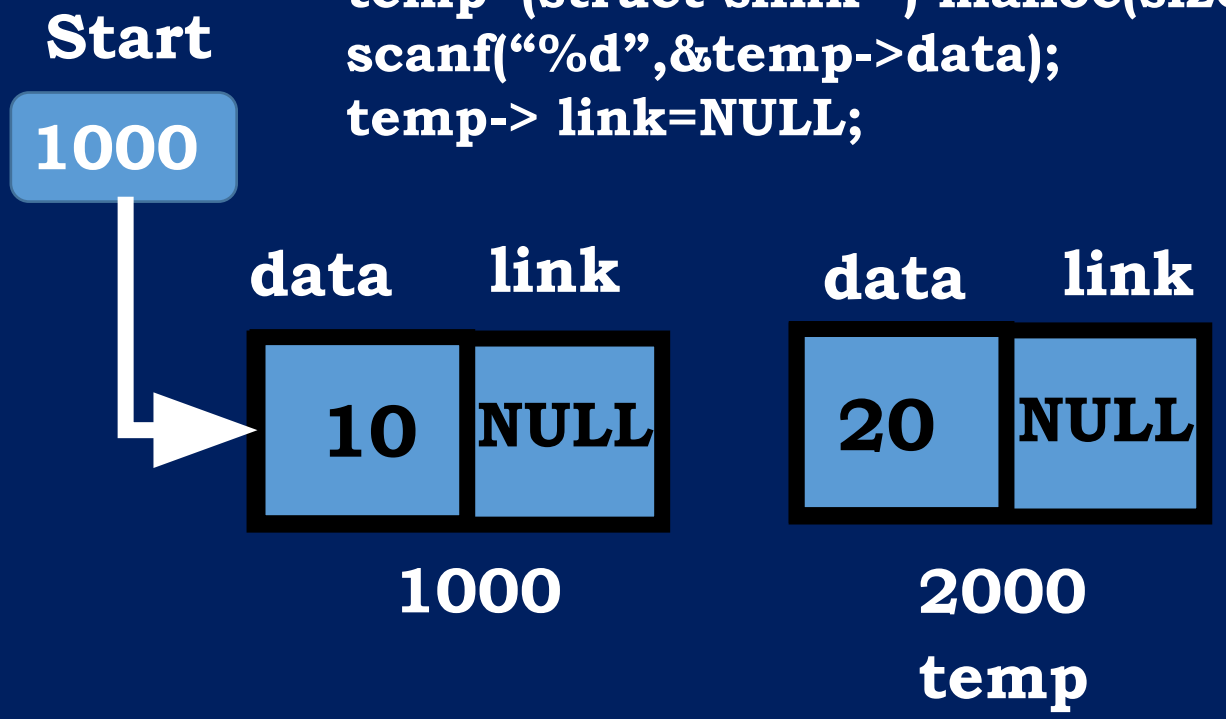
Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```



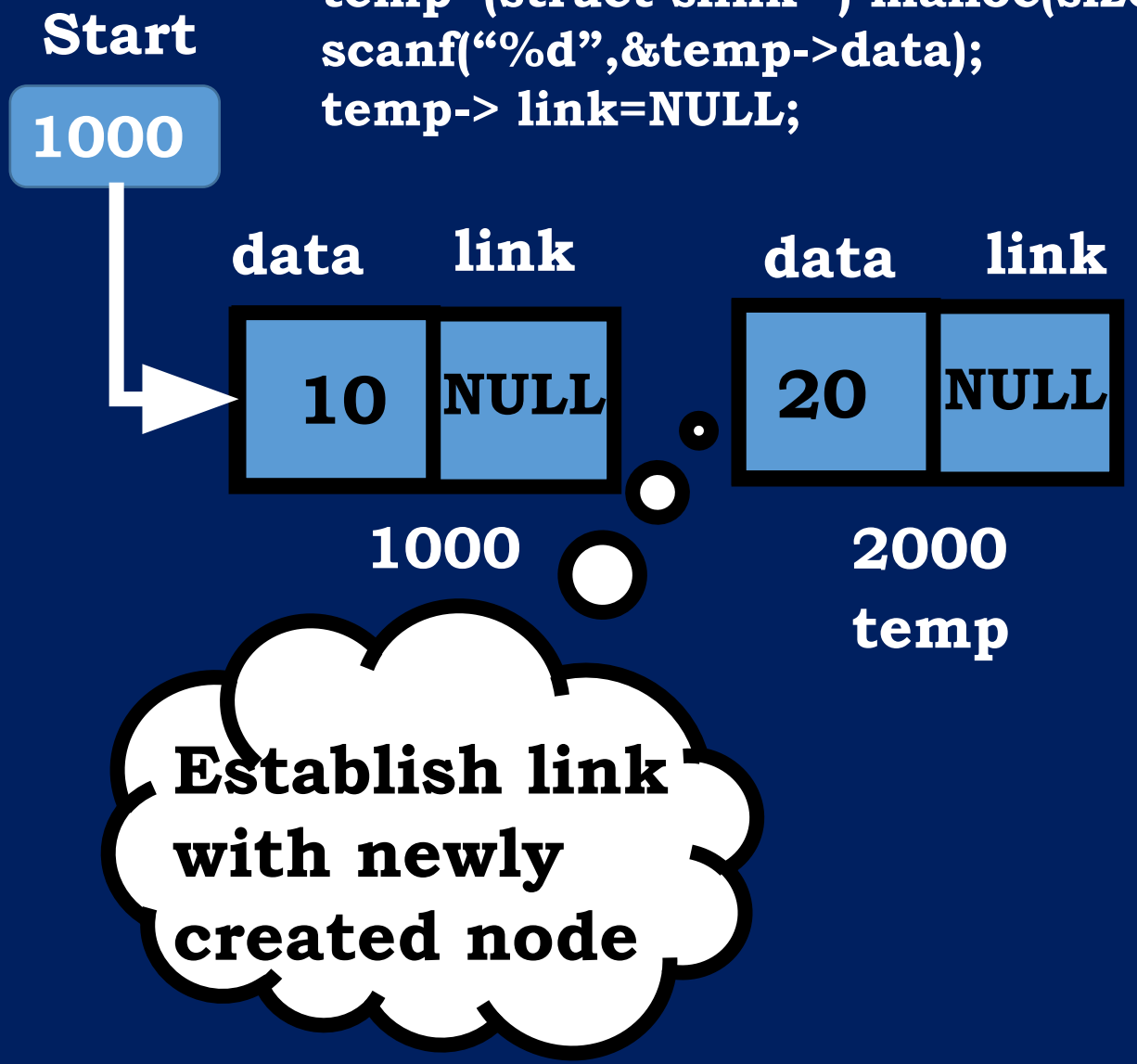
Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```



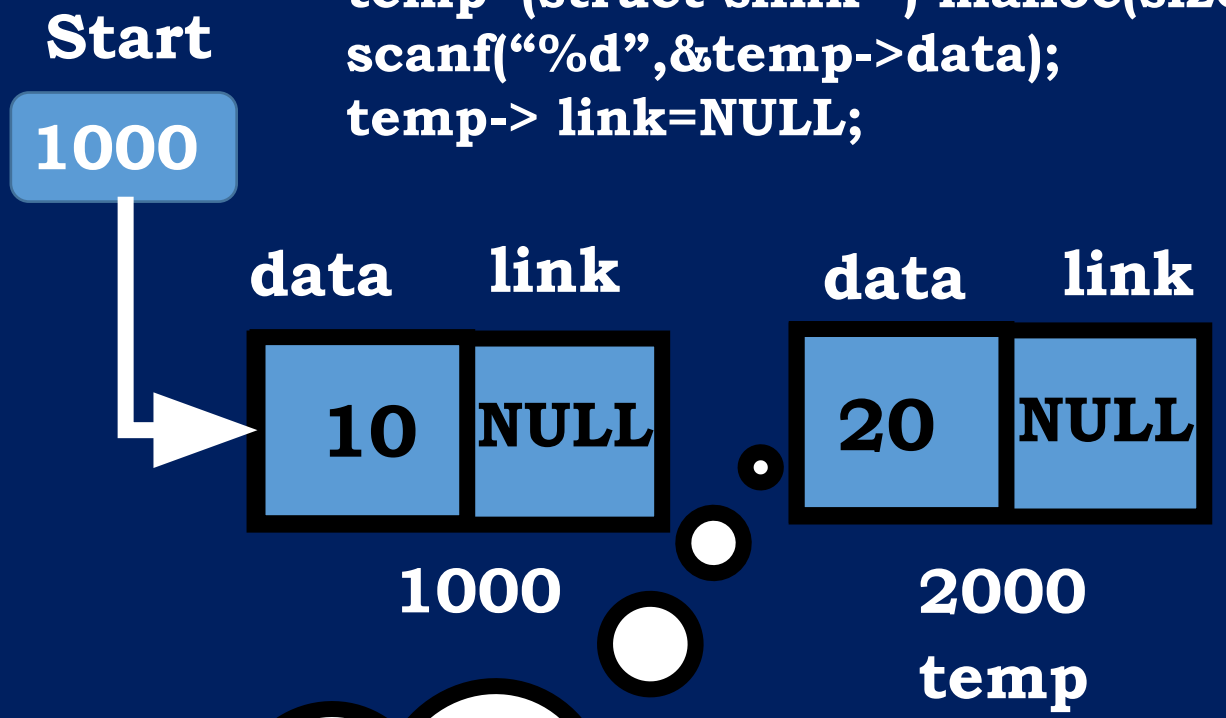
Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```



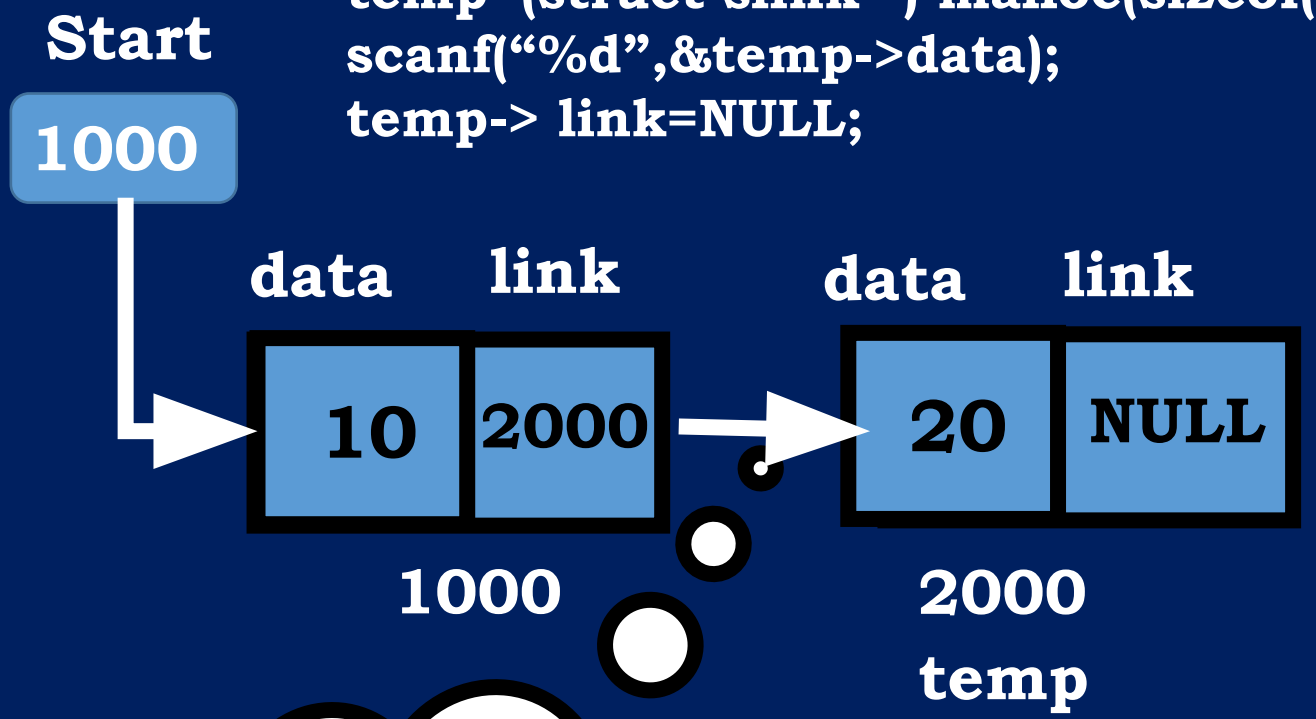
Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```



Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```

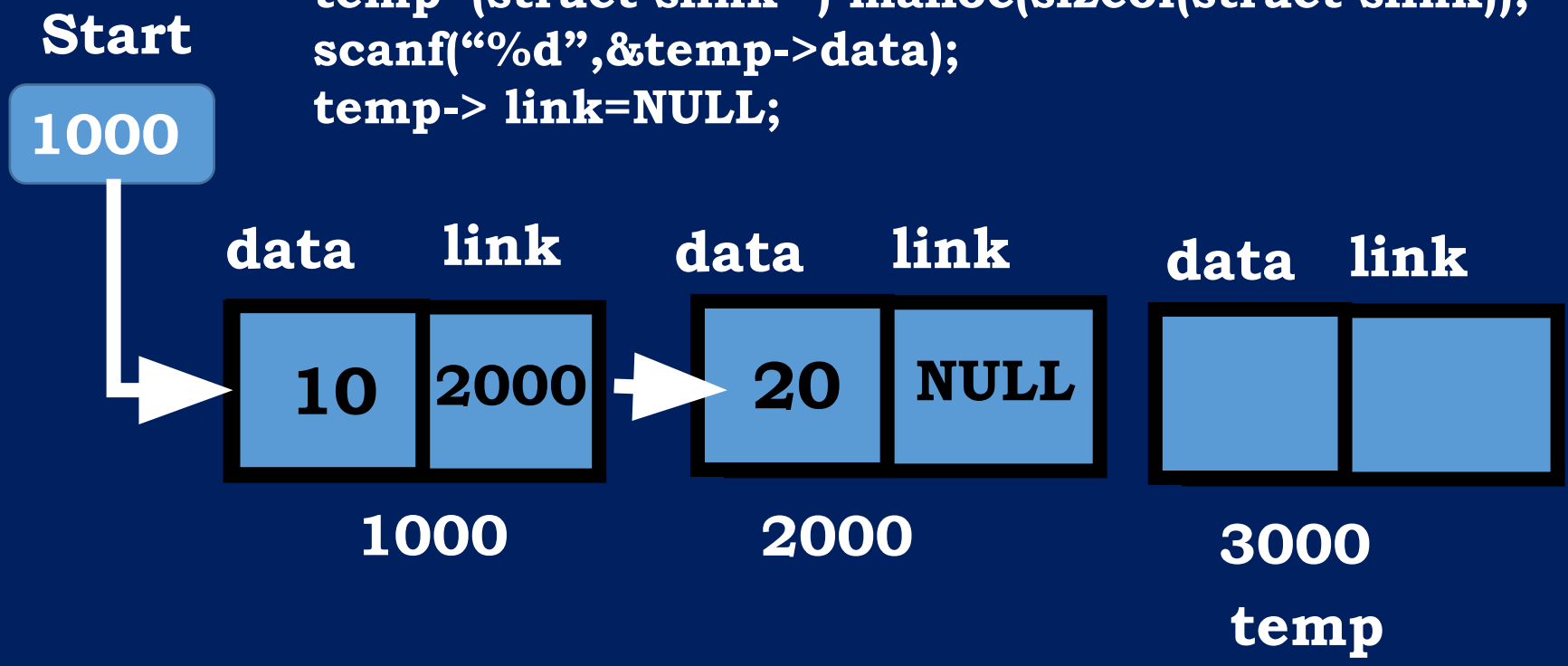


Establish link with newly created node

Start->link=temp

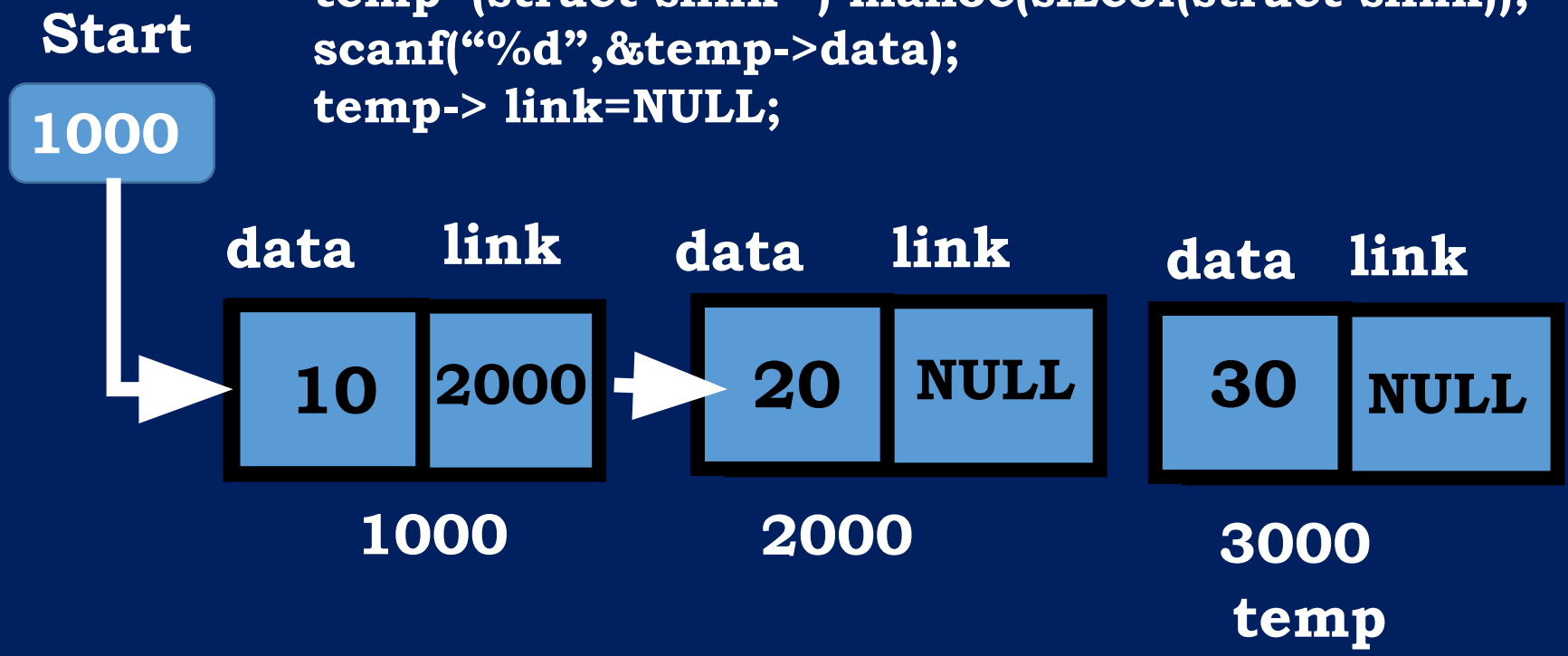
Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```



