Arrays & Strings
Review C Programming Language

Data Abstractions
CSCI-2320

Dr. Tom Hicks
Computer Science Department
Integer Array
Create a static array, called Nos1, which has the capacity to store 5 integers.

```cpp
int Nos1[5];
```

Display the contents Nos1

```cpp
for (int pos = 0; pos <= 4; pos++)
    cout << setw(15) << Nos1[pos];
```

-------------------------------- Start Of Main --------------------------------

-858993460  -858993460  -858993460  -858993460  -858993460

Nos1 is a static array!

The initial contents are garbage - your answers may be different than mine.

Arrays will always be stored in one contiguous block of memory.
Create An Array, Called Nos1, Which Has The Capacity To Store 5 Integers

```
int Nos[5];
```

What Happens If We Try To Access Elements Outside The Acceptable Array Boundary?

```
printf("Nos[5] = %ld\n", Nos[5]);
printf("Nos[-1] = %ld\n", Nos[-1]);
```

Most C/C+ Compilers Will Not Flag The Boundary Errors You Must Be Careful!
Fill Nos1 With Values 1,2,3,4,5

```c
int
    Nos1[5];

for (int pos = 0; pos <= 4; pos++)
    Nos1[pos] = pos + 1;

Display The Contents Nos1

for (int pos = 0; pos <= 4; pos++)
    printf("%15ld", Nos1[pos]);
```
What Happens If We Try To Change Elements Outside The Acceptable Array Boundary?

```c
int
    Nos1[5];
Nos1[5] = 11;
Nos1[-1] = -11;
printf("Nos[5] = %ld\n", Nos[5]);
printf("Nos[-1] = %ld\n", Nos[-1]);
```

Most C/C+ Compilers Will Not Flag The Boundary Errors
You Must Be Careful!

Display The Size Of Nos1

```c
sizeof Does Not Work For Dynamic Arrays
```

```c
printf("sizeof(Nos1) = %ld\n\n", sizeof(Nos1));
```
int Nos1[5];

for (int pos = 0; pos <= 4; pos++)
    Nos1[pos] = pos + 1;

Display The Contents Nos1

for (int pos = 0; pos <= 4; pos++)
    printf("%15ld", Nos1[pos]);
int Nos2[] = { 1, 2, 3, 4 };

Display The Contents Nos1

for (int pos = 0; pos <= 3; pos++)
    printf("%8ld", Nos2[pos]);

Display The Number Of Bytes In Nos2

printf("sizeof(Nos2) = %ld\n\n\n", sizeof(Nos2));
Display The Number Of Elements In Array Nos2

```c
int Nos2[] = { 1, 2, 3, 4 };

printf("No Elements In Nos2 = %ld\n\n", sizeof(Nos2)/sizeof(int));
```

Create A Display That Will Work With Any Number Of Integers In The Static Array Nos2

```c
for (int pos = 0; pos < sizeof(Nos2)/sizeof(int); pos++)
    printf("%8ld", Nos2[pos]);
```

Start Of Main

1 2 3 4
int Nos3[] = { 0, 1, 2, 3, 4 };
printf("&Nos3 = %ld = %X\n\n", &Nos3, &Nos3);

Display The Address & Contents Of Nos3[0]

printf("Nos3[0] = %ld\n\n", Nos3[0]);
printf("&Nos3[0] = %ld = %X\n\n", &Nos3[0], &Nos3[0]);

The Locations Where The Compiler Chooses To Store Variables Can & Will Change

printf("&Nos3 = %ld = %X\n\n", &Nos3, &Nos3);
printf("Nos3[0] = %ld\n\n", Nos3[0]);
printf("&Nos3[0] = %ld = %X\n\n", &Nos3[0], &Nos3[0]);

The Base Address Of Nos3 (currently 9697803) Will Always Be The Same As The Address Of Nos3[0] ⇒ They Are The Same!
Display The Contents & Address Of Nos4

```c
int
Nos4[] = { 0,1,2,3,4,5 };
```

<table>
<thead>
<tr>
<th>Pos</th>
<th>Nos4[Pos]</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0x00000000</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>0x00000008</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>0x0000000C</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>0x00000010</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>0x00000014</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>0x00000018</td>
</tr>
</tbody>
</table>

Arrays Will Always Be Stored In One Contiguous Block Of Memory

Note That Each Of The Elements Are 4 Bytes Apart; The Size Of An int Is 4 Bytes On Our 64 bit compiler!
Character Array
String
Create A Static Array, Called First, Which Has The Capacity To Store A 5 Character First Name

```c
char First[6];
```

Don't Forget We Need One Character For The End Of String Marker

Place "Sarah" Into First

```c
First[0] = 'S';
First[1] = 'a';
First[2] = 'r';
First[3] = 'a';
First[4] = 'h';
First[5] = '\0';
```

We Could Fill It One Character At A Time
Create a static array, called `First`, which has the capacity to store a 5 character first name.

```c
char First[6];
```

Don't forget we need one character for the end of string marker.

