

Arrays, Type Casting and Constants

Engineering 1D04, Teaching
Session 5

High Score System

- We want to produce a program to keep track of the high scores in a video game.
- The top ten scores and the names of the players must be recorded in order.
- We must have a mechanism for inserting a new score and a mechanism for displaying the current scores.

High Score System

- Following the ideas from the ticket system example, write a high score system that allows the names and scores of the top ten players to be saved.
- Provide a method to add a new score and a method to print the current high scores.

remember: Ticket Example

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    int ticket;

    void reset()
    {
        ticket = 0;
    }

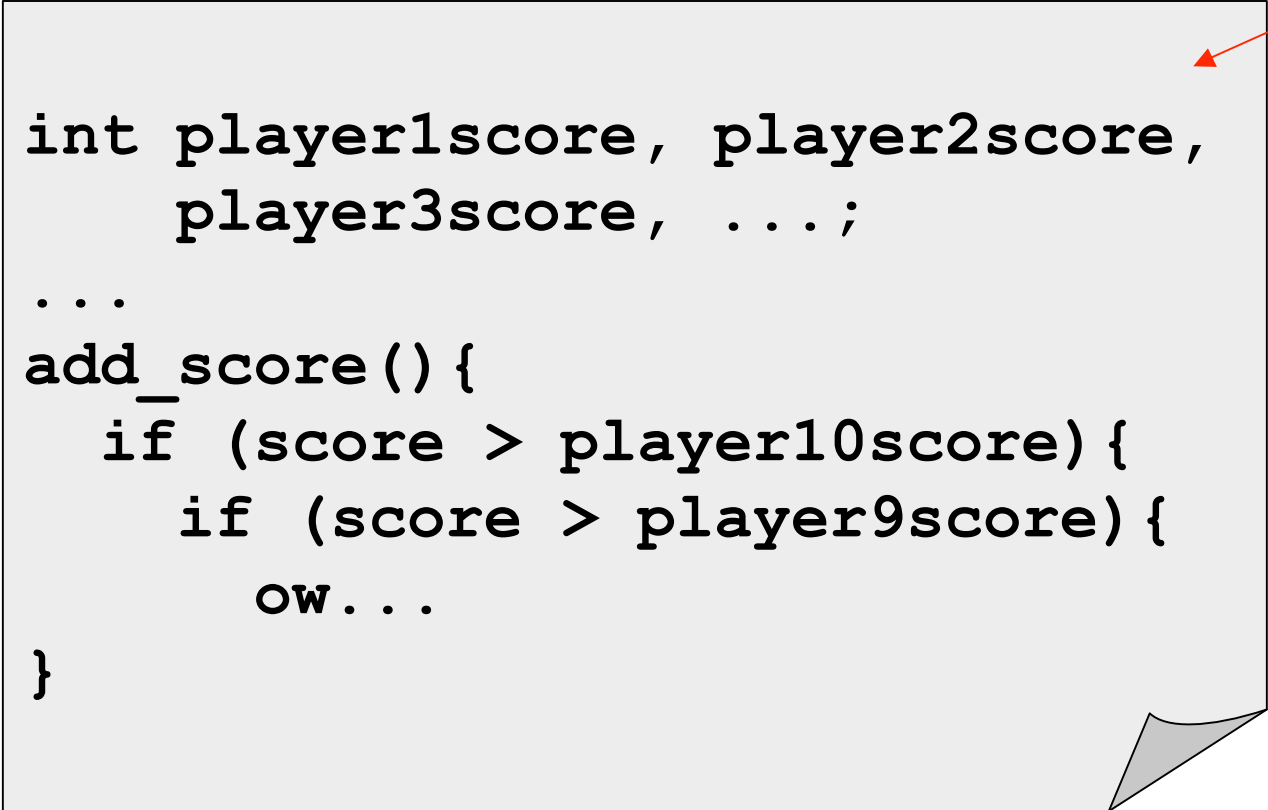
    int getTicket()
    {
        ticket = ticket+1;
        return ticket;
    }
}
```

Plan a high score
system ...

High Score System

- Not so easy ...

maybe it looked something like this



```
int player1score, player2score,  
    player3score, ...;  
...  
add_score() {  
    if (score > player10score) {  
        if (score > player9score) {  
            ow...  
        }  
    }
```

High Score System