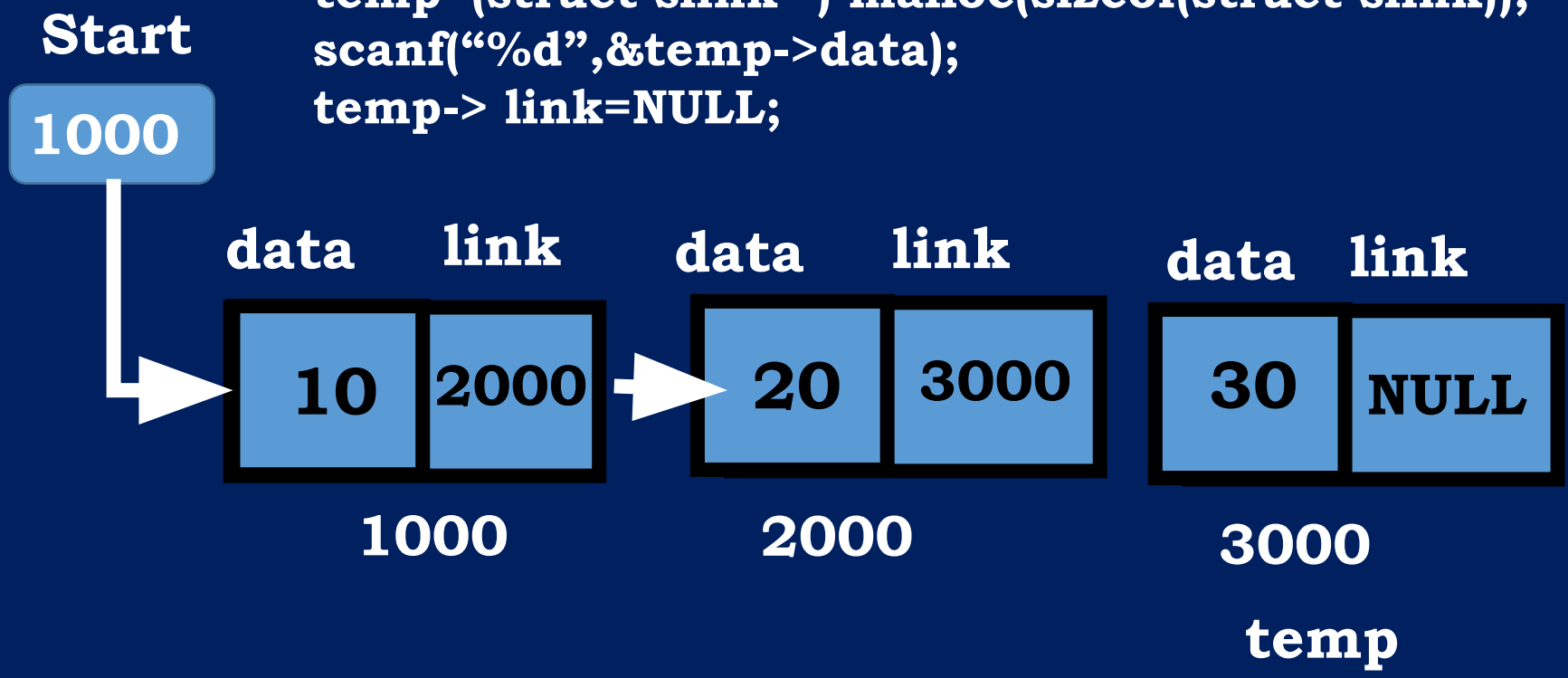
Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```



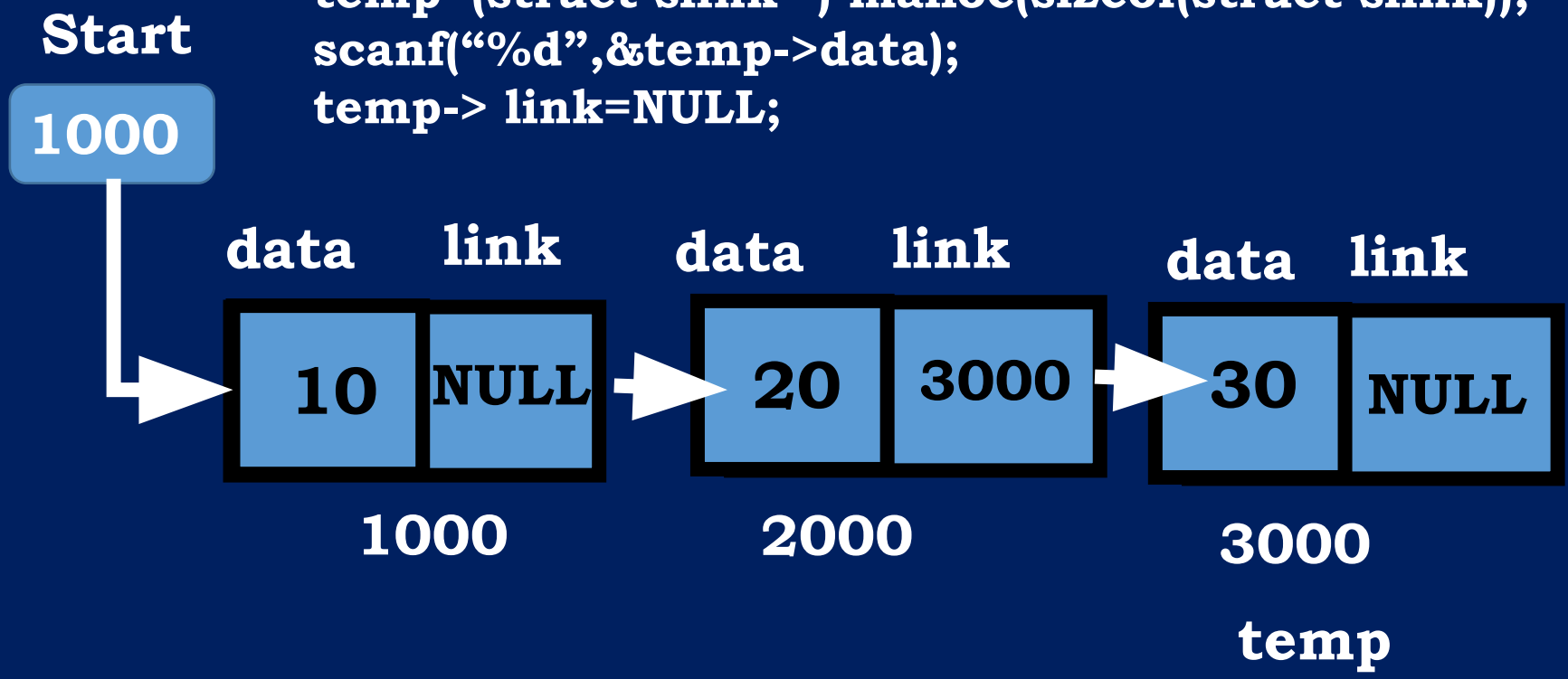
Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```



Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```



Thank You

VIGNAN Institute of Technology and Science

Department of Computer Science and Engineering

Creating a Singly Linked List

Data Structures
B.Tech II year I Sem

by
S Akshara Reddy
Assistant Professor
Email id: akshara.revelly@gmail.com
Blog: http://aksharatechnicalstuff.blogspot.com/

Creating a Linked List

- **Steps to be followed while creating a list**

1. **Allocate block of memory for node & Divide it into data part and link part**
2. **Accept data part and assign Null to link**
3. **Establish links to newly created node.**

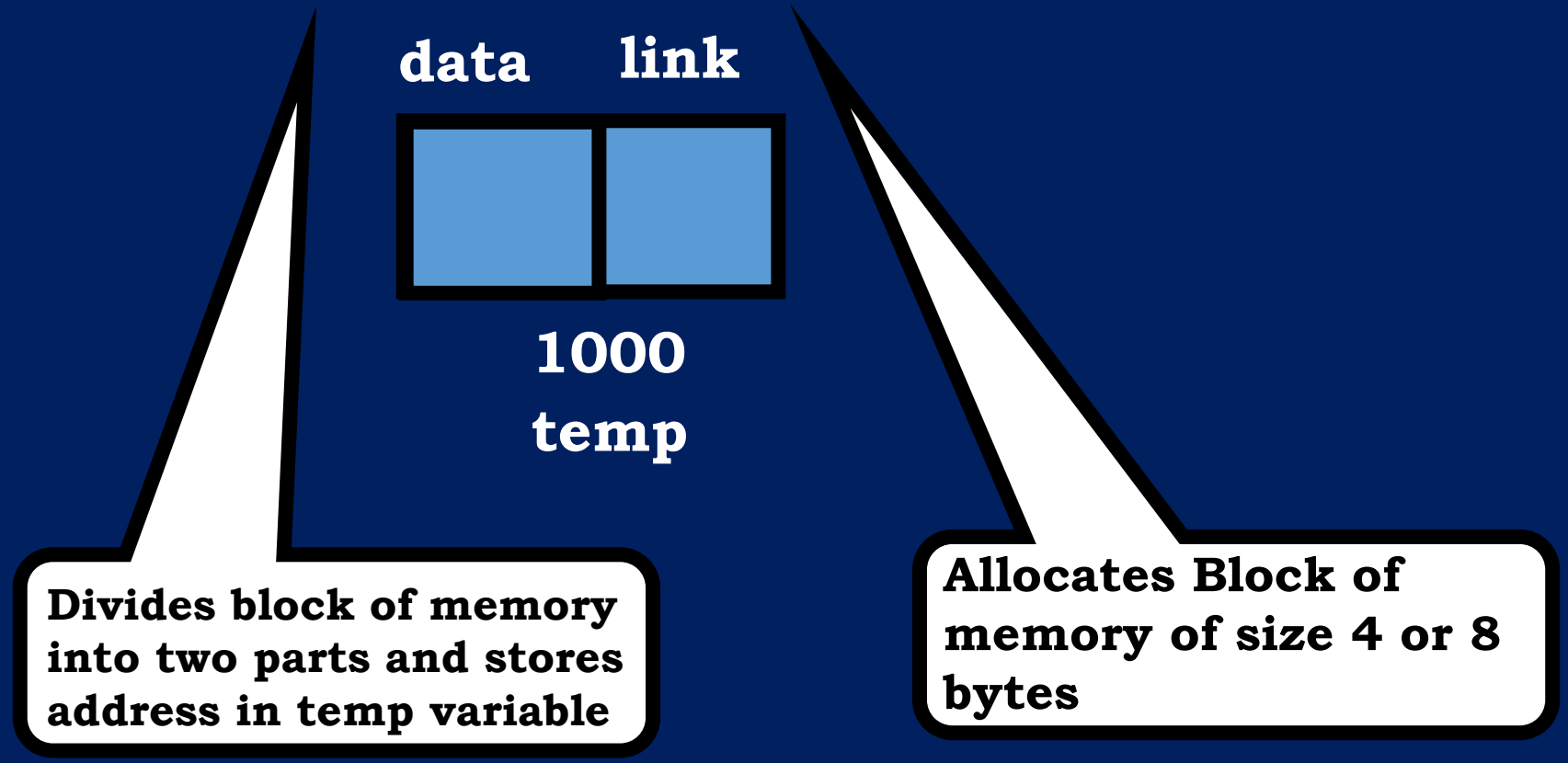
Repeat Step1 to step3 for creating nodes.

Creating a Linked List

Allocate block of memory for node &

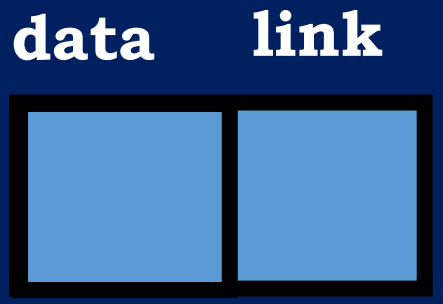
Divide it into data part and link part

```
temp=(struct slink *) malloc(sizeof(struct slink));
```

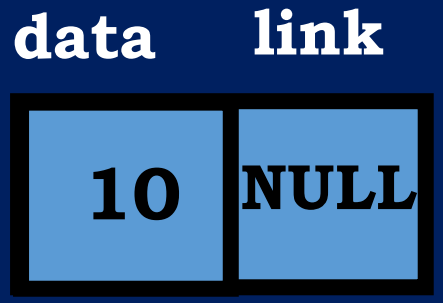


Creating a Linked List

Accept data part and assign Null to link

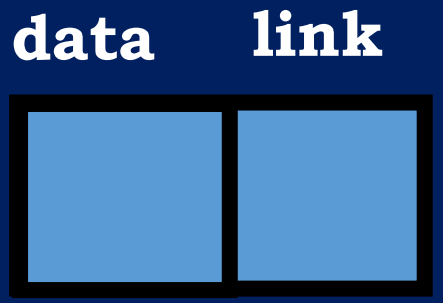



```
scanf("%d",&temp->data);
temp->link=NULL;
```



Creating a Linked List

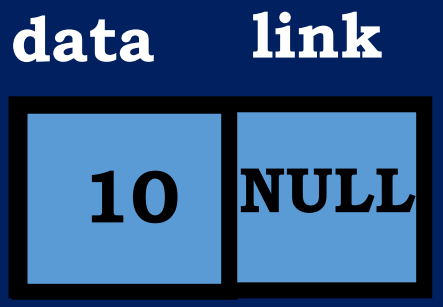
Accept data part and assign Null to link



1000

temp

```
scanf("%d",&temp->data);
temp->link=NULL;
```



1000

temp

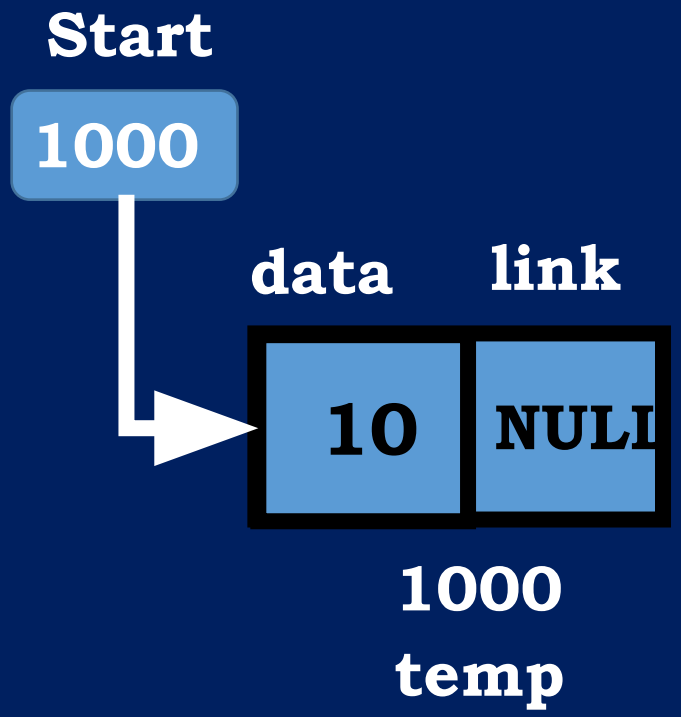
If it is first node in the list then assign address of that node to start pointer

start = temp

Creating a Linked List

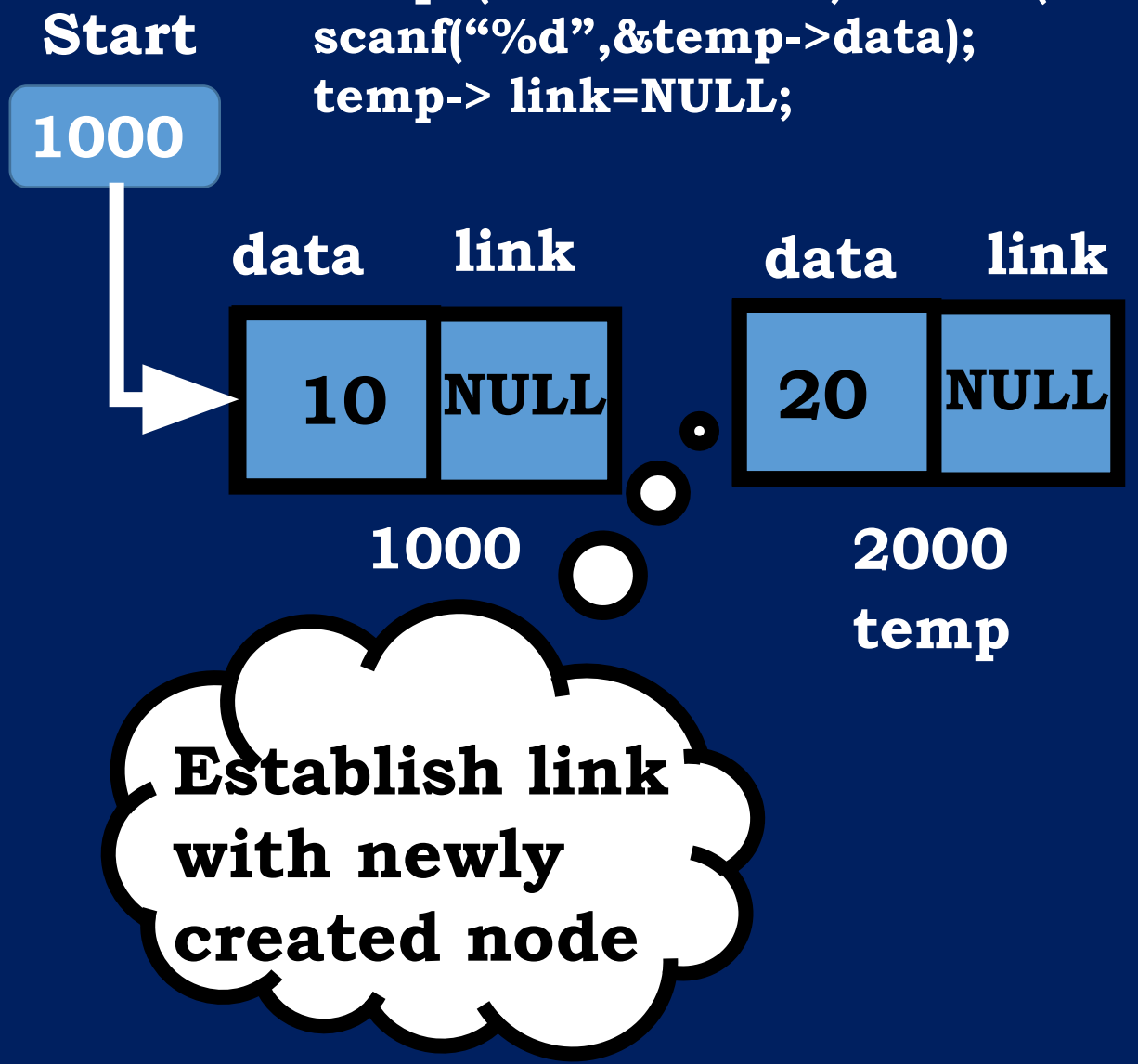
```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp->link = NULL;
```

```
if (start == NULL)
    start = temp;
```



