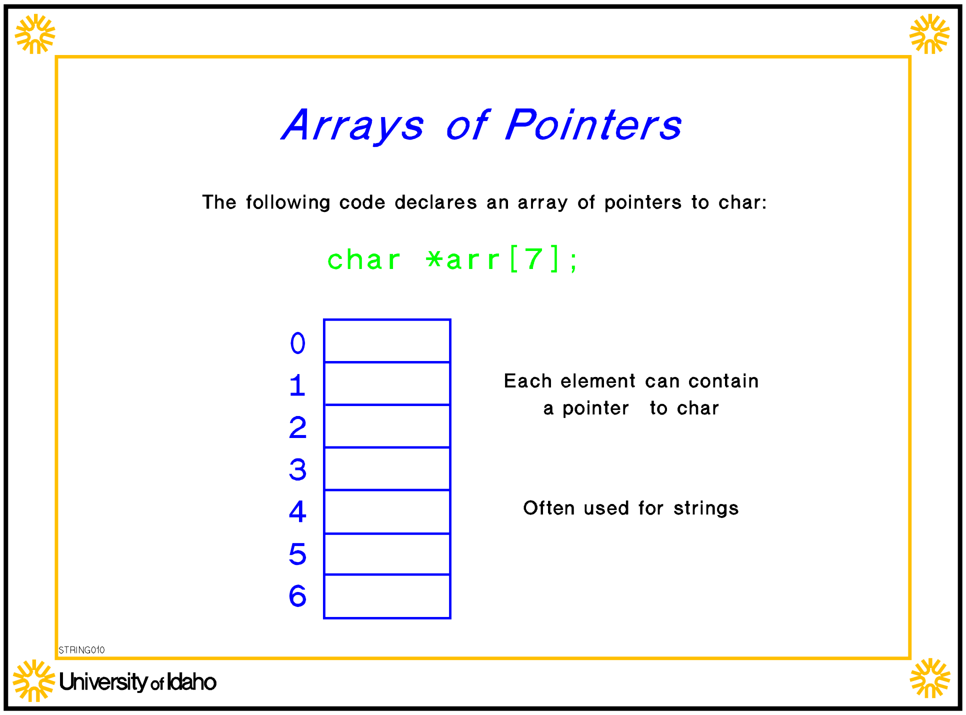


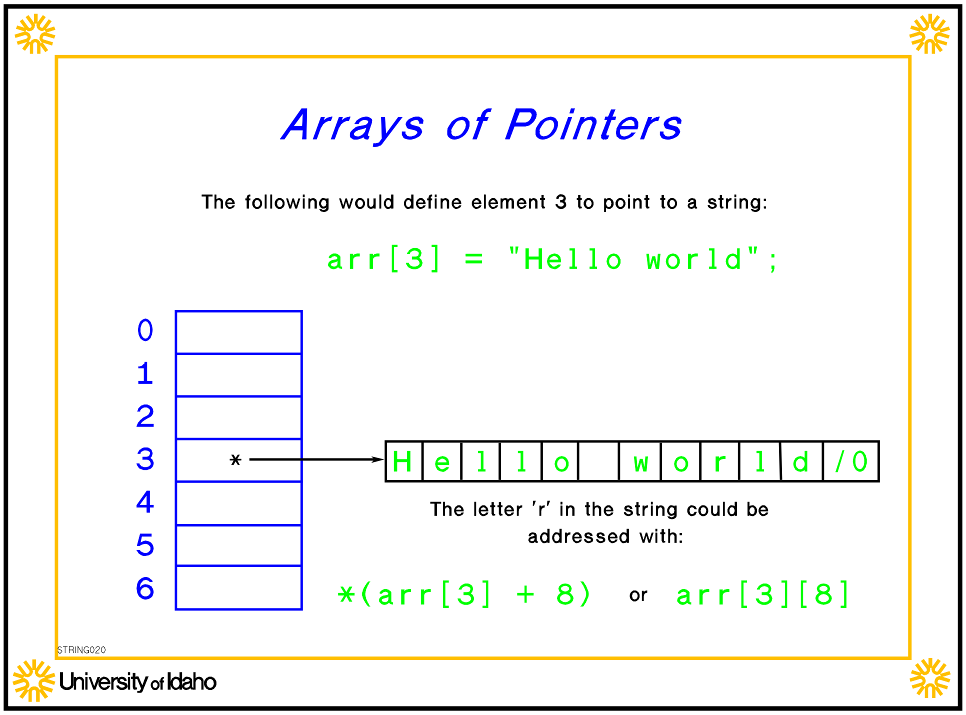
Arrays of Pointers

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



Arrays of Pointers

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

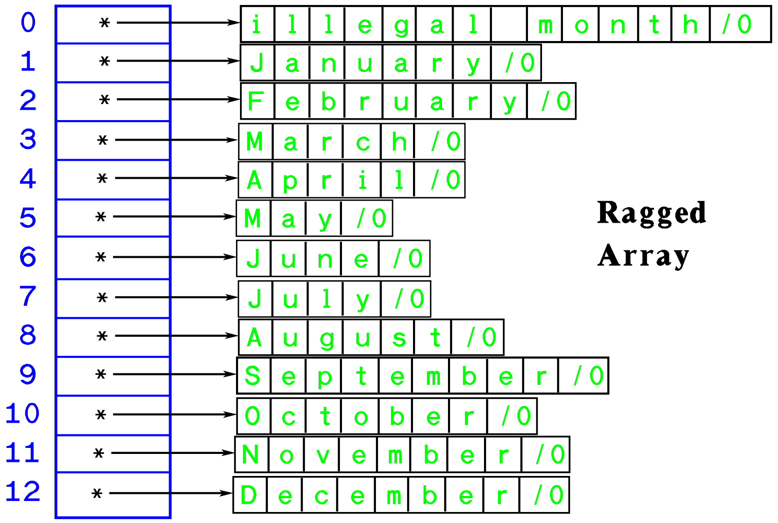
```
arr[3] = "Hello world";
```

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

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

```
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



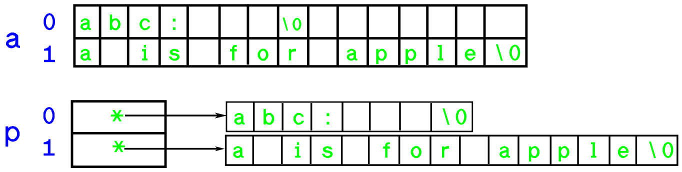
Ragged Array

STRING040

Are These Two Declarations Different?

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

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



Command Line Arguments

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

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

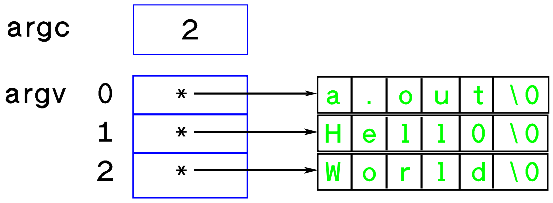
argv – pointer to array of pointers to char – an "array of strings" containing the command line arguments

argv[0] – the command name itself

Command Line Interface

Typing the following command will produce
the values shown:

```
a.out Hello world
```



a	.	o	u	t	\0
H	e	l	l	0	\0
W	o	r	l	d	\0

```
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
```

```
#include <stdio.h>

 void printn(int val, int base)
 {
    int remainder, leftover;
    char digit;

    if(val < 0)
      {
       val = -val;
       putchar(’-’);
      }

    leftover = val / base;
    remainder = val % base;

    if(leftover > 0)
       printn(leftover, base);

    digit = "0123456789ABCDEF"[remainder];

  putchar(digit);

  } /* END printn */
```