

Arrays of Pointers

The following code declares an array of pointers to char:

```
char *arr[7];
```



Each element can contain
a pointer to char

Often used for strings

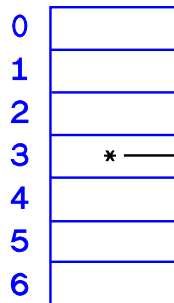
STRING010

University of Idaho

Arrays of Pointers

The following would define element 3 to point to a string:

```
arr[3] = "Hello world";
```



H e l l o w o r l d / 0

The letter 'r' in the string could be
addressed with:

```
*(arr[3] + 8) or arr[3][8]
```

STRING020

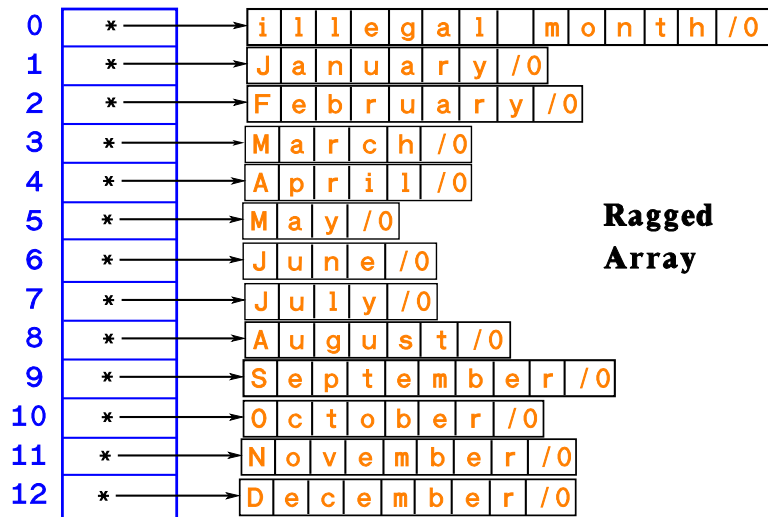
University of Idaho

```

char *month_name(int n) // return name of nth month
{
    static char *name[] =
    {
        "illegal month",
        "January",
        "February",
        "March",
        "April",
        "May",
        "June",
        "July",
        "August",
        "September",
        "October",
        "November",
        "December"
    };
    return((n < 1 || n > 12) ? name[0] : name[n]);
} // END month_name

```

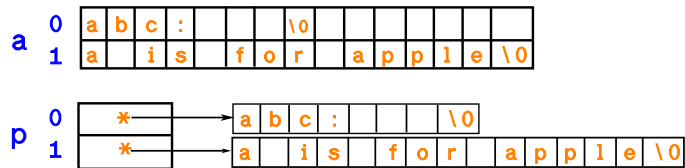
STRING030



STRING040

Are These Two Declarations Different?

```
char a[2][15] =  
    {"abc:  ", "a is for apple"};  
  
char *p[2] =  
    {"abc:  ", "a is for apple"};
```



STRING060

Command Line Arguments

```
int main(int argc, char *argv[])
```

argc – the number of arguments on the command line (incl the command name itself)

argv – pointer to array of pointers to char – an "array of strings" containing the command line arguments

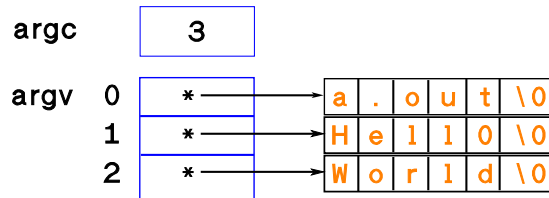
argv[0] – the command name itself

STRING060

Command Line Interface

Typing the following command will produce the values shown:

```
a.out Hello world
```



STRING070

```
int main(int argc, char *argv[])
{
    int i;

    cout << "This program was called with "
         << argc << " arguments\n";

    for(i = 0; i < argc; i++)
        cout << argv[i] << "\n";

} // END main
```

STRING080