Place "Sarah" into `First`

```c
First[0] = 'S';
First[1] = 'a';
First[2] = 'r';
First[3] = 'a';
First[4] = 'h';
First[5] = '\0';
```

Here are 2 different ways we can insert the end-of-string marker in `First[5]`.

We must do so!
Create A Static Array, Called First1 and Initialize It With "Sarah"

```c
char
First1[] = {'S', 'a', 'r', 'a', 'h', 0};
```

Display The String In First - One Character At A Time

```c
for (int pos = 0; pos < 5; pos++)
    printf("%c", First1[pos]);
```

Display The String Using cout

```c
cout << First1 << endl;
```

Display The String Using printf

```c
printf("%s\n", First1);
```
Create A Static Array, Called First2 and Initialize It With "Sarah"

```c
char First2[] = {'S', 'a', 'r', 'a', 'h', '\0'};
```

Display The String Using puts

```c
puts(First2);
```

Display The Number Of Bytes In First2

```c
printf("sizeof(First2) = %ld\n", sizeof(First2));
```
Display The Address Of First3

```c
char First3[] = {'A','B','C','D','E', '\0'};
```

```c
printf("&First3 = %ld = %X\n\n", &First3, & First3);
```

Display The Address & Contents Of First3[0]

```c
The Locations Where The Compiler Chooses To Store Variables Can & Will Change
```

```c
&First3 = 19921868 = 12FFBCC
&First3 = 5373528 = 51FE58
First3 [0] = A
&First3 ] = 5373528 = 51FE58
```

```c
printf("&First3 = %ld = %X\n\n", &First3, & First3);
printf("First3[0] = %c\n\n", First3[0]);
printf("&First3 ] = %ld = %X\n\n", & First3[0], & First3[0]);
```

The Base Address Of First3 (currently 5373528) Will Always Be The Same As The Address Of First3[0] ➜ They Are The Same!
char

First3[] = {'A','B','C','D','E', '\0'};

for (int Pos = 0; Pos <= 5; Pos ++)
    printf("First3[%d] = %c --> & = %ld\n\n", Pos, First3[Pos], &First[Pos]);

Arrays Will Always Be Stored In One Contiguous Block Of Memory

Note That Each Of The Elements Are 1 Bytes Apart; The Size Of An char Is 1 Byte On Our 64 bit compiler!
Characters Stored In ASCII → print the contents with %d

```c
char
First3[] = {'A','B','C','D','E', '\0'};

for (int Pos = 0; Pos <= 5; Pos ++)
    printf("First3[%d] = %d\n\n", Pos, First3[Pos]);
```

Yea Binary!
A = 65 → Stored As 01000001

All The Stupid Computer Knows Is Binary ➔ BUT IT IS FAST!

ASCII for A = 65, ASCII for B = 66, etc.

Note That The EOS Marker = 0!
strlen

string.h
Create A Static Array That Can Hold 7 Characters → Capacity of 8 → Called First3

```
char First3[8];
```

Display The Number Of Bytes In First3

```c
printf("sizeof(First3) = %ld\n", sizeof(First3));
```

Display The Number Of Characters In First3

```c
printf("strlen(First3) = %ld\n", strlen(First3));
```

(strlen → Since The string has not been properly filled, strlen does not work correctly.)

```c
strlen(First3) = 23
```
Create a static array that can hold 7 characters → Capacity of 8 → Called First3

```c
char First3[8];
```

---

Place Sam in First3

```c
First3[0] = 'S';
First3[1] = 'a';
First3[2] = 'm';
First3[3] = '\0';
puts(First3);
```

---

Display the number of characters in First3

```c
printf("strlen(First3) = %ld\n", strlen(First3));
```

**strlen** → Works correctly when the character array is filled properly with the EOF

```
strlen(First3) = 3
```
strcpy_s

strcpy_s inserts an EOS marker

string.h
Create A Static Array, Called First4, With A Capacity Of 7

```c
char First4[7];
```

Place The String "Alex" In First4

```c
First4 = "Alex";
```

We Have Shown That You Can Fill The String (Character Array → 1 Character At A Time

But You Can't Fill It With The = Assignment

You Can Fill The String With The strcpy_s Function

```c
strcpy_s(First4, "Alex");
```

```c
printf("First4 = %s\n", First4);
```

Display The Number Of Characters In First4

```c
printf("strlen(First4) = %ld\n", strlen(First4));
```

```c
First4 = Alex
```

```c
strlen(First4) = 4
```
Overflowing Arrays

It is the programmer's responsibility to make sure that their arrays are of the proper size.