```
print_scores()  
{  
    ...  
    "score1: " + player1score +  
    "score2: " + player2score +  
    "score3: " + player3score +  
    ...  
}
```

- Aren't computers supposed to make it easier to deal with large amounts of similar information?

Introduction to Arrays

- Arrays are a method of storing a number of similar items in a way that is easy to access.
- As an example, let us see how we could find the sum of a number of integers.

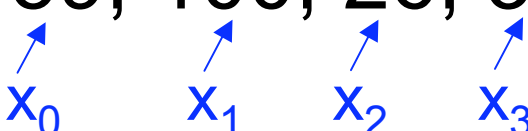
Introduction to Arrays

- Consider the following numbers:

We can name
those numbers:

59, 100, 26, 35

x_0 x_1 x_2 x_3



- (This would form an *array* x , with elements x_0 , x_1 , x_2 , x_3 and **each element must be the same *type* of number.**)
- Once we have named them like this how do we add them?

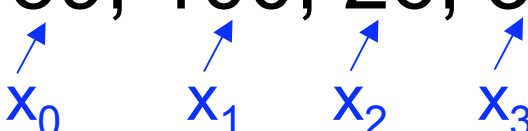
Introduction to Arrays

- Consider the following numbers:

We can name
those numbers:

59, 100, 26, 35

x_0 x_1 x_2 x_3



- Once we have named them like this how do we add them?

$s = 0$

for $i = 0, 1, 2, 3$

└ $s = s + x_i$

Introduction to Arrays

- Consider the following numbers:

We can name
those numbers:

59, 100, 26, 35

x_0 x_1 x_2 x_3

- Once we have named them like this how do we add them?

$s = 0$

for $i = 0, 1, 2, 3$

↑ $s = s + x_i$

```
s = 0;
for (i=0; i<=3; i++)
{
    s = s+x[i];
}
```

Introduction to Arrays

- Consider the following numbers:

We can name
those numbers:

59, 100, 26, 35

x_0 x_1 x_2 x_3

- Once we have named them like this how do we add them?

$s = 0$

for $i = 0, 1, 2, 3$

$s = s + x_i$

```
s = 0;
```

```
for (i=0; i<=3; i++)
```

```
{
```

```
    s = s+x[i];
```

```
}
```

index in square
brackets

Introduction to Arrays

Indicates the variable is an array

```
int [] x = {59, 100, 26, 35};  
int i, s;  
double ave;
```

We can declare an array with values for each element in the array

```
s = 0;  
for (i=0; i<=3; i++)  
{  
    s = s+x[i];  
}
```

The elements in the array are automatically numbered, starting from 0

Introduction to Arrays

```
int [] x = {59, 100, 26, 35};  
int i, s;  
double ave;
```

```
s = 0;  
for (i=0; i<=3; i++)  
{  
    s = s+x[i];  
}
```

do we really have to count
the elements in the array?

Introduction to Arrays

```
int [] x = {59, 100, 26, 35};  
int i, s;  
double ave;  
  
s = 0;  
for (i = 0; i < x.Length; i++)  
{  
    s = s+x[i];  
}
```

do we really have to count the elements in the array? No

Introduction to Arrays

- How do we declare the array if we are not initializing the values in the array?

```
int [] x = new int[10];
```

we will discuss “new” in
some detail later

maximum elements in
the array

remember the index
number starts from 0

what do you think x.Length returns now?

Introduction to Arrays

- How do we declare the array if we are not initializing the values in the array?

```
int [] x = new int[10];
```

we will discuss “new” in
some detail later

maximum elements in
the array

remember the index
number starts from 0

what do you think x.Length returns now? **10 = the max number of
elements in x**

recap: Arrays

- Arrays are a method of storing a number of *similar* items in a way that is easy to access.

```
int [] score = new int[10];  
string [] name = new string[10];
```

- Accessing an item in the array is easy - just put its *index* number in the square brackets.

```
score[5] = 1000;  
name[43] = "alan";
```

recap: Arrays

- You can also use an integer variable to access a specific element in an array.

```
int [] score = new int[10];  
int i = 5;  
score[i] = 1000;
```

- Using a variable to select an element from an array is very useful when combined with loops.

recap: Arrays

- Recall an example from an earlier session:

```
double b=25, c=30, d=60, e=10,  
       sum, average;  
int numTerms = 4;  
  
sum = b;  
sum = sum + c;  
sum = sum + d;  
sum = sum + e;  
average = sum / numTerms;
```

Take a few minutes
now to do this using
an array instead of
b,c,d and e.

In other words, use
an array called *terms*
instead of b,c,d,e.

recap: Arrays

- Using an array (each element is a double)

```
double [] terms = {25.0, 30.0, 60.0, 10.0};  
double sum, average;  
int i;
```

make sure you understand why this is not <=

```
sum = 0.0;  
for (i = 0; i < terms.Length; i++)  
    sum = sum + terms[i];  
average = sum / terms.Length;
```

is this okay if we want an answer
that is a double?

recap: Arrays

- Using an array (each element is a double)