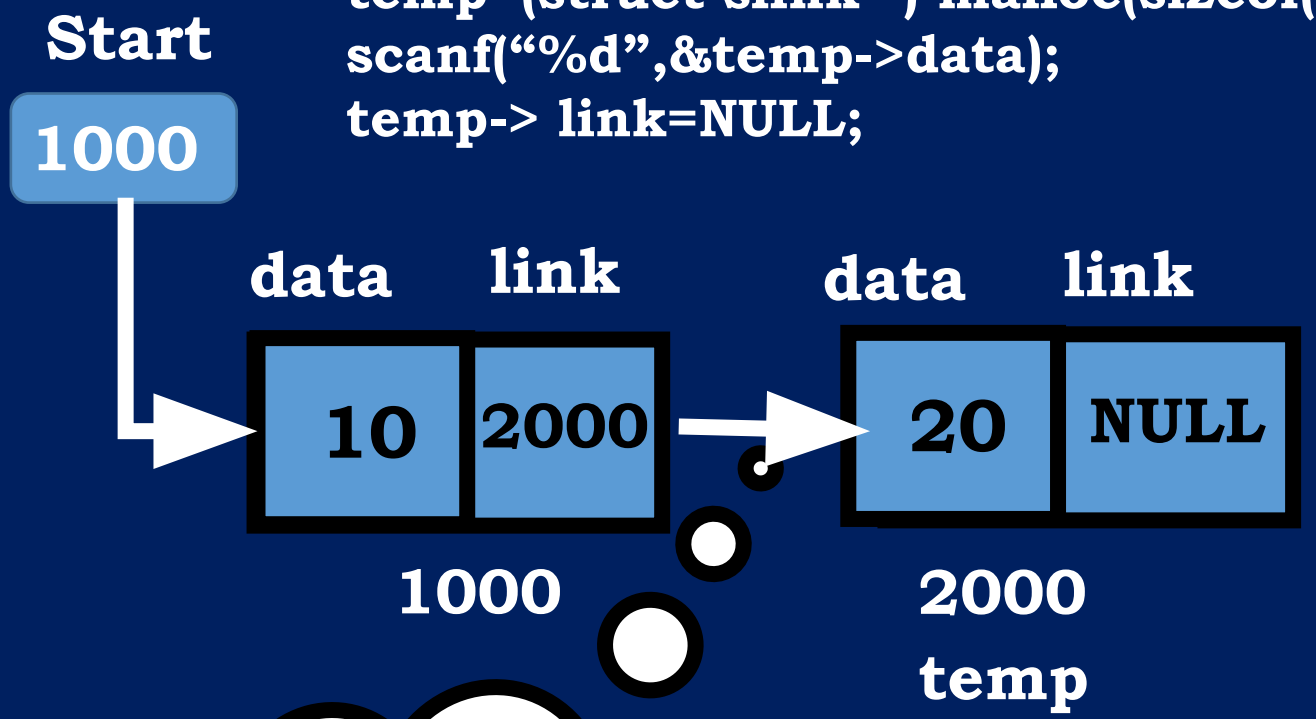
Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```



Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```

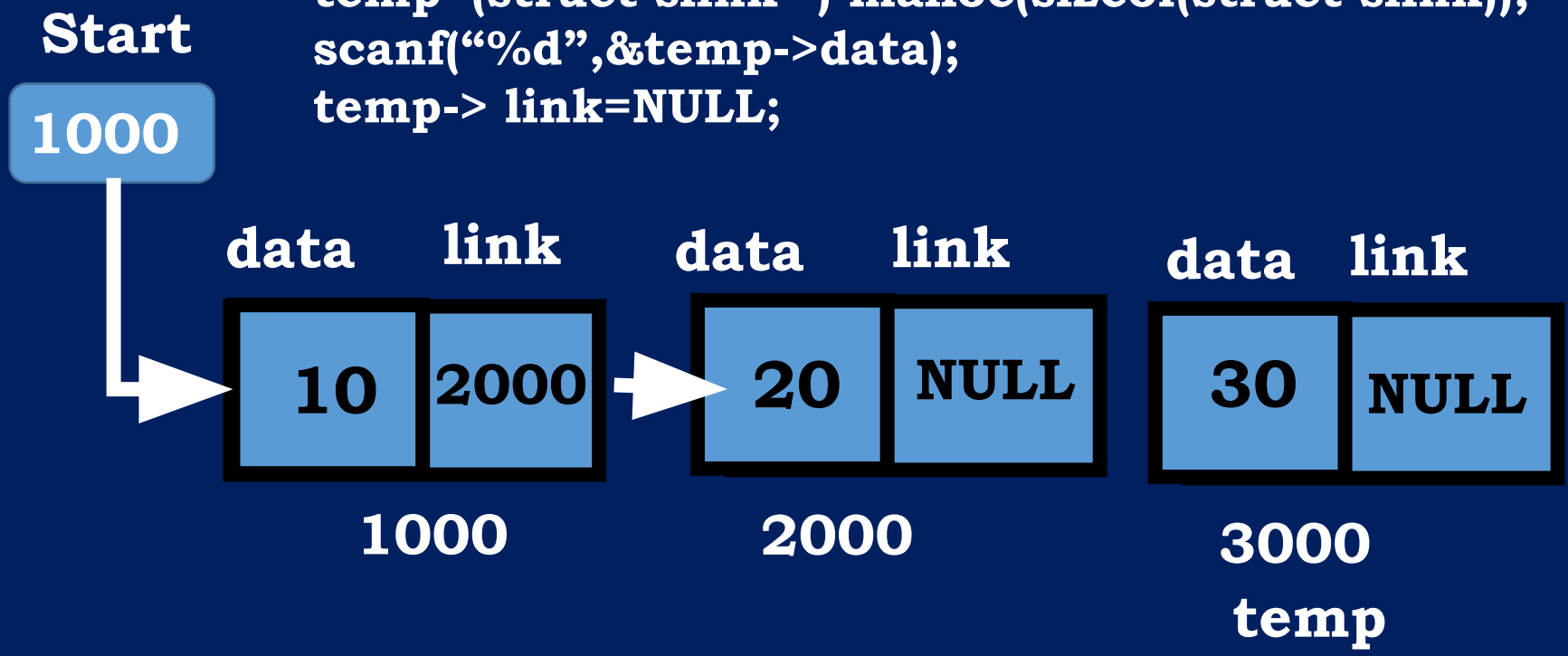


Establish link with newly created node

Start->link=temp

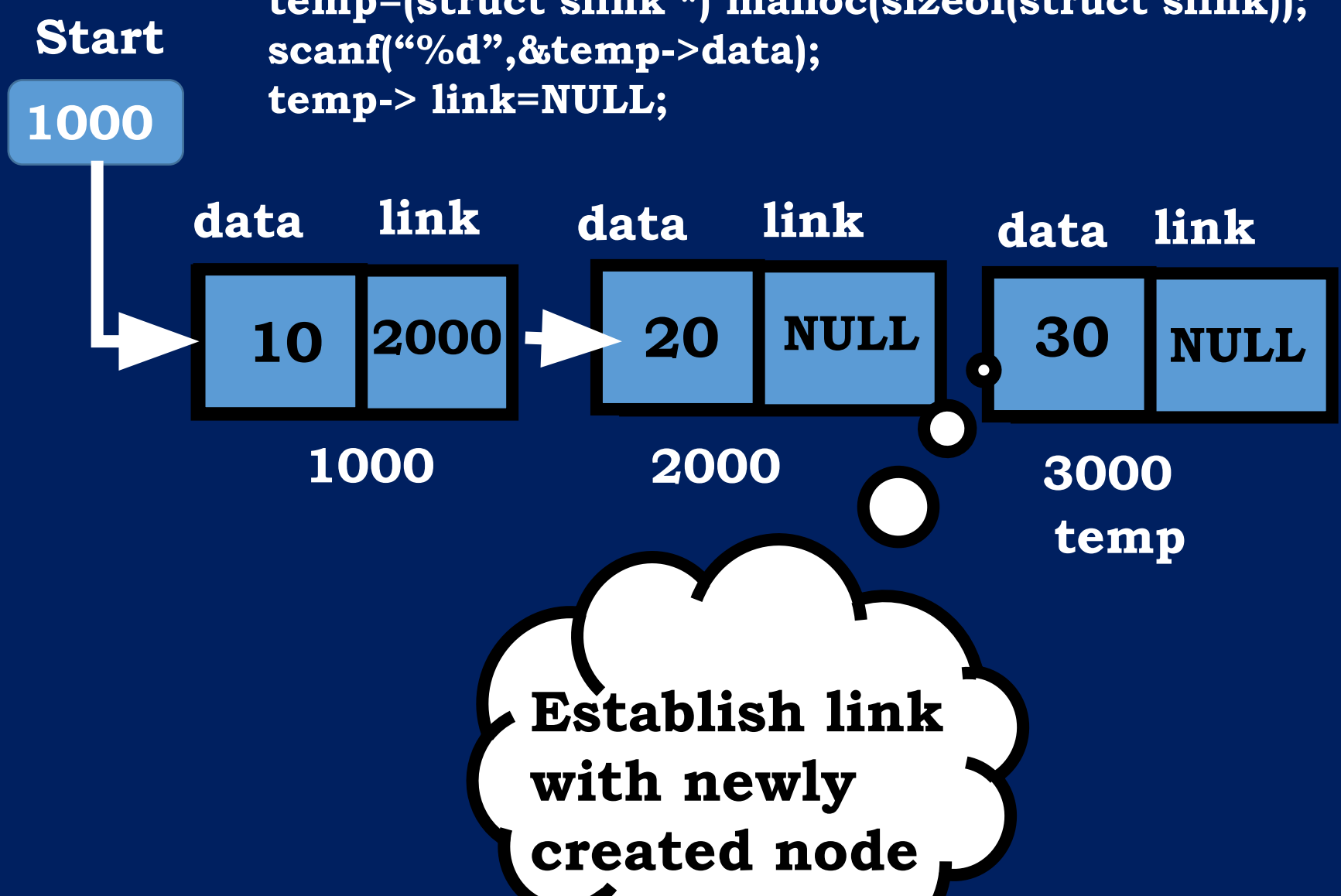
Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```

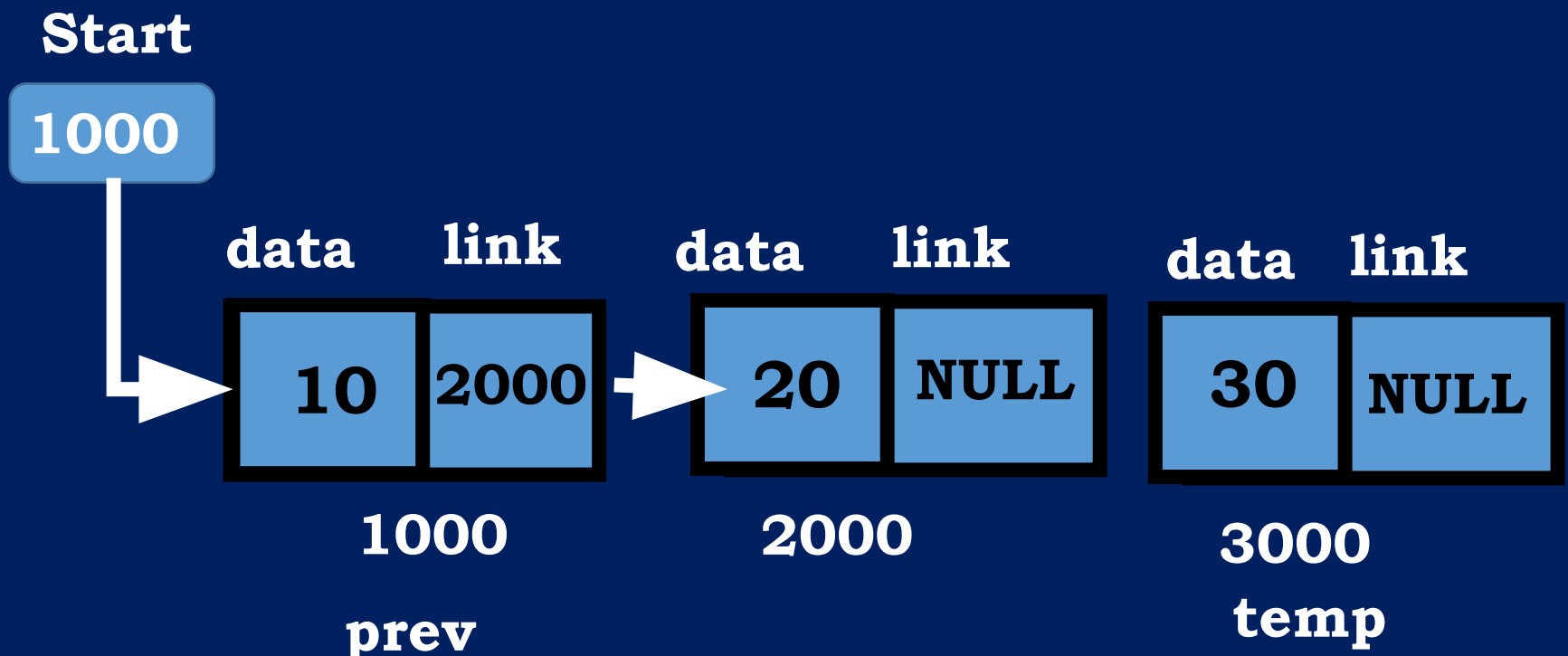


Creating a Linked List

```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```

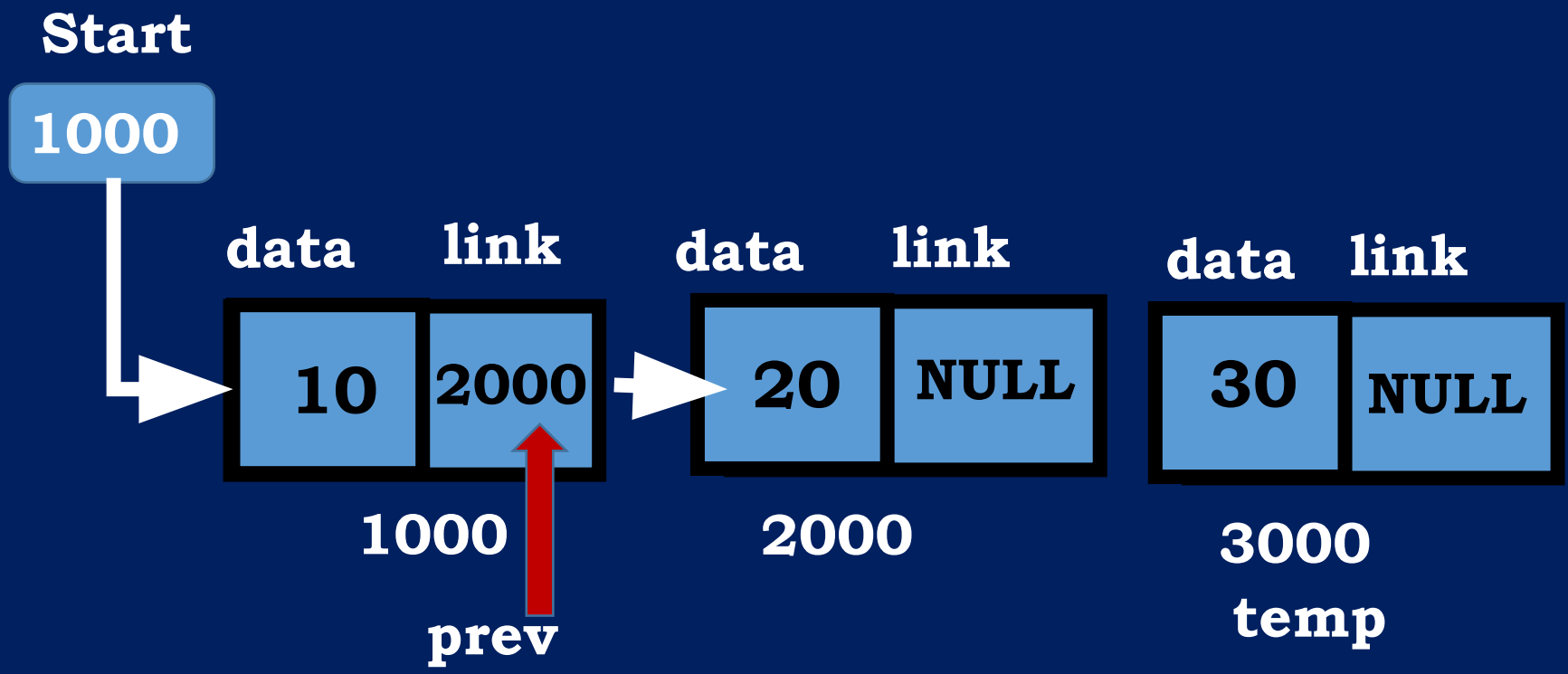


Creating a Linked List



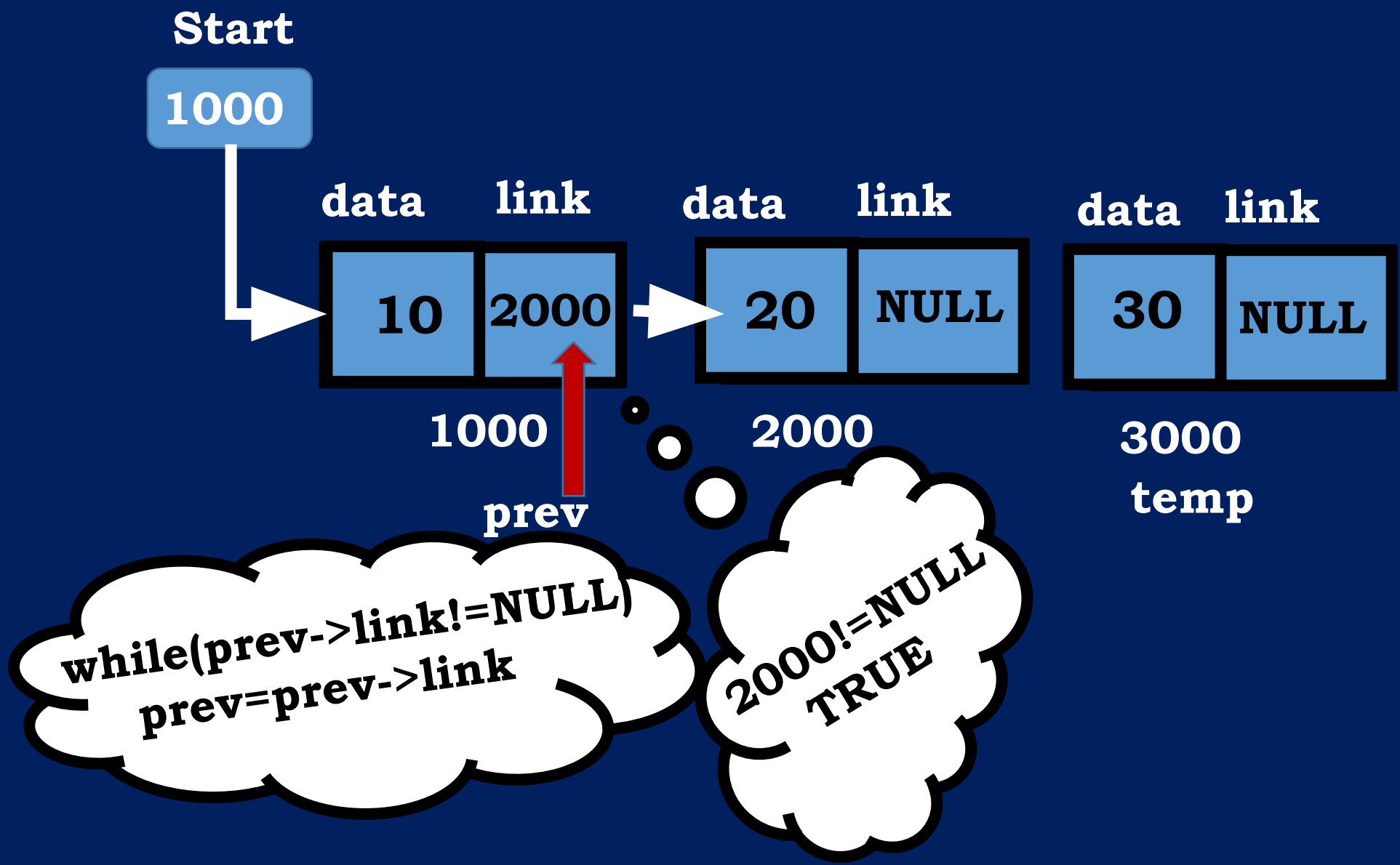
prev=start;

Creating a Linked List

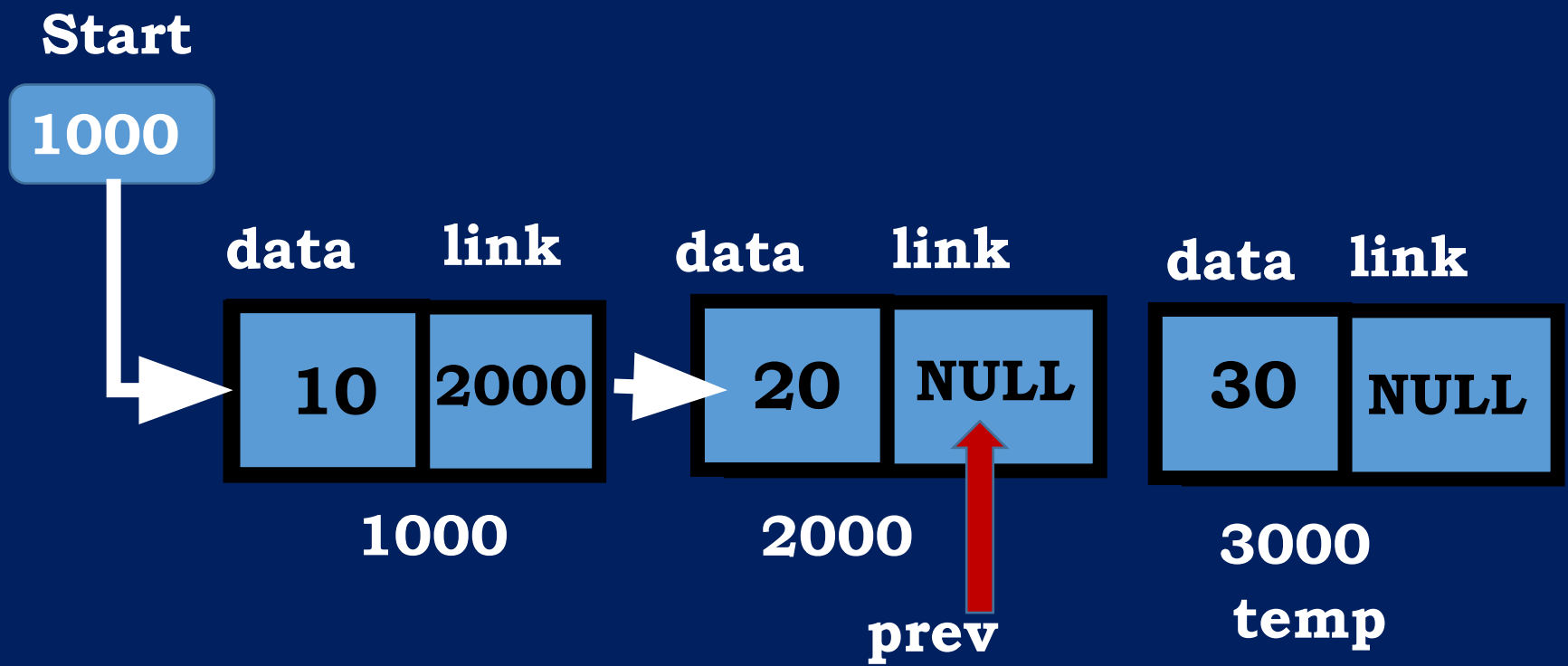