Bad things happen when you start changing elements outside the boundaries of your array!
Load 3 Strings With "Tom", "Dick" & "Harry"

```c
char
Str11[] = { 'T', 'o', 'm', 0 },
Str12[] = { 'D', 'i', 'c', 'k', '\0' },
Str13[] = { 'T', 'O', 'm', 0 },
Str14[] = { 'T', 'o', 'm', 'm', 'y', 0 },
Str15[] = { ' ', 'T', 'o', 'm', 0 };

puts(Str11);
puts(Str12);
puts(Str13);
puts(Str14);
puts(Str15);
```

strlen Overflowed The capacity of Str2 → When You Overflow The Capacity Of An Array, It Always Destroys Something Else (this time Str1).
• It Is The Programmer's Responsibility To Make Sure That Their Arrays Are Of The Proper Size

• Bad Things Happen When You Start Changing Elements Outside The Boundaries Of Your Array!

• The Overflow Changes May, Or May Not, Destroy The Integrity Of Your Program → Sometimes The Changes Are To Memory Your Program Is Not Using!

• Your Program May Run Correctly One Time, But Not Another!

• You Can Actually Destroy Other Programs Running In Memory → Or Your Ability To Print, etc.
Size Your Containers Properly!
scanf inserts an EOS marker

stdio.h
Create a container, called \texttt{First8}, that has the capacity to store a 10 character name.

```c
char First8[11];
```

Use scanf to fill the first name \texttt{→ Don't forget to prompt!}

```c
printf("Enter First8: ");
scanf("%s", &First8);

printf("First8 = %s\n\n", First8);
```

\texttt{scanf might work \texttt{→ But it is a great idea to flush the buffer before hand!}}

```c
flush_stream(stdin); \ // or fflush(stdin);
```

\texttt{Visual Studio 2015/2017 \texttt{→} flush_stream(stdin)}
\texttt{Other C++ Compilers & Earlier Visual Studio \texttt{→} fflush(stdin);}
Create a container, called First8, that has the capacity to store a 10 character name.

```c
char
First8[11];
```

Use scanf to fill the first name → Don't forget to prompt!

```c
flush_stream(stdin);
printf("Enter First8: ");
scanf("%s", &First8);
printf("First8 = %s

", First8);
```

scanf is only good for reading the first token → At the moment, our buffer contains Jin & The Carriage Return → If we don't flush the buffer after the scanf, it will mess up the next scanf

```c
flush_stream(stdin);  // or fflush(stdin);
```
Create a container, called `First8`, that has the capacity to store a 10 character name. Create a container, called `Last8`, that has the capacity to store a 15 character name.

```c
char
    First8[11],
    Last8[16];
```

Use `scanf` to fill the `First` & `Last`

```c
flush_stream(stdin);
printf("Enter First8: ");
scanf("%s", &First8);

printf("Enter Last8: ");
scanf("%s", &Last8);

printf("First8 = %s\n\n", First8);
printf("Last8 = %s\n\n", Last8);
```

We need the flush before & after each `scanf` that does not solve the single token issue:

```c
Enter First8: Jo Jin
Enter Last8: First8 = Jo
Last8 = Jin
```

That does not solve the single token issue:

```c
Use gets_s
```
Create a container, called `First8`, that has the capacity to store a 10 character name. Create a container, called `Last8`, that has the capacity to store a 15 character name.

```c
char
First8[11],
Last8[16];
```

Use `scanf` to fill the first & last

```c
flush_stream(stdin);
printf("Enter First8: ");
scanf("%s", &First8);
flush_stream(stdin);
printf("Enter Last8: ");
scanf("%s", &Last8);
flush_stream(stdin);
printf("First8 = %s\n", First8);
printf("Last8 = %s\n\n", Last8);
```

That does not solve the single token issue:

Use `gets_s`
gets

inserts an EOS marker

string.h
Create a container, called First8, that has the capacity to store a 10 character name.

```c
char First8[11];
```

Use `gets` to fill the first name → Don't forget to prompt!

```c
flush_stream(stdin);
printf("Enter First8: ");
gets_s(First8);

printf("First8 = %s\n", First8);
```

gets might work → But it is a great idea to flush the buffer before hand!

