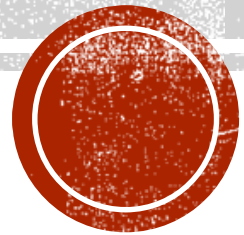
DYNAMIC STACKS



18.2 DYNAMIC STACKS

- A dynamic stack is built on a linked list instead of an array.
- A linked list-based stack offers two advantages over an array-based stack.
  - No need to specify the starting size of the stack. A dynamic stack simply starts as an empty linked list, and then expands by one node each time a value is pushed.
  - A dynamic stack will never be full, as long as the system has enough free memory.

CONTENTS OF DYNINTSTACK.H

```
class DynIntStack
{
private:
	struct StackNode
	{
		int value;
		StackNode *next;
	};

	StackNode *top;

public:
	DynIntStack(void)
		{	top = NULL; }
	void push(int);
	void pop(int &);
	bool isEmpty(void);
};
```

CONTENTS OF DYNINTSTACK.CPP

```
#include <iostream.h>
#include "dynintstack.h"

//*********************************************
// Member function push pushes the argument onto *
// the stack.                                     *
//*********************************************

void DynIntStack::push(int num)
{
        stackNode *newNode;

        // Allocate a new node & store Num
        newNode = new stackNode;
        newNode->value = num;
```

CONTENTS OF DYNINTSTACK.CPP

```
        // If there are no nodes in the list
        // make newNode the first node
        if (isEmpty())
        {
                top = newNode;
                newNode->next = NULL;
        }
        else    // Otherwise, insert NewNode before top
        {
                newNode->next = top;
                top = newNode;
        }
}


//**************************************************
// Member function pop pops the value at the top     *
// of the stack off, and copies it into the variable *
// passed as an argument.                            *
//**************************************************
```

CONTENTS OF DYNINTSTACK.CPP

```
void DynIntStack::pop(int &num)
{
        stackNode *temp;

        if (isEmpty())
        {
                cout << "The stack is empty.\n";
        }
        else    // pop value off top of stack
        {
                num = top->value;
                temp = top->next;
                delete top;
                top = temp;
        }
}
```

CONTENTS OF DYNINTSTACK.CPP

```
//*************************************************
// Member funciton isEmpty returns true if the stack *
// is empty, or false otherwise.                     *
//*************************************************

bool DynIntStack::isEmpty(void)
{
        bool status;

        if (!top)
                status = true;
        else
                status = false;

        return status;
}
```

PROGRAM 18-3

```
// This program demonstrates the dynamic stack
// class DynIntClass.

#include <iostream.h>
#include "dynintstack.h“

void main(void)
{
        DynIntStack stack;
        int catchVar;


        cout << "Pushing 5\n";
        stack.push(5);
        cout << "Pushing 10\n";
        stack.push(10);
        cout << "Pushing 15\n";
        stack.push(15);
```

PROGRAM 18-3 (CONTINUED)

```
        cout << "Popping...\n";
        stack.pop(catchVar);
        cout << catchVar << endl;
        stack.pop(catchVar);
        cout << catchVar << endl;
        stack.pop(catchVar);
        cout << catchVar << endl;

        cout << "\nAttempting to pop again... ";
        stack.pop(catchVar);
}
```

Program Output

```
Pushing 5
Pushing 10
Pushing 15
Popping...
15
10
5

Attempting to pop again... The stack is empty.
```