```
while(prev->link!=NULL)
    prev=prev->link
```

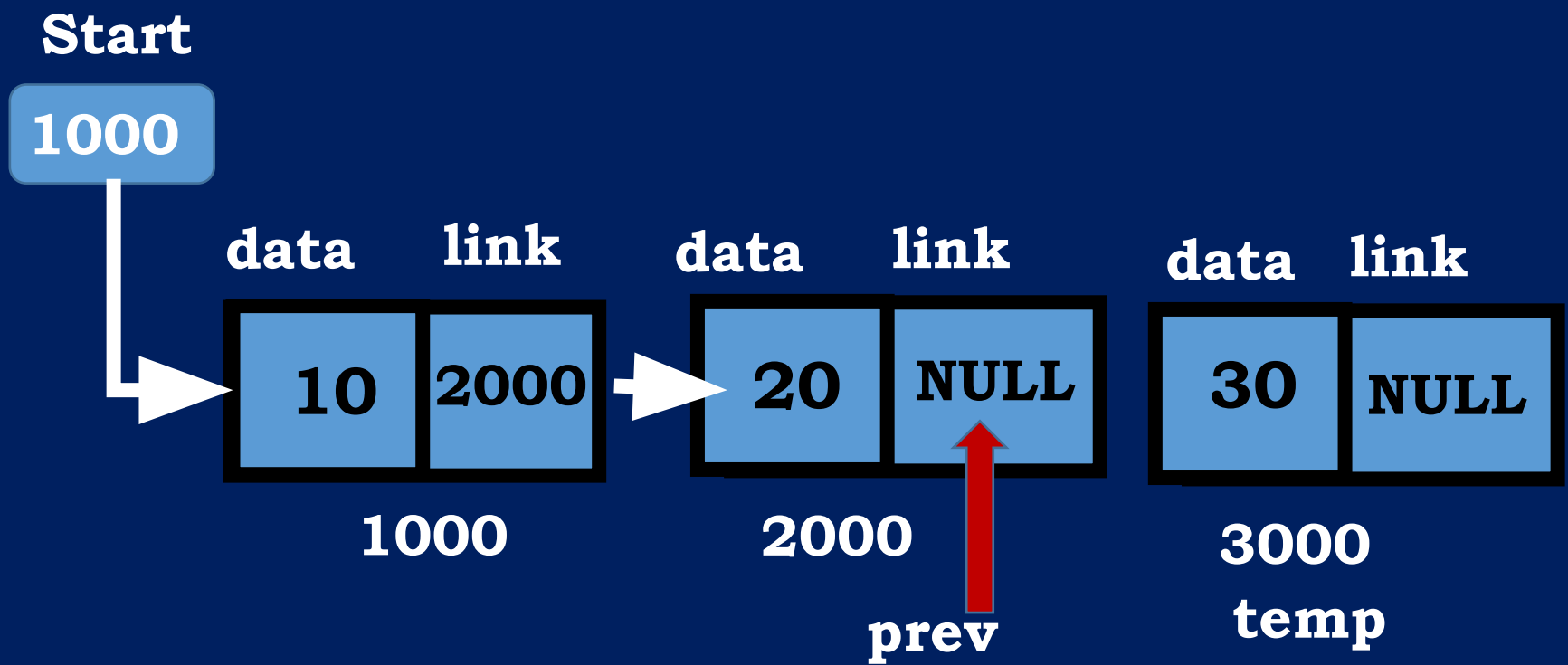
Creating a Linked List



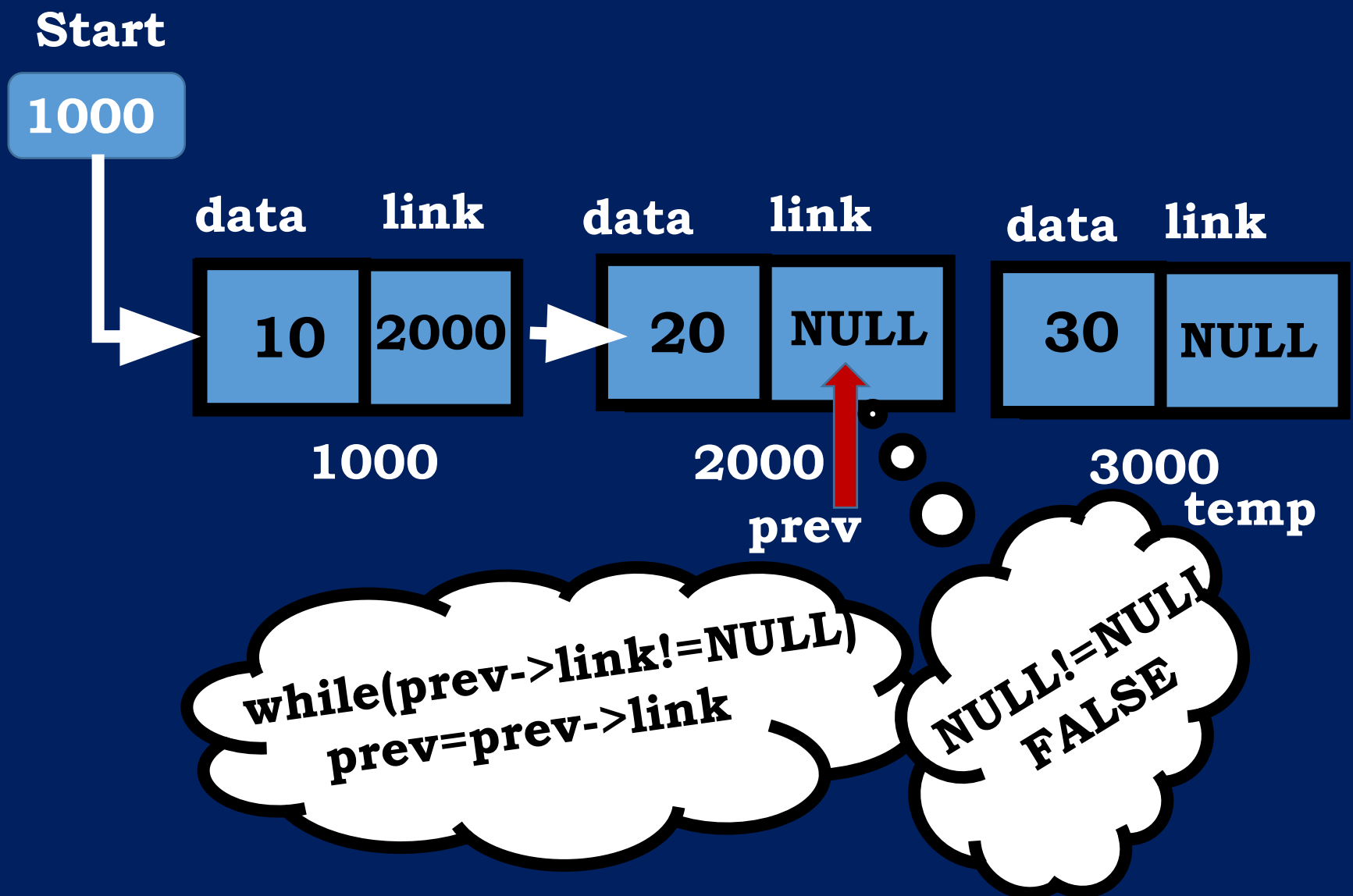
Creating a Linked List



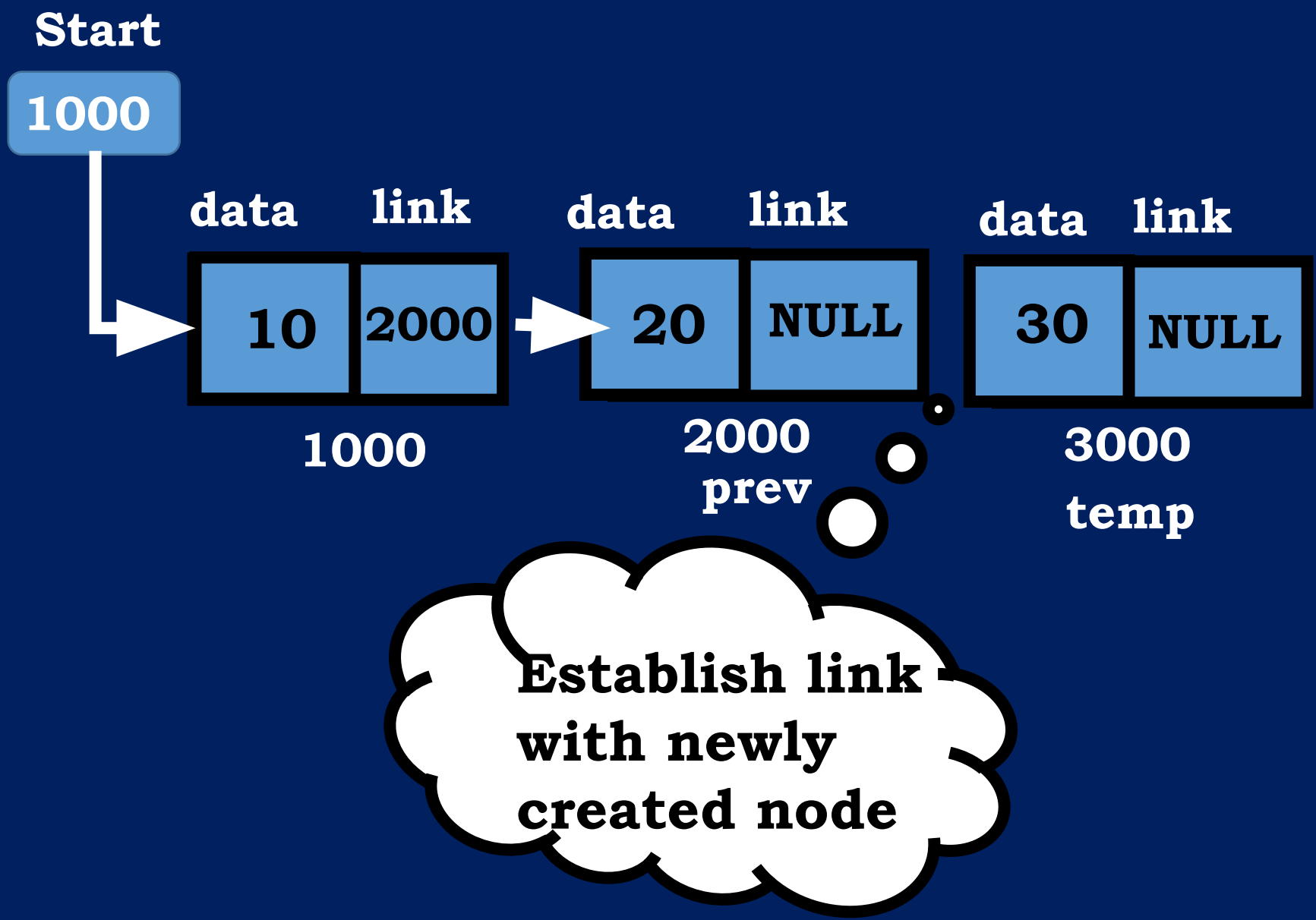
Creating a Linked List



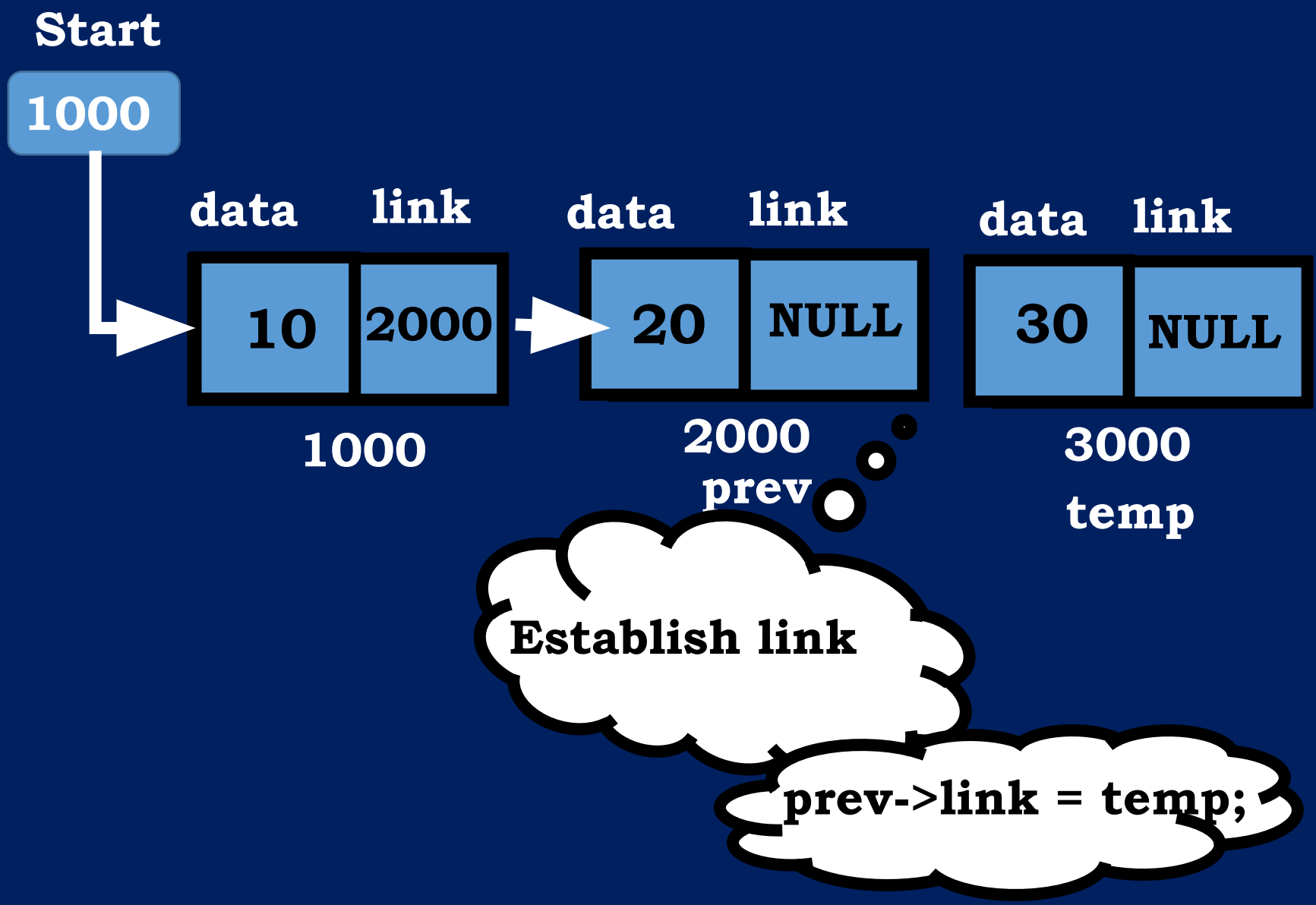
Creating a Linked List



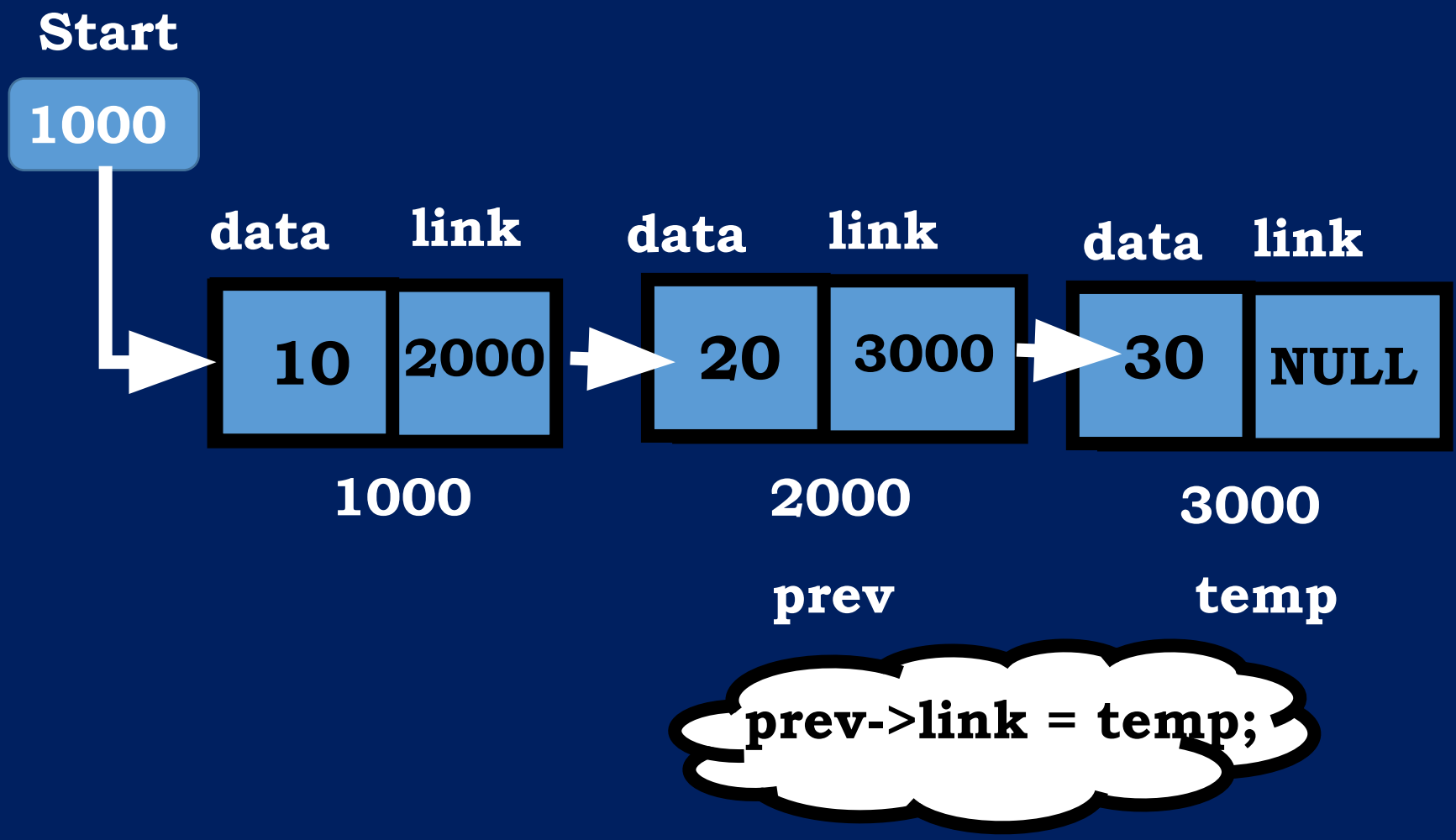
Creating a Linked List



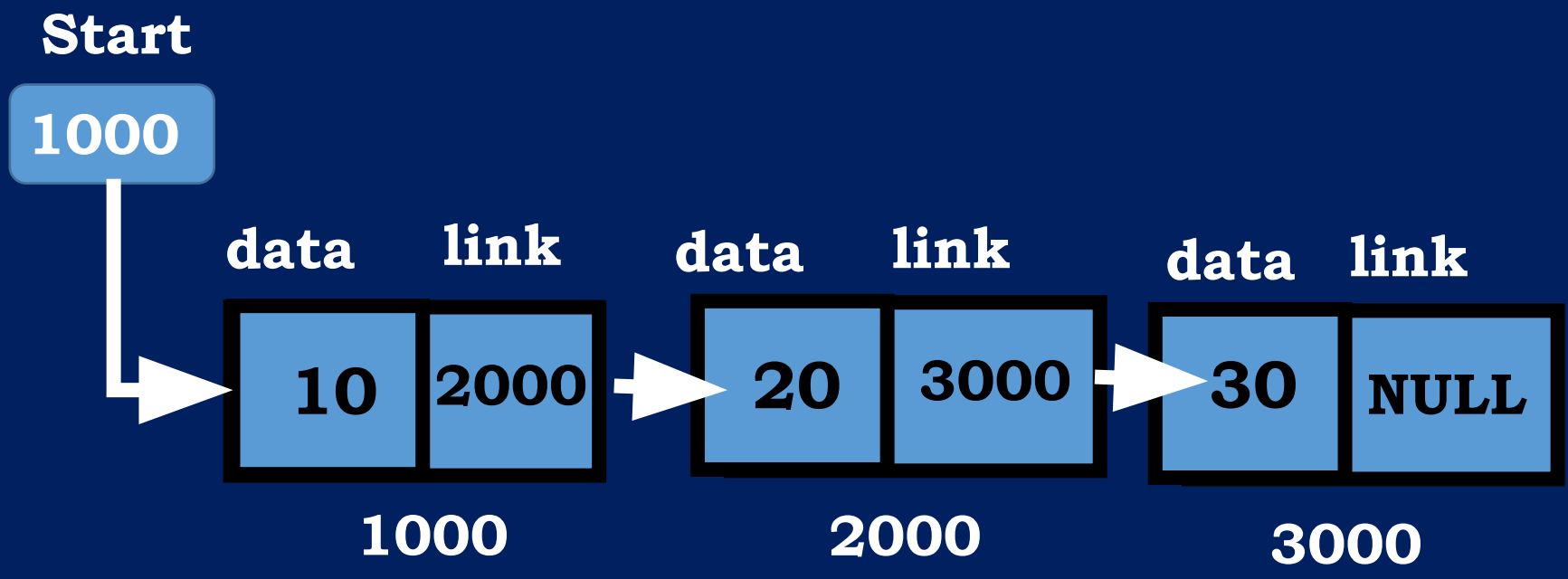
Creating a Linked List



Creating a Linked List



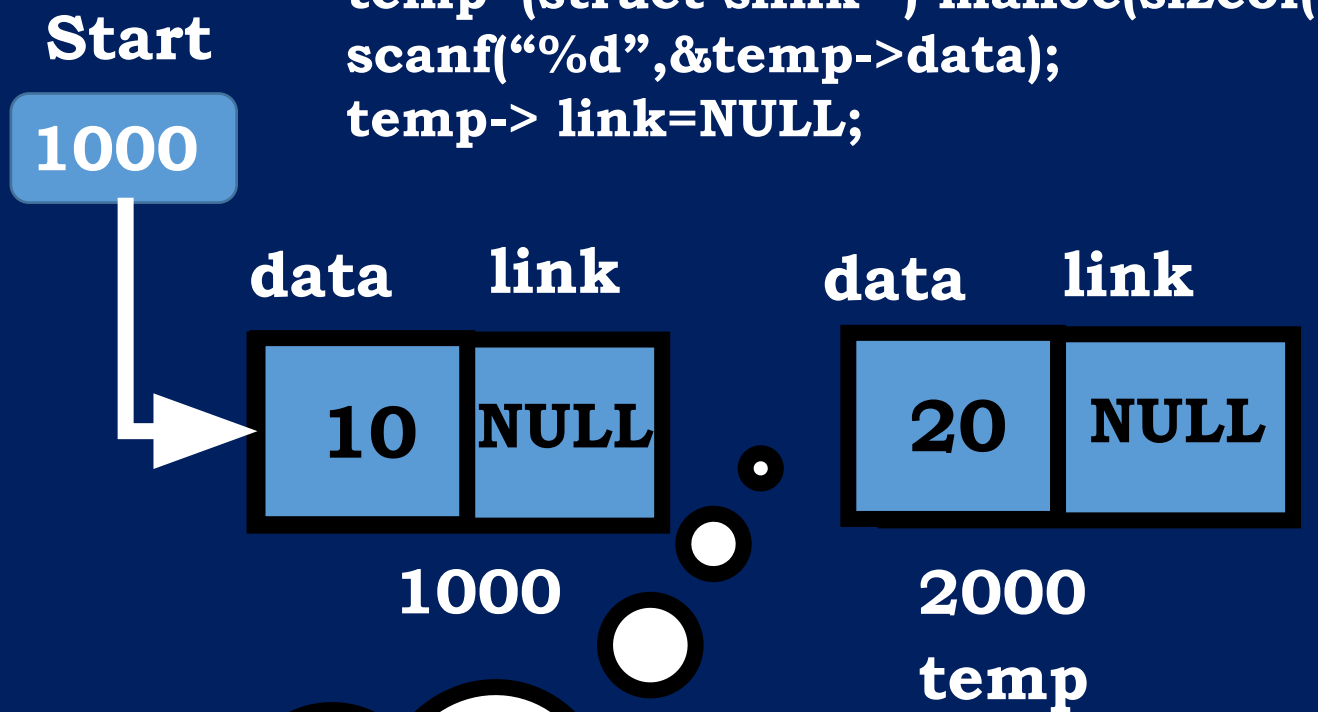
Creating a Linked List



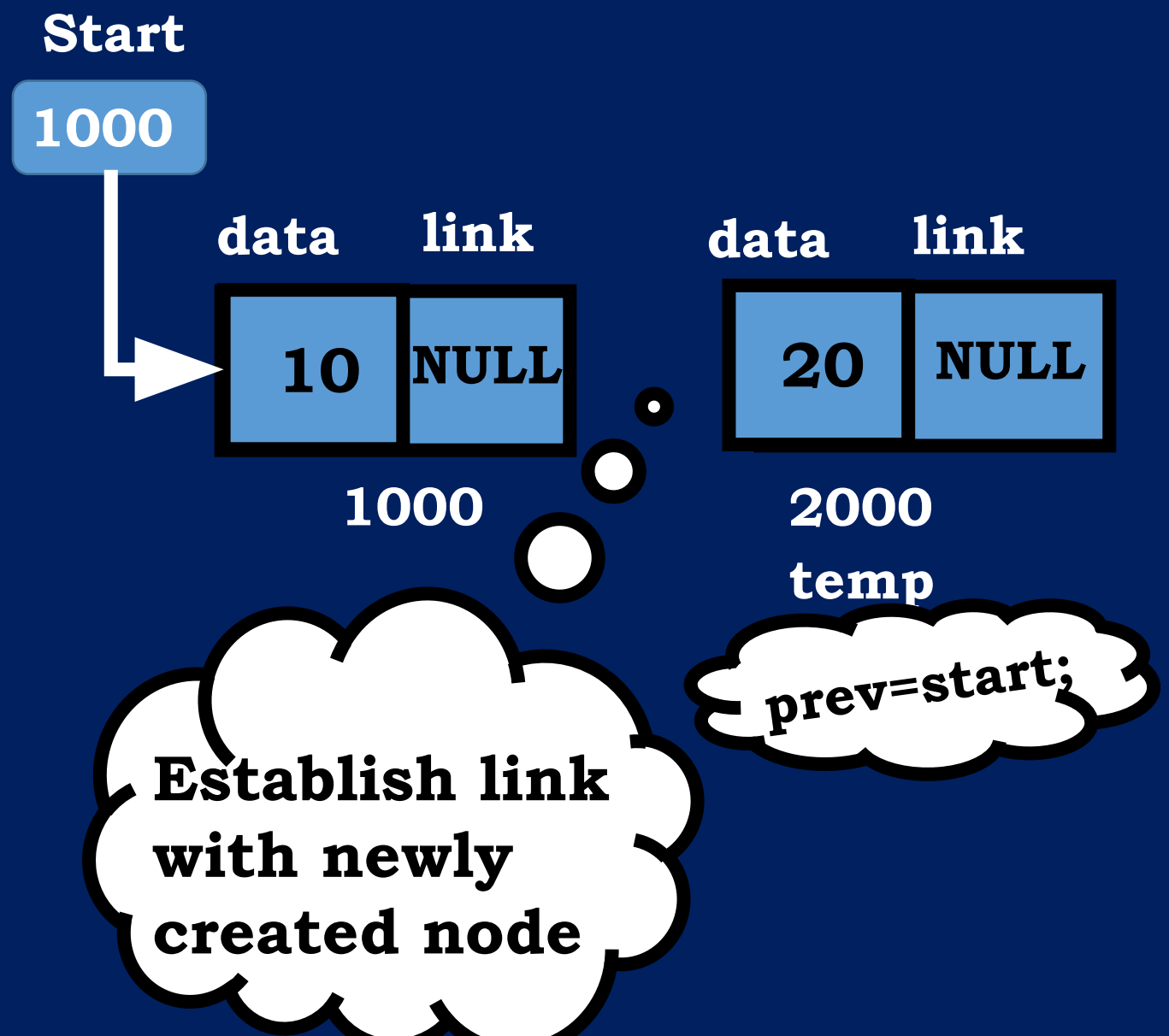