Visual Studio 2015/2017 (safer) ➔ `gets_s(First8);`
Other C++ Compilers & Earlier Visual Studio ➔ `gets(First8);`
Create a container, called First8, that has the capacity to store a 10 character name. Create a container, called Last8, that has the capacity to store a 15 character name.

```c
char
    First8[11],
    Last8[16];

Use scanf to fill the First & Last

flush_stream(stdin);
printf("Enter First8: ");
gets_s(First8);

printf("Enter Last8: ");
gets_s(Last8);

printf("First8 = %s\n", First8);
printf("Last8 = %s\n\n", Last8);
```

I place a flush before the first gets in case someone else was careless with scanf.

Enter First8: Jo Jin
Enter Last8: Le
First8 = Jo Jin
Last8 = Le

We don't need flush before, or after other gets.

Gets solves the single token issue:

I almost always use gets_s instead of scanf.
strncpy

strcpy_s does not insert an EOS marker

string.h
Strncpy Can Help You Avoid String Overflow

Many Programs Are Interactive → Where We Ask The User For Data

We Can Use strncpy To Make Sure The User Does Not Overflow The Container.
Interactive Programming
Strncpy Can Help You Avoid String Overflow

On Professional Systems, I Will Read The User Response Into A Really Large Temporary Container → And Then I Use `strncpy` To Make Sure That The User Does Not Overflow My Original Container!

Sometimes I Create A Loop & Continue To Give The User The Opportunity To Re-Enter Their Choices Until They Finally Enter One That Fits! This Can Be Done With A Function That Can Be Reused Throughout The System.

Never Allow The User To Trash Your System Integrity!

```c
char University [12], Temp[500];
flush_stream(stdin);
printf("Enter University: ");
gets_s(Temp);
strncpy(University, Temp, 15);
University[15] = 0;
printf("University = %s\n\n", University);
```

Enter University: Golden State University
University = Golden State University

Enter University: Trinity University
University = Trinity University
char
University[12],
Temp[500];

strcpy_s(University, "ABCDEFGHIJKLMNOPQRSTUVWXYZ");
flush_stream(stdin);
printf("Enter University: ");
gets_s(Temp);
strncpy(University, Temp, 15);
University[15] = 0;
printf("University = %s\n\n", University);
for (int Pos = 0; Pos <= 15; Pos++)
printf("University[%d] = %c %d\n", Pos, University[Pos], University[Pos]);
strcmp

string.h
Fill Strings With Some Sample Data For Comparisons

```c
char
Str11[] = {'T', 'o', 'm', 0 },
Str12[] = {'D', 'i', 'c', 'k', '\0' },,
Str13[] = {'T', 'O', 'm', 0 },
Str14[] = {'T', 'o', 'm', 'y', 0 },,
Str15[] = {' ', 'T', 'o', 'm', 0 };

puts(Str11);
puts(Str12);
puts(Str13);
puts(Str14);
puts(Str15);
```

### Strcmp Match?

<table>
<thead>
<tr>
<th>str</th>
<th>cmp</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom</td>
<td>Tom</td>
<td>0</td>
</tr>
<tr>
<td>Tom</td>
<td>Dick</td>
<td>1</td>
</tr>
<tr>
<td>Tom</td>
<td>Zebra</td>
<td>-1</td>
</tr>
</tbody>
</table>

- `strcmp` → Returns 0 → If The Two Strings Match → Are The Same
- `strcmp` → Returns A Positive → If The First Is Greater
- `strcmp` → Returns A Negative → If The Second Is Greater
Fill Strings With Some Sample Data For Comparisons

char
Str11[] = { 'T', 'o', 'm', 0 },
Str12[] = { 'D', 'i', 'c', 'k', '\0' },
Str13[] = { 'T', 'O', 'm', 0 },
Str14[] = { 'T', 'o', 'm', 'm', 'y', 0 },
Str15[] = { ' ', 'T', 'o', 'm', 0 };
string
    st = "Tom";
cout << st.length() << endl;
cout << st.data() << endl;
st.append("Hicks");
cout << st.length() << endl;
cout << st.data() << endl;
st = "Donald Duck";
cout << st.length() << endl;
cout << st.data() << endl;