The steps which we have followed for adding third node can also be used for adding second node

Creating a Linked List

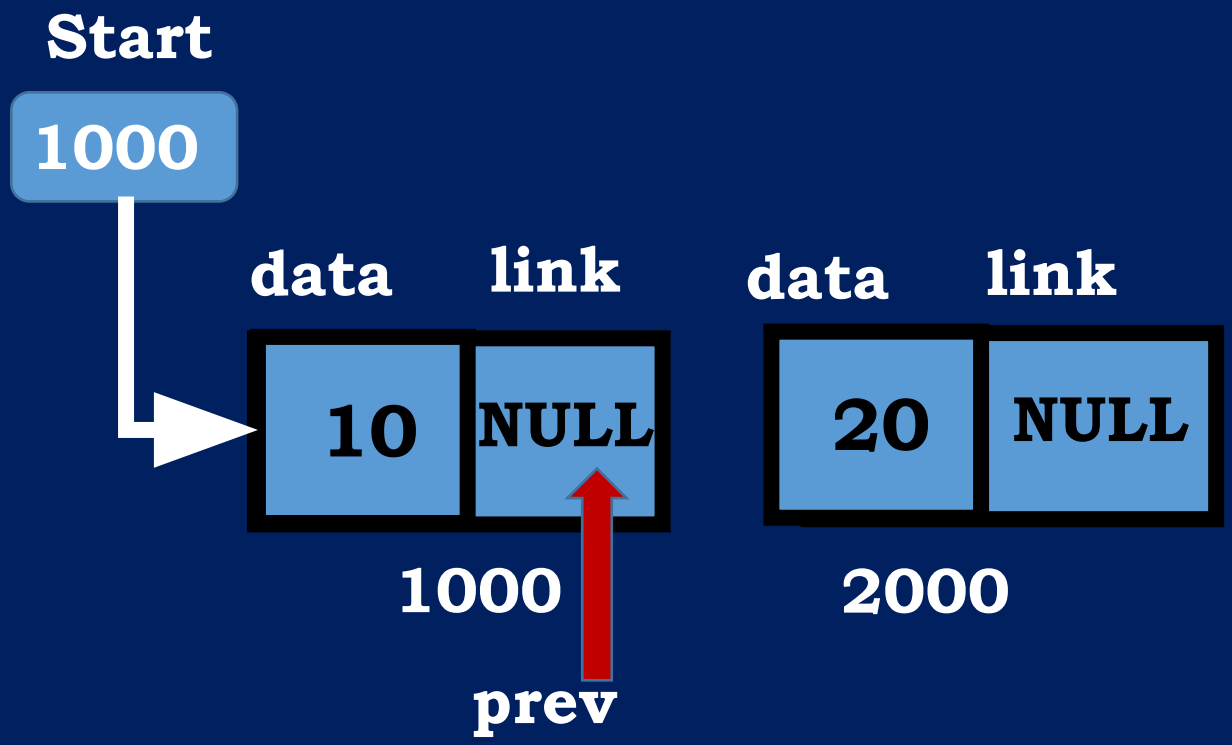
```
temp=(struct slink *) malloc(sizeof(struct slink));
scanf("%d",&temp->data);
temp-> link=NULL;
```



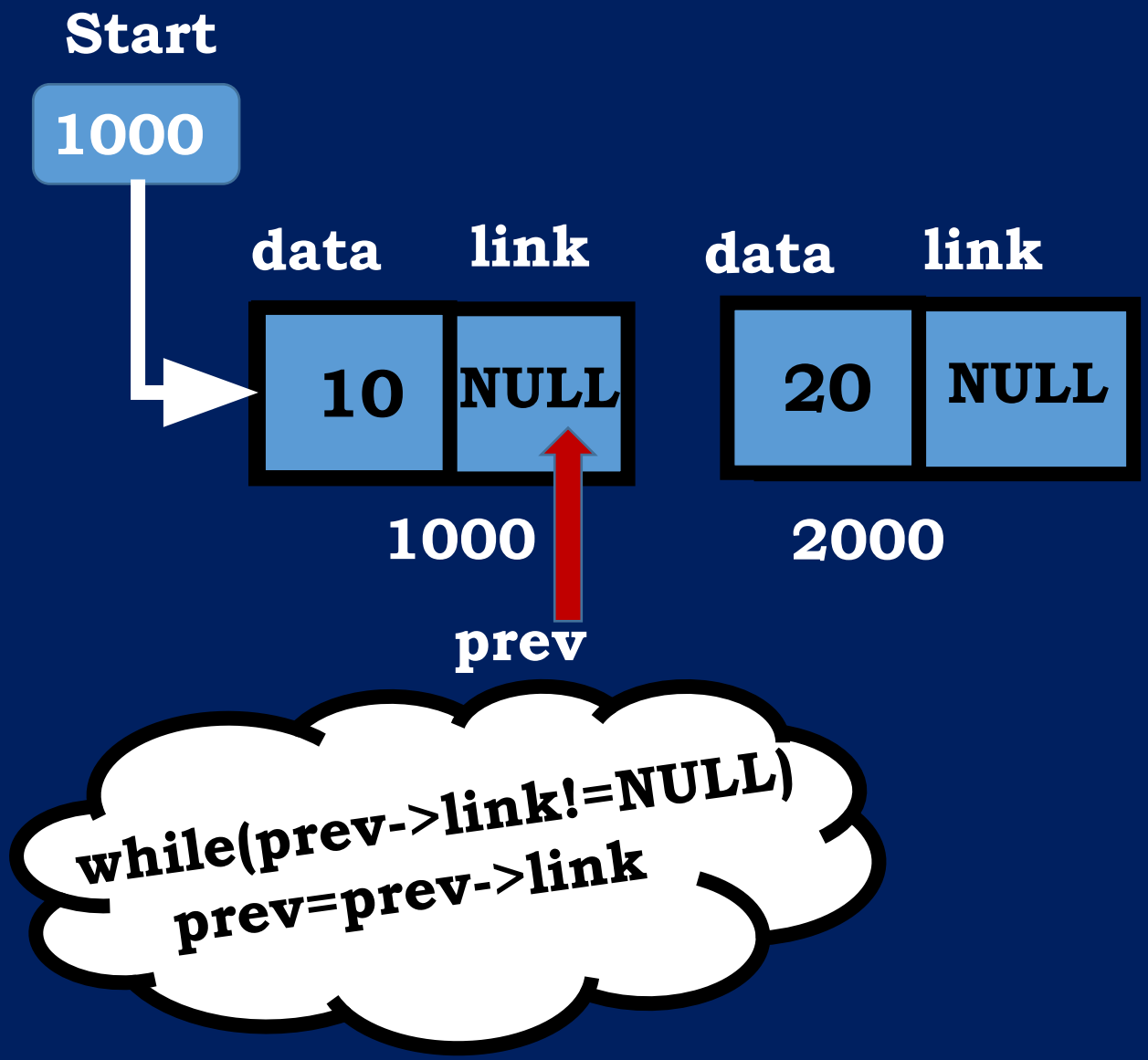
Creating a Linked List



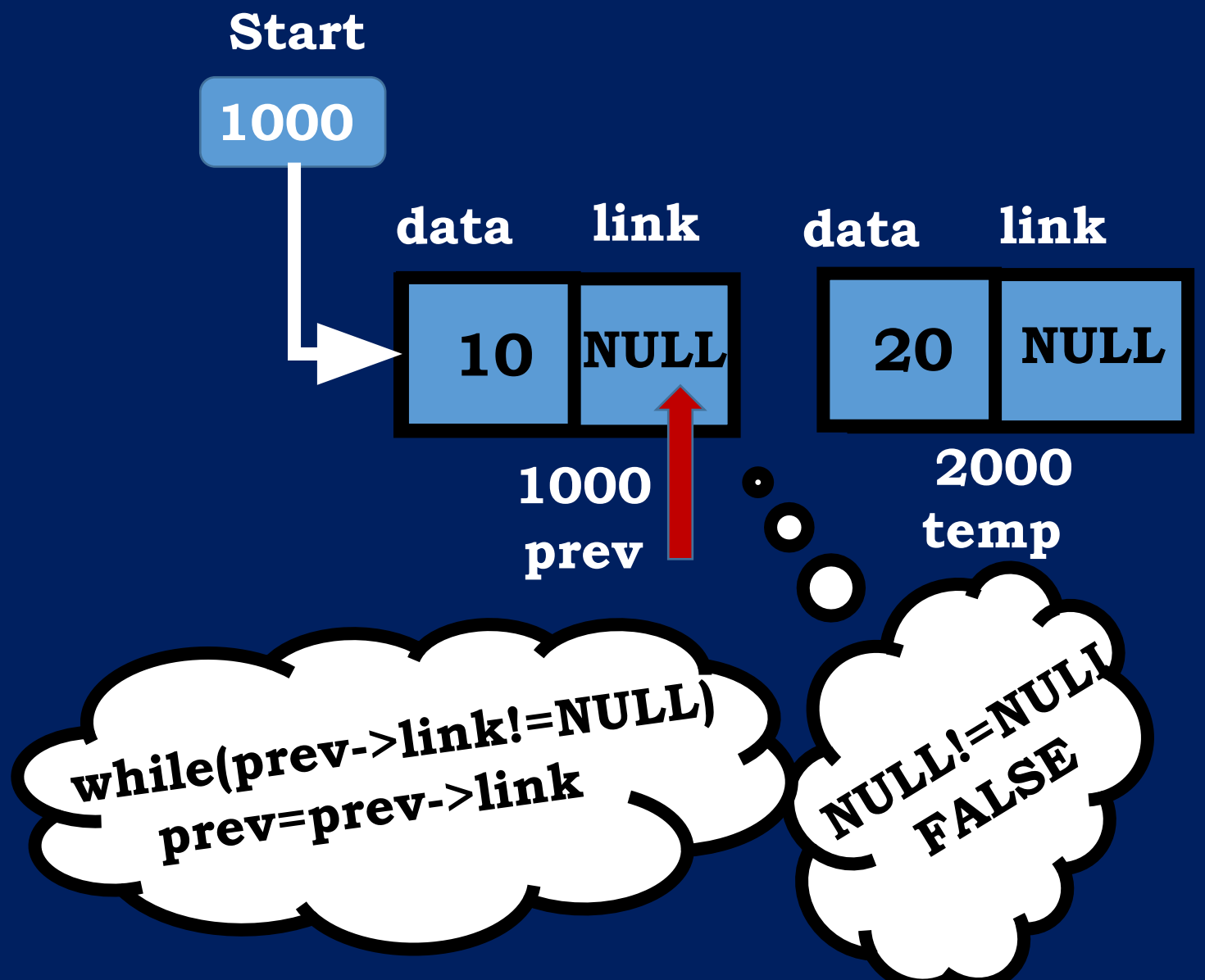
Creating a Linked List



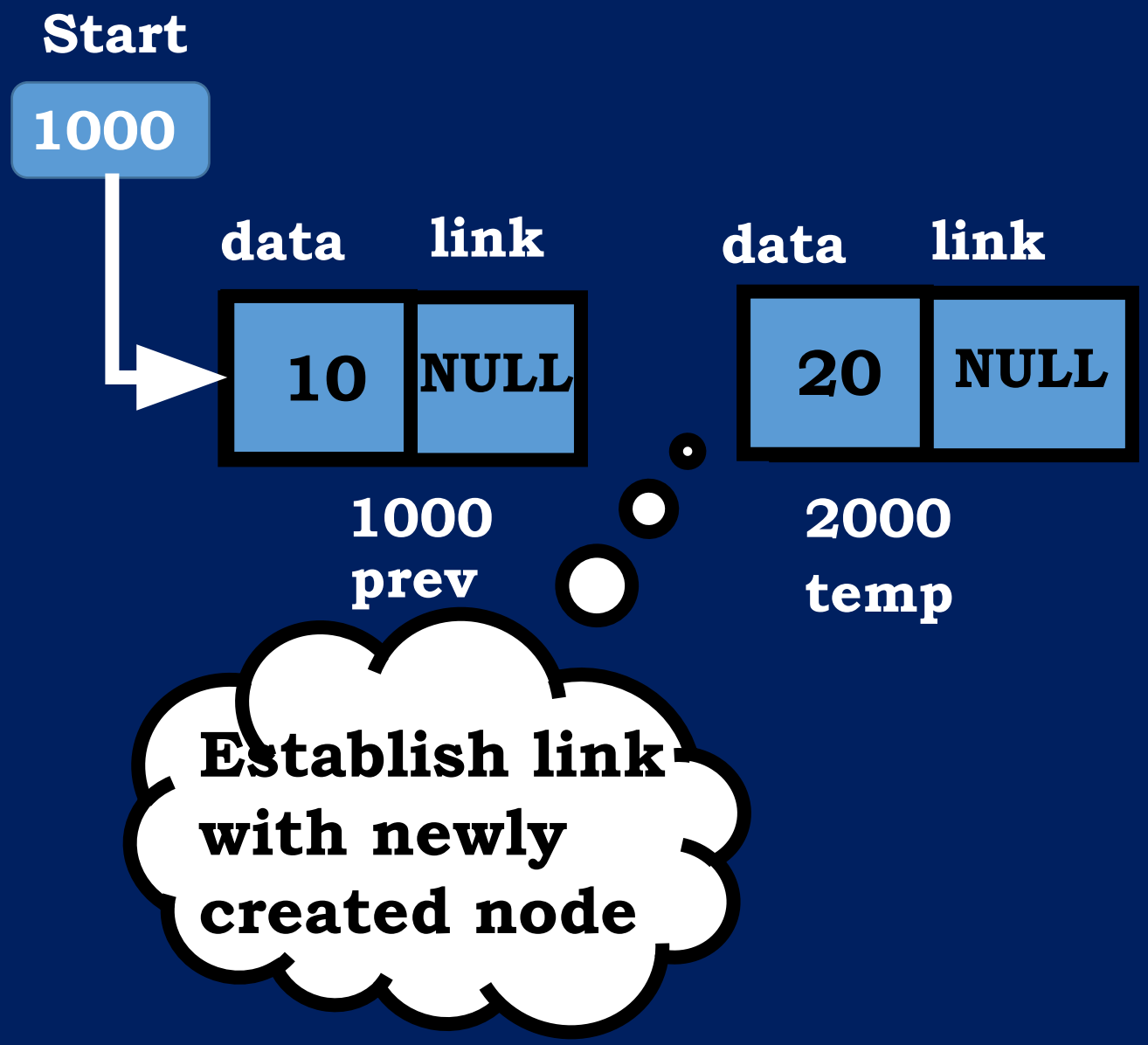
Creating a Linked List



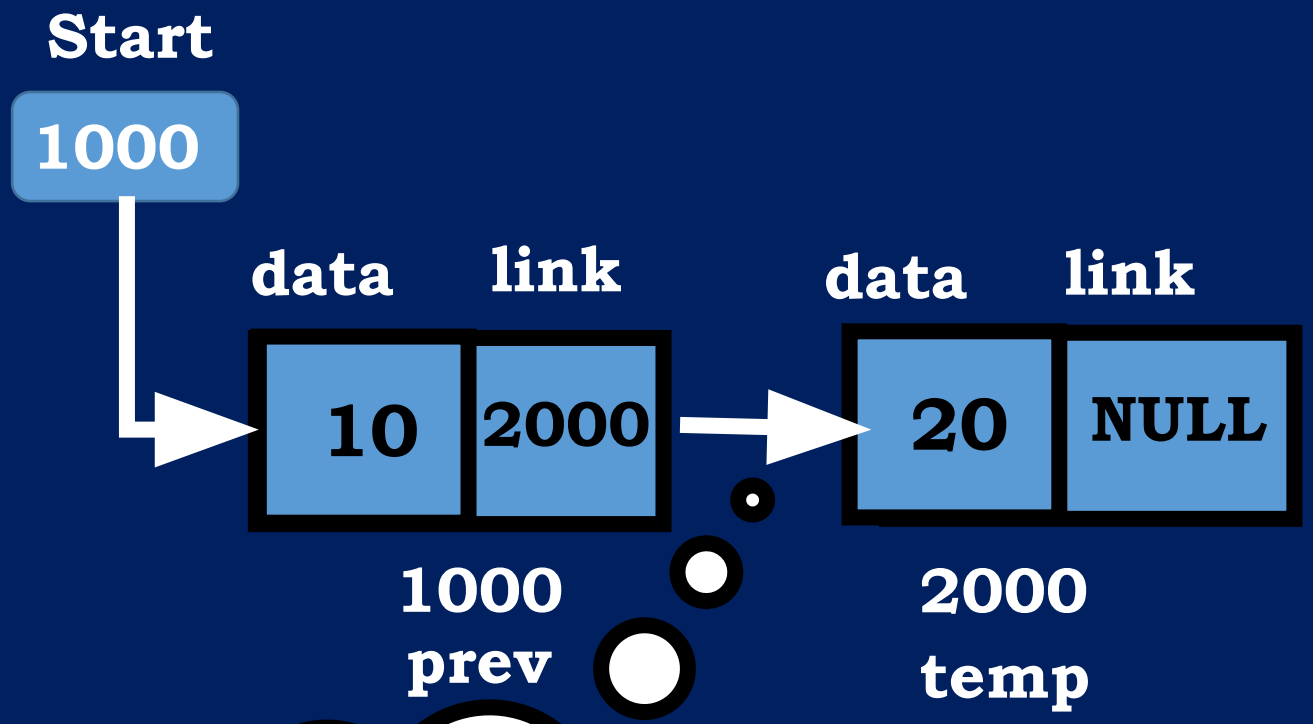
Creating a Linked List



Creating a Linked List



Creating a Linked List



CODE TO CREATE

```
void create()
{
   struct slink *t;
   char cch='y';
   while(cch=='y'||cch=='Y')
   {
      temp=(struct slink *)malloc(sizeof(struct slink));
      printf("\n Enter New Node Value (integer) : ");
      scanf("%d",&temp->data);
      temp->link = NULL;
      if(start == NULL)
    start = newNode;
```

CODE TO CREATE

```
    else
    {
        prev = start;
        while (prev->link != NULL)
            prev = prev->link;
        prev->link = temp
    }
    printf("\n Do u want to create another node
    (y-yes/n-no) :");
    fflush(stdin);
    scanf("%c",&cch);
  }
}
```

VIGNAN Institute of Technology and Science

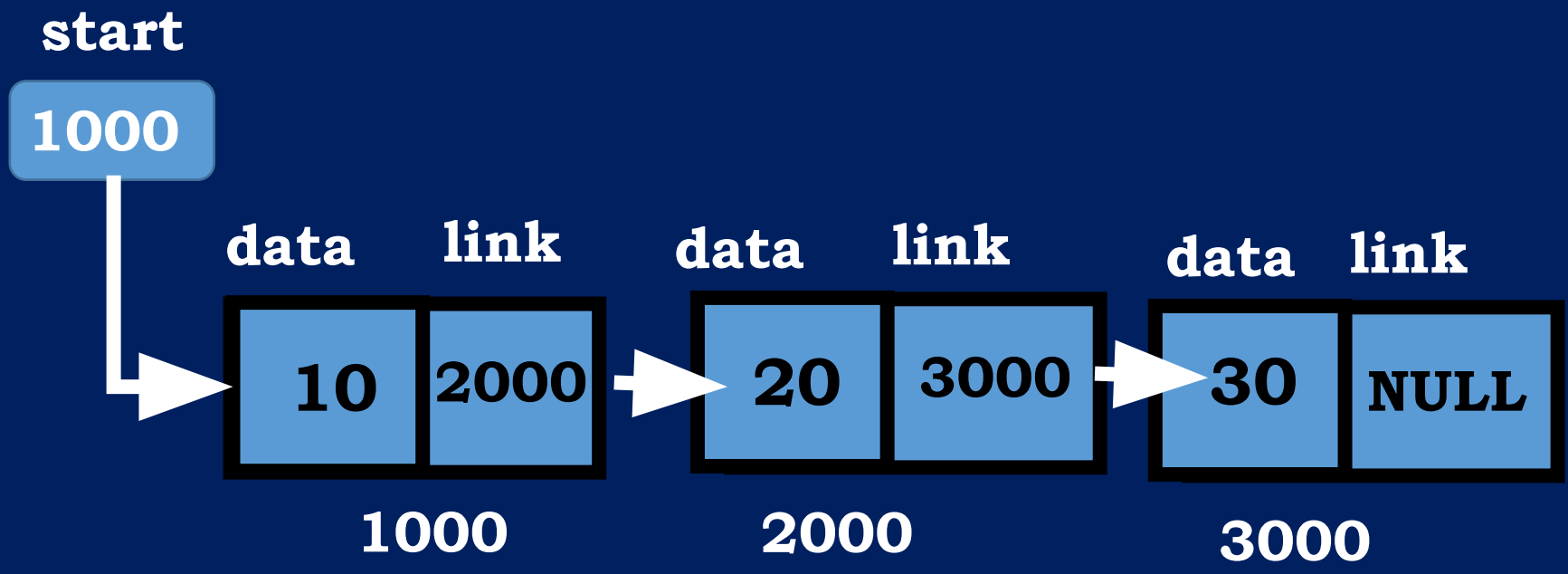
Department of Computer Science and Engineering

Traversing a Singly Linked List

Data Structures
B.Tech II year I Sem

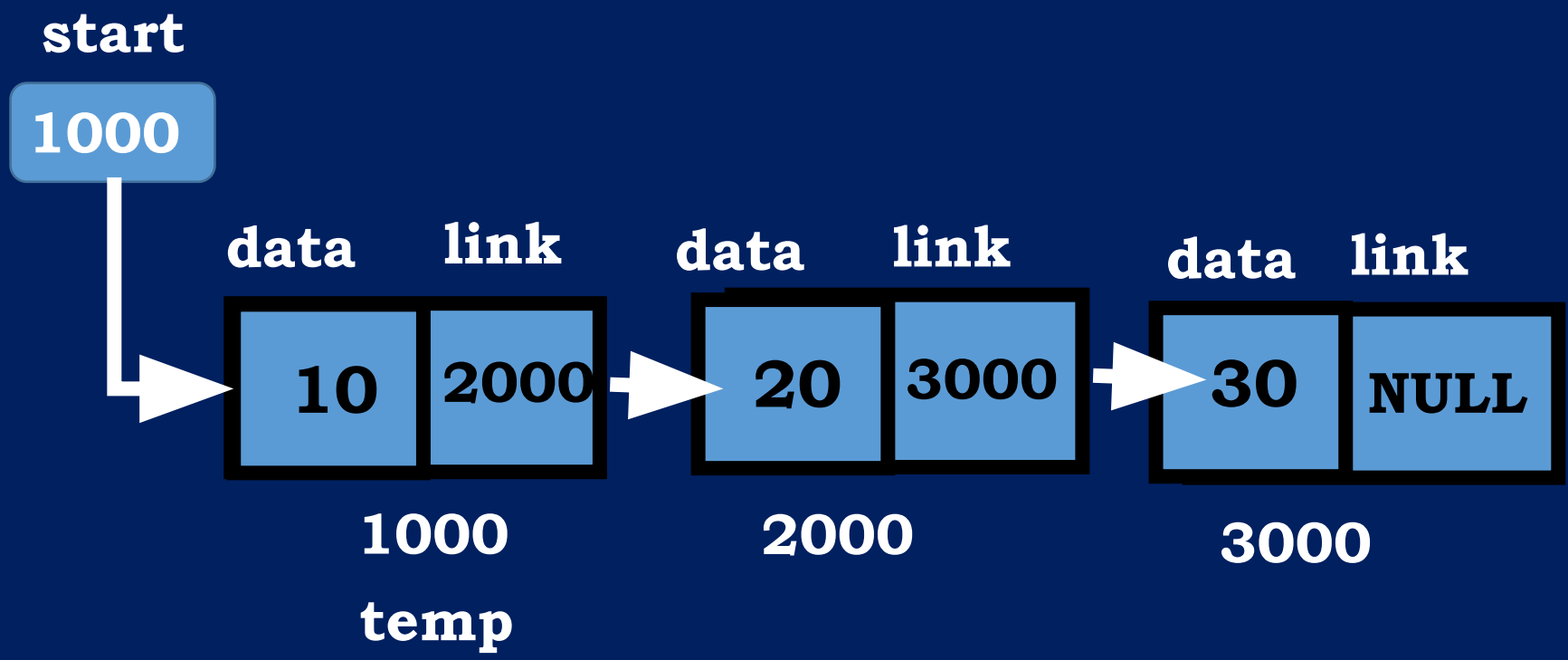
by
S Akshara Reddy
Assistant Professor
Email id: akshara.revelly@gmail.com
Blog: http://aksharatechnicalstuff.blogspot.com/

Traversing a Linked List



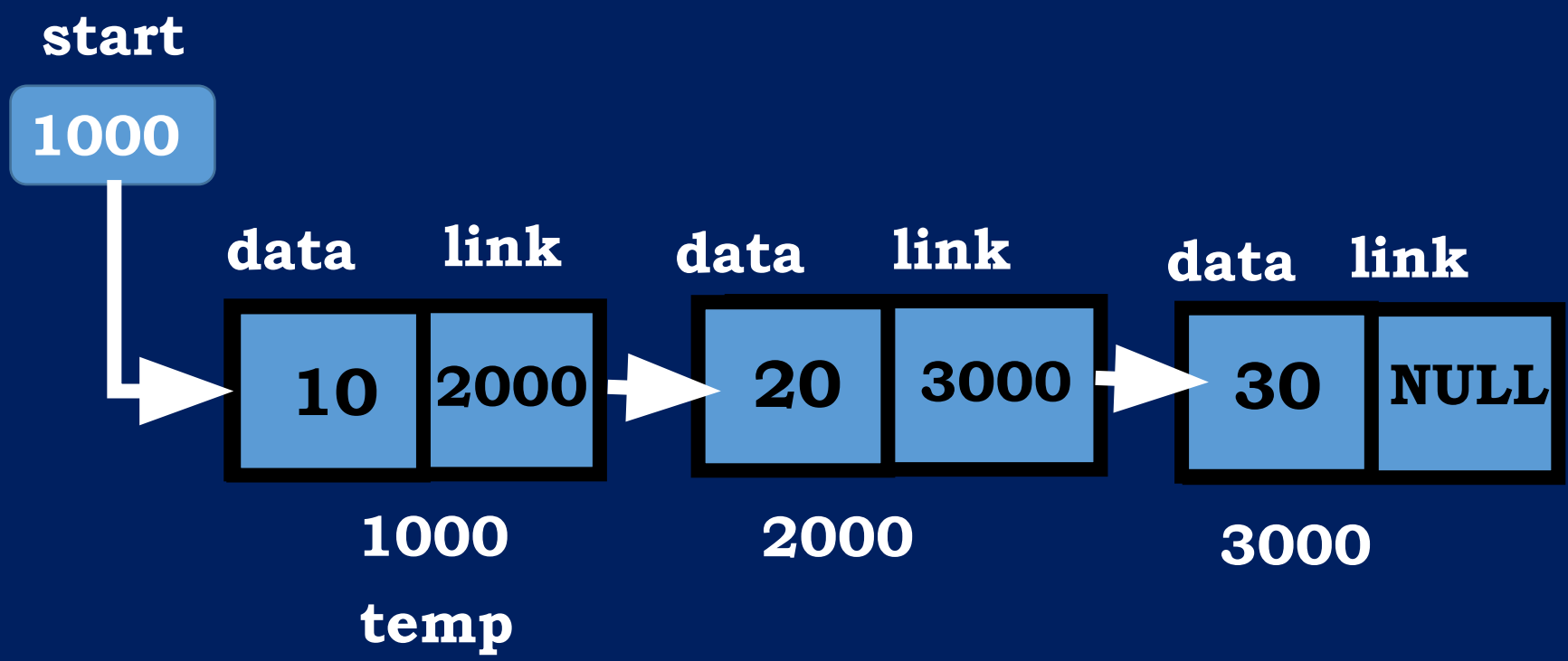
Traversing a Linked List

temp = start

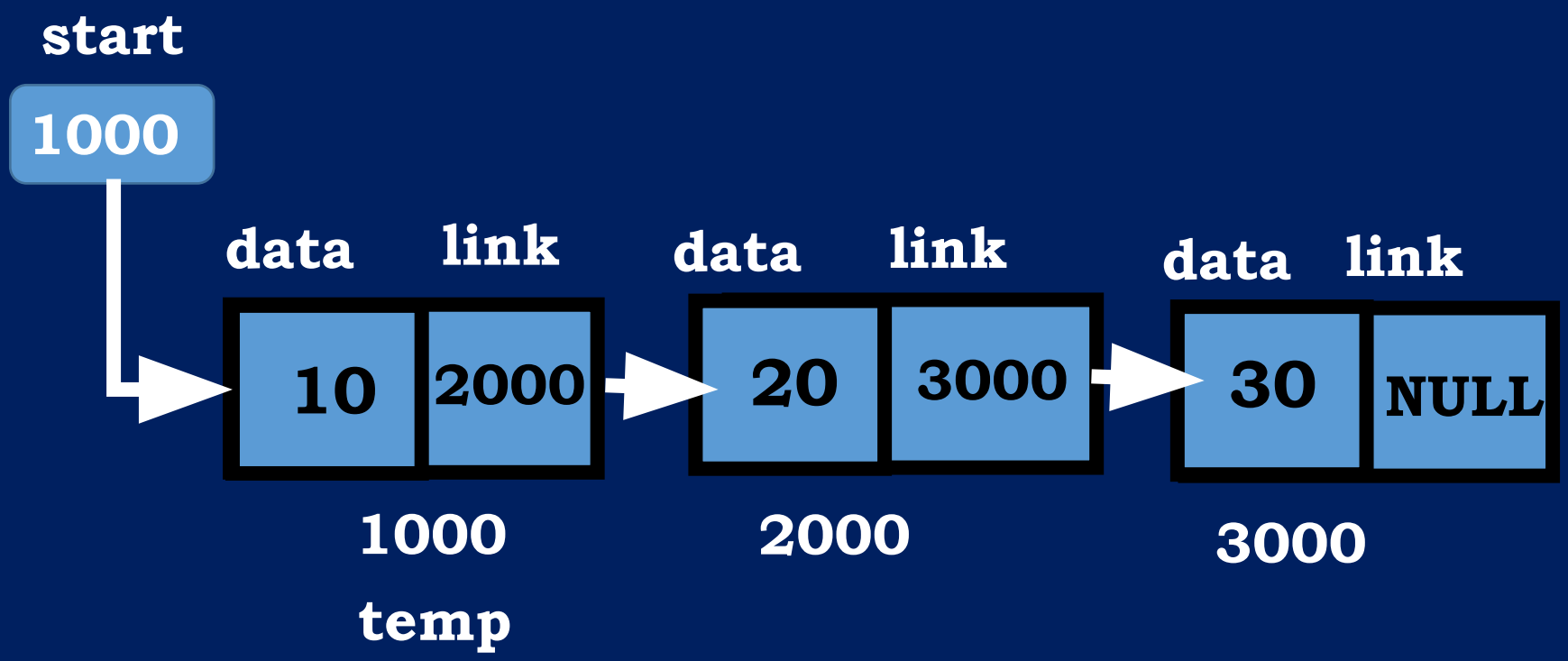


Traversing a Linked List

```
if(temp != NULL)
    printf("%d", temp->data);
```



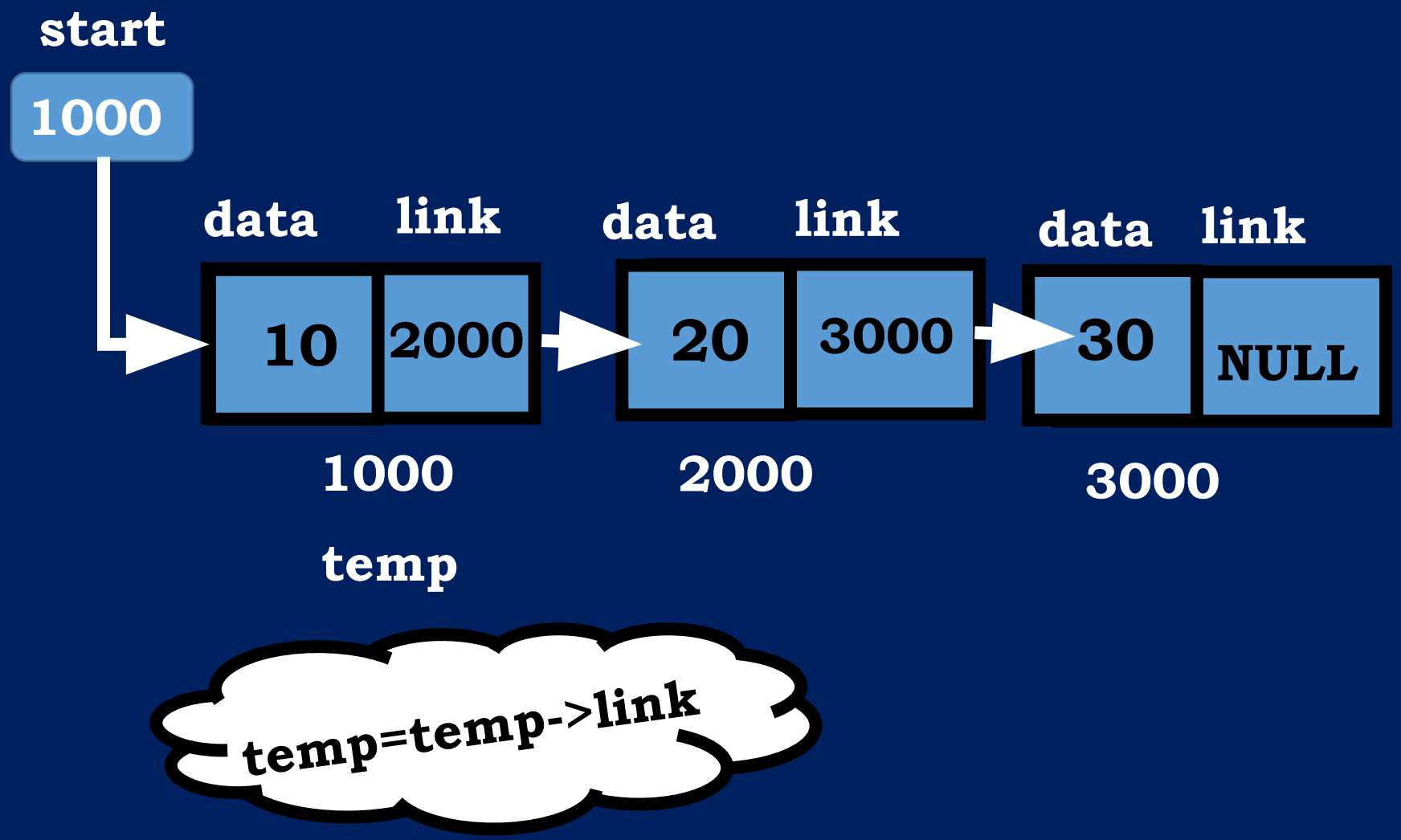
Traversing a Linked List



1000!=NULL □ 1 (TRUE)

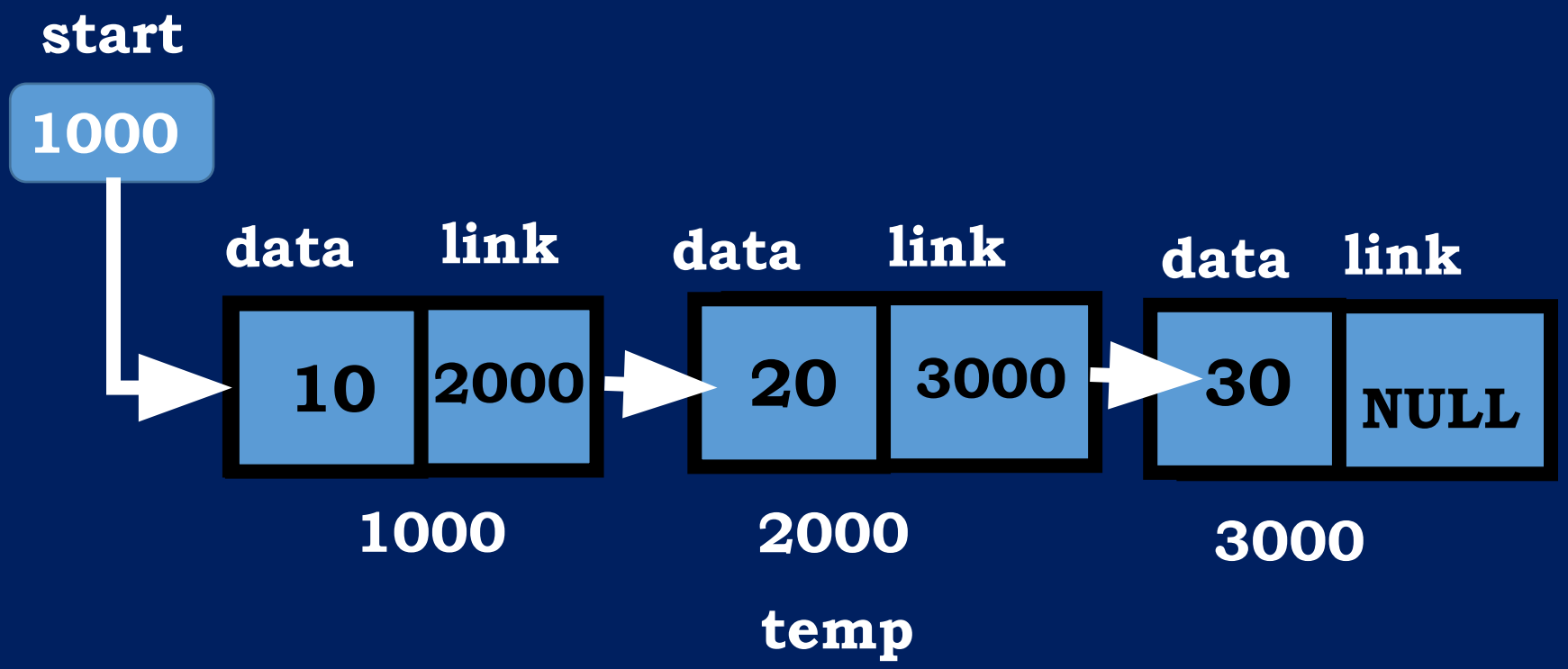
prints 10

Traversing a Linked List

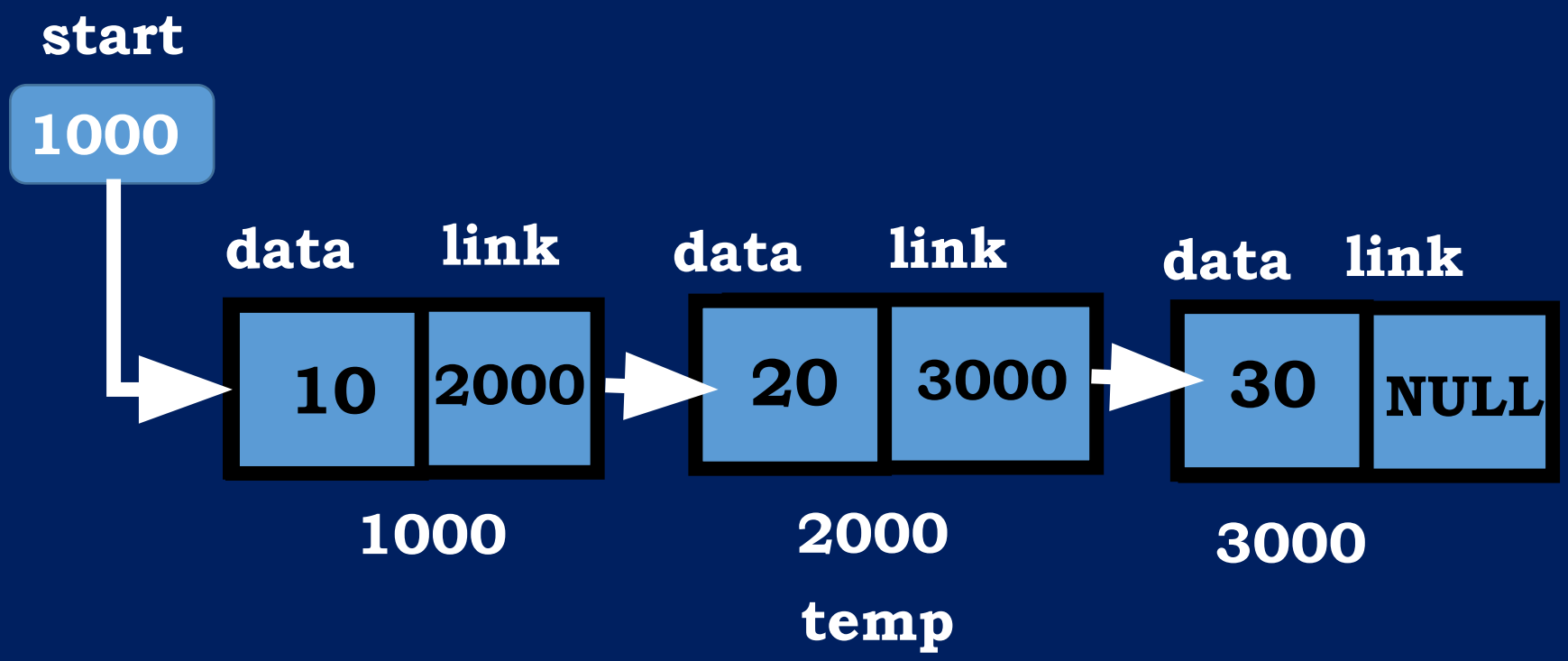


Traversing a Linked List

```
if(temp != NULL)
    printf("%d", temp->data);
```



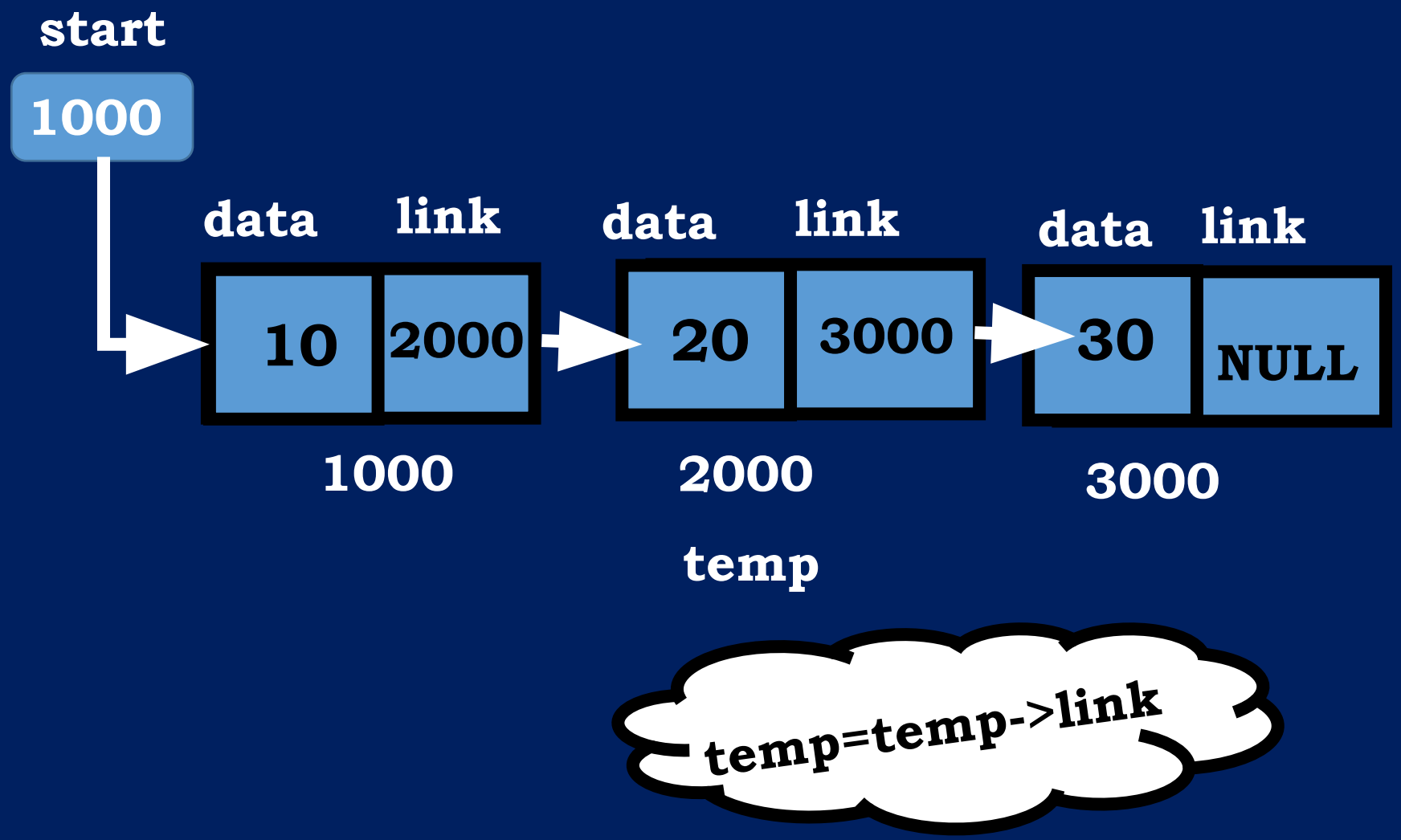
Traversing a Linked List



2000!=NULL □ 1 (TRUE)

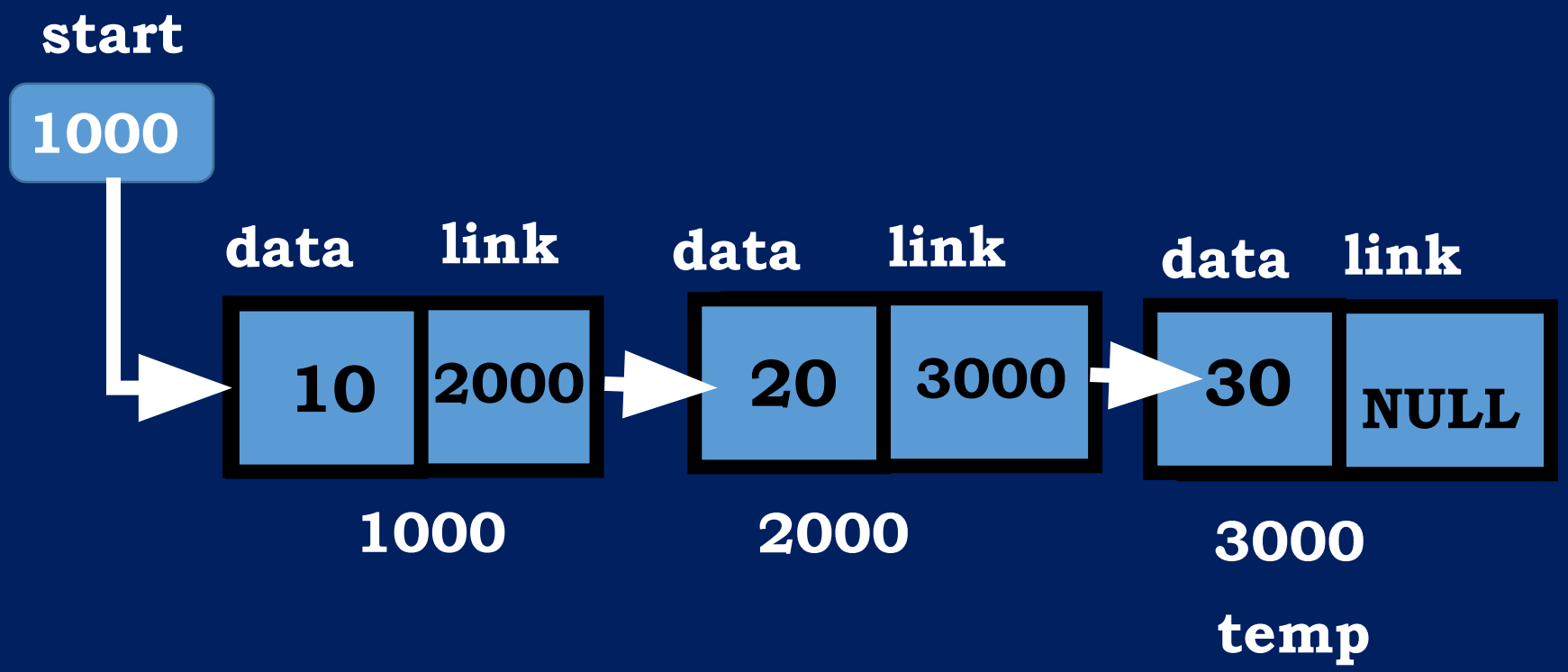
prints 20

Traversing a Linked List

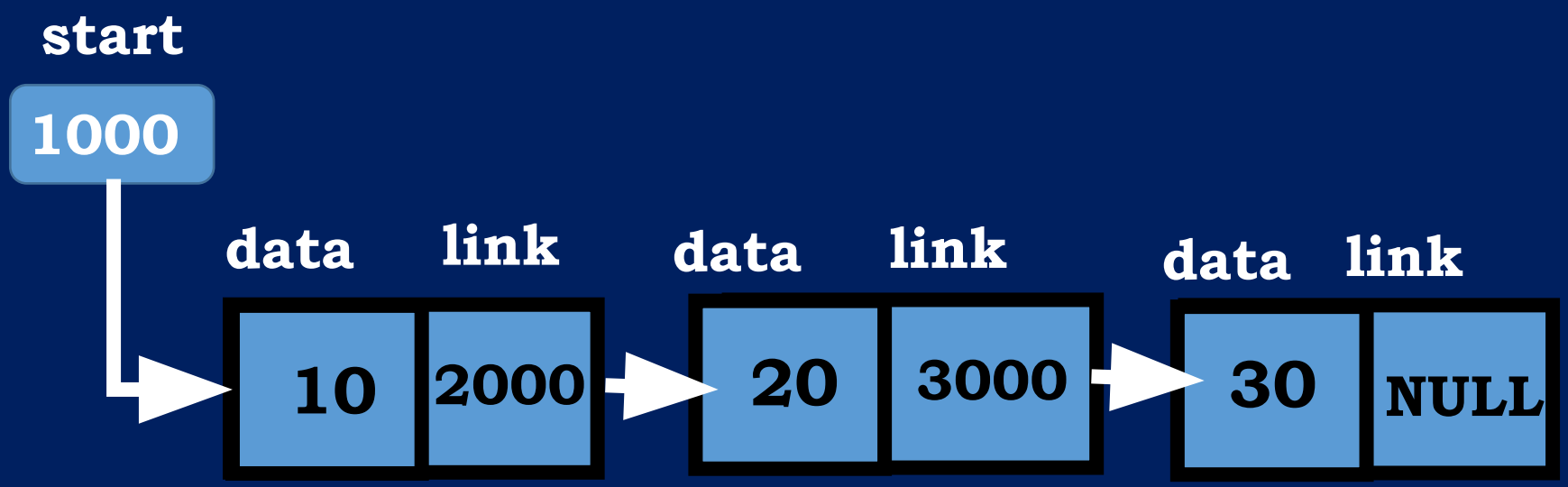


Traversing a Linked List

```
if(temp != NULL)
    printf("%d", temp->data);
```



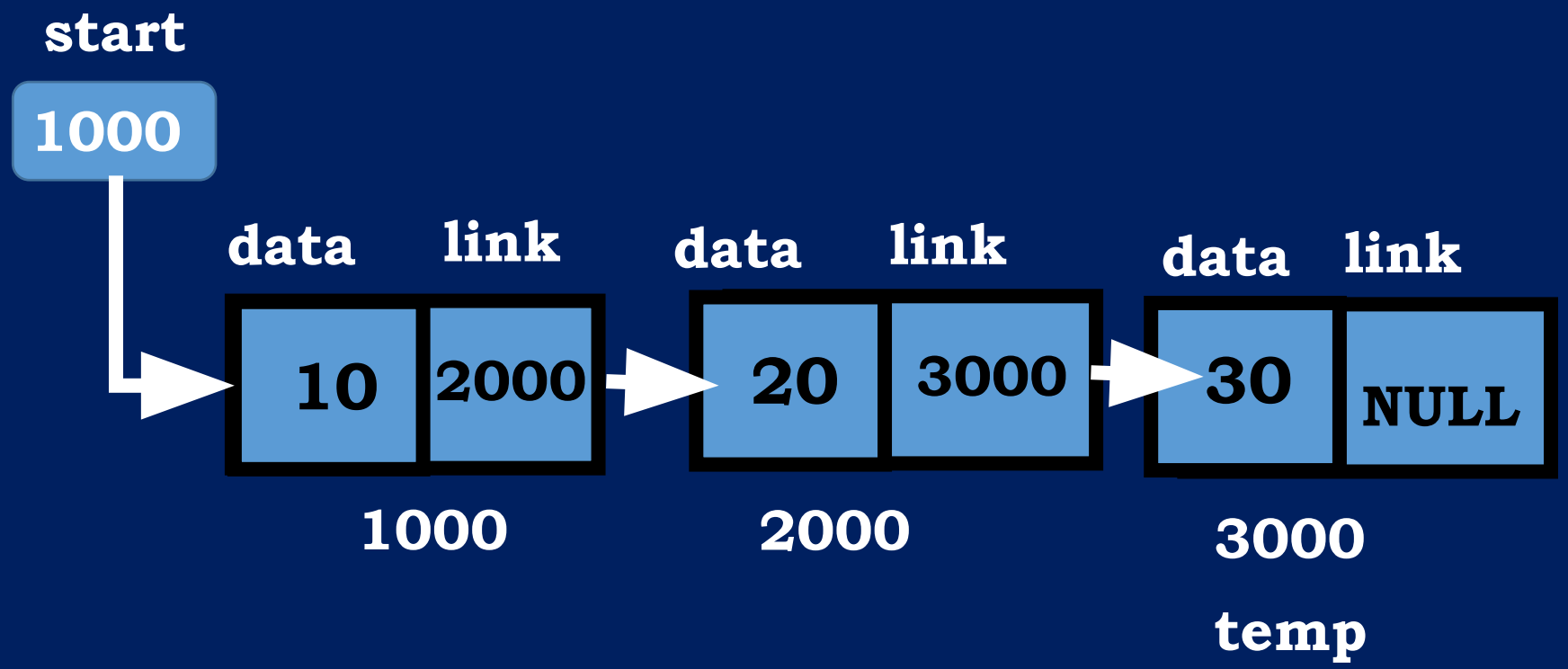
Traversing a Linked List



3000!=NULL □ 1 (TRUE)

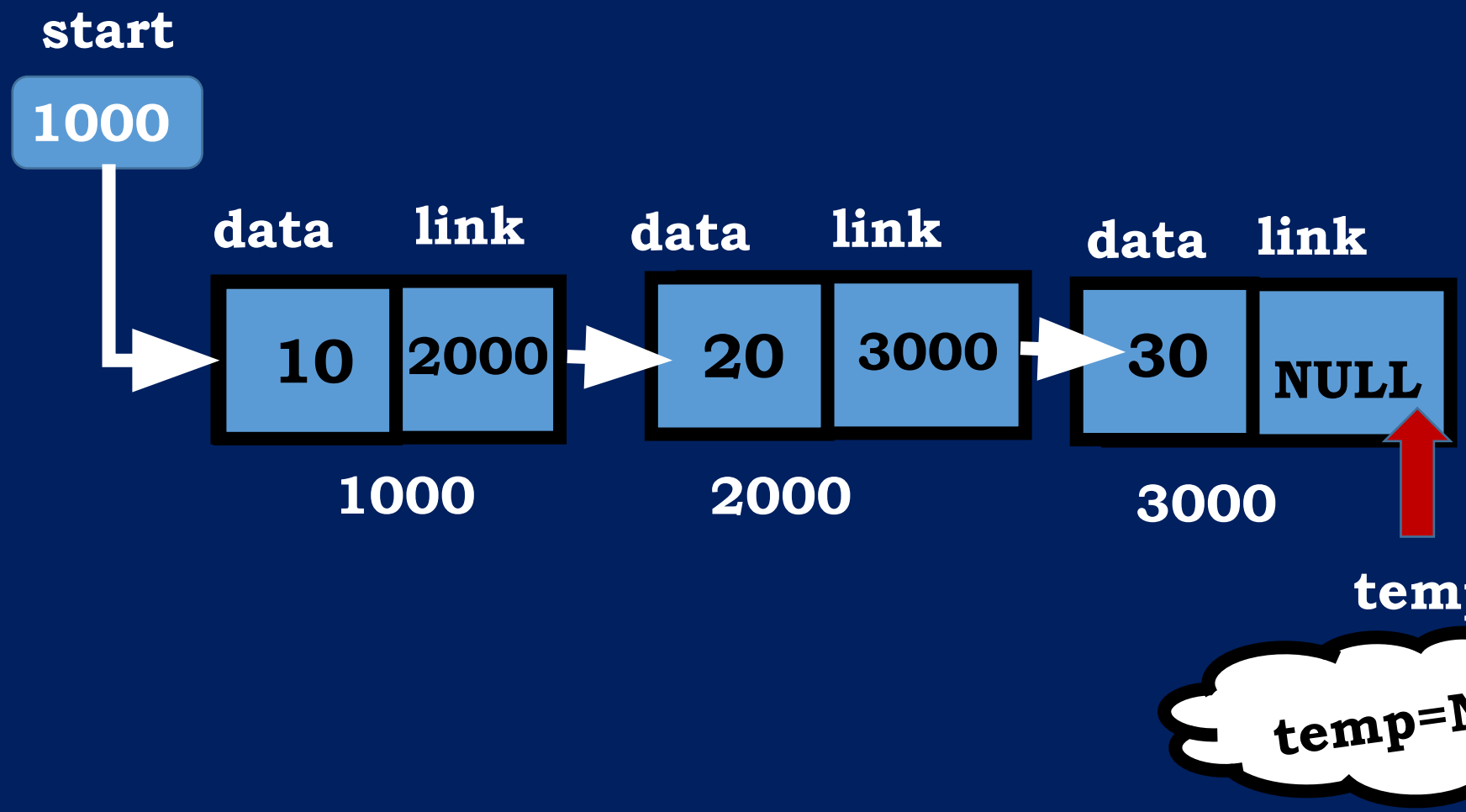
prints 30

Traversing a Linked List



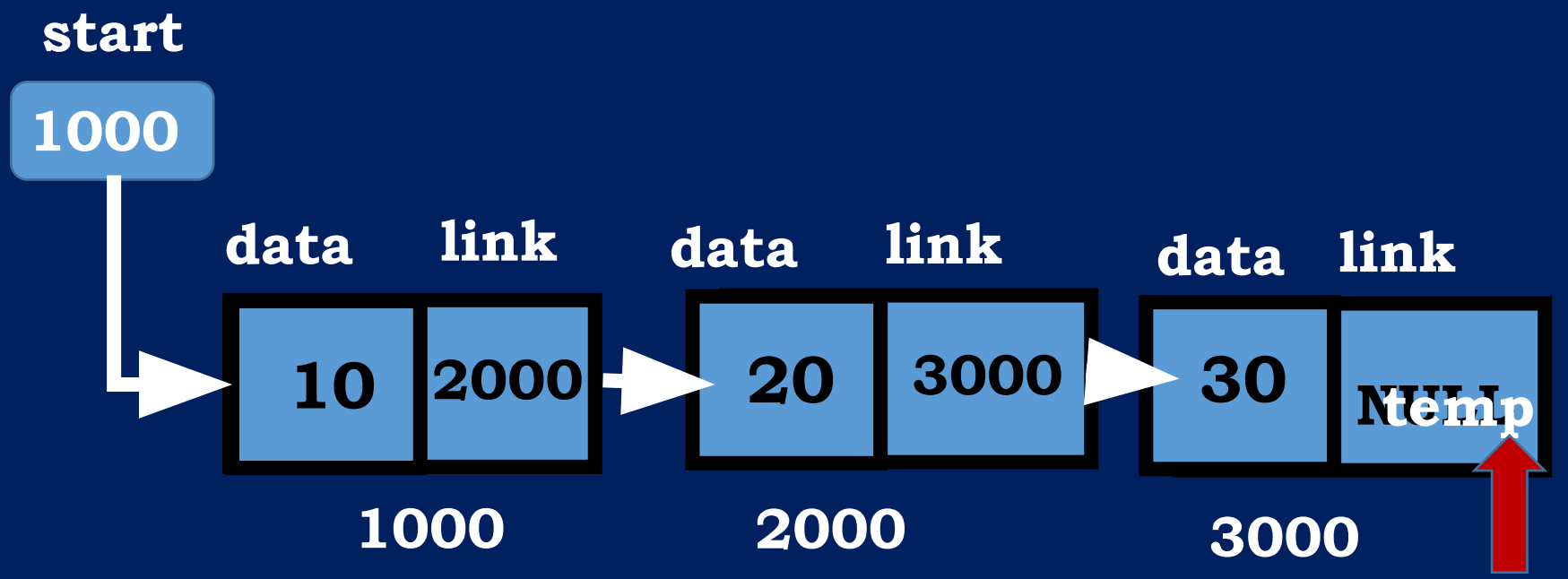
temp=temp->link

Traversing a Linked List



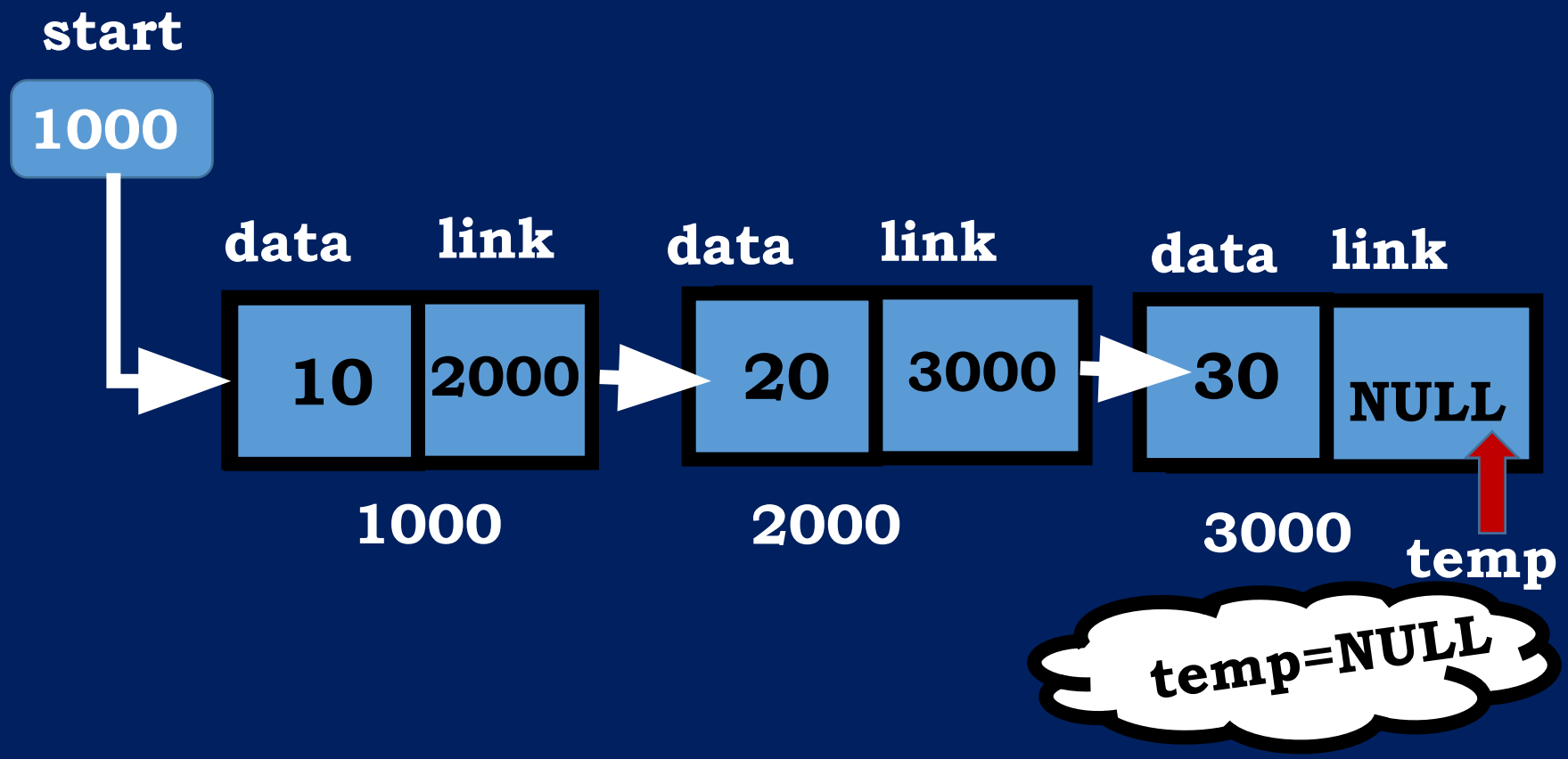
Traversing a Linked List

```
if(temp != NULL)
    printf("%d", temp->data);
```



temp=NULL

Traversing a Linked List



NULL!=NULL □ 0 (FALSE)

STOP

CODE TO TRAVERSE

```
void display()
{
    if(start==NULL)
     printf("\n Sorry No Node to Display ");
    else
    {
     printf("\n Node values are :\n ");
     temp=start;
     while(temp!=NULL)
        {
            printf("\t%d",temp->data);
            temp=temp->link;
        }
    }}
```