```
double [] terms = {25.0, 30.0, 60.0, 10.0};  
double sum, average;  
int i;
```

make sure you understand why this is not <=

```
sum = 0.0;  
for (i = 0; i < terms.Length; i++)  
    sum = sum + terms[i];  
average = sum / terms.Length;
```

is this okay if we want an answer
that is a double? What if terms and sum
are integers?

Aside: Type Casting

Type Casting

- Now each element is an integer

```
int [] terms = {25, 30, 60, 10};  
double average;  
int i, sum;  
  
sum = 0;  
for (i = 0; i < terms.Length; i++)  
    sum = sum + terms[i];  
average = (double)sum / terms.Length;
```

this is called *type casting*

at least one of the values must be a double
or the computer will perform an integer division!

Implicit Versus Explicit Casting

- In C# $1/3 = 0$ and $1.0/3.0 = 0.33333$
- What does $1.0/3$ equal?

Implicit Versus Explicit Casting

- In C# $1/3 = 0$ and $1.0/3.0 = 0.33333$
- $1.0/3$ equals 0.33333
- The reason is that the 3 is implicitly promoted to be a double before the division operation
- It is best to not rely on the promotion rules
- For all literals that could be floating point numbers, use .0 on the end

Back to Arrays

recap: Arrays

- Use an array (each element is a double)


```
double [] terms = {25.0, 30.0, 60.0, 10.0};  
double sum, average;  
int i;  
  
sum = 0.0;  
for (i = 0; i < terms.Length; i++)  
    sum += terms[i];  
average = sum / terms.Length;
```

more shorthand

High Score System

- Using an array for our high score system would vastly simplify the situation.

this is how we declare a
constant



```
const int numScores = 10;  
  
int [] score = new int[numScores];  
string [] name = new string[numScores];
```

It is much better to use a constant for the maximum size of the array rather than a magic literal. why?

Aside: Constants

Constants

- Constants do not change
- Equates a value with a name
- Improve readability by using the symbolic name throughout the program
- Change value in only one place
- A loop over numScores is easier to read and to modify than a loop over a hard coded literal
- Constants often use ALL_CAPS

Back to Arrays

High Score System

- Assuming that we have entered/calculated the high scores, how could we print out all of them?

High Score System

- How could we print out all the high scores?

```
void print_scores()  
{  
    int i;  
    string result = "";  
    for (i = 0; i < score.Length; i++)  
        result = result + "score[" + (i + 1) +  
            "]" + " = " + score[i] + " - " +  
            name[i] + "\n";  
  
    MessageBox.Show(result);  
}
```

what will the output
look like?

High Score System

- How do we enter the high scores into the array so that they are in the order we want?

High Score System

- Assume the *name* and *score* arrays hold the following values?

| | |
|---------|------|
| Alice | 7000 |
| Bob | 4000 |
| Charlie | 2000 |
| Dave | 1000 |

High Score System

- Assume the name and score arrays hold the following values?

| | |
|---------|------|
| Alice | 7000 |
| Bob | 4000 |
| Charlie | 2000 |
| Dave | 1000 |

Now add this new entry:
Eve 5000

High Score System

- Assume the name and score arrays hold the following values?

| | |
|---------|------|
| Alice | 7000 |
| Bob | 4000 |
| Charlie | 2000 |
| Dave | 1000 |

← 5000 > 7000? No.

Now add this new entry:
Eve 5000

High Score System


- Assume the name and score arrays hold the following values?

| | | | |
|---|---------|------|------------------------------|
| → | Alice | 7000 | |
| | Bob | 4000 | ← 5000 > 4000? Yes. Mark it. |
| | Charlie | 2000 | |
| | Dave | 1000 | |

Now add this new entry:
Eve 5000

High Score System

- Assume the name and score arrays hold the following values?



| | |
|---------|------|
| Alice | 7000 |
| Bob | 4000 |
| Charlie | 2000 |
| Dave | 1000 |


Move the last entry 1 *down*
in the array



Now add this new entry:
Eve 5000

High Score System

- Assume the name and score arrays hold the following values?




| | |
|---------|------|
| Alice | 7000 |
| Bob | 4000 |
| Charlie | 2000 |
| Dave | 1000 |

Then move the second last entry 1 *down* in the array

Now add this new entry:
Eve 5000

High Score System

- Assume the name and score arrays hold the following values?



| | |
|---------|------|
| Alice | 7000 |
| Bob | 4000 |
| Charlie | 2000 |
| Dave | 1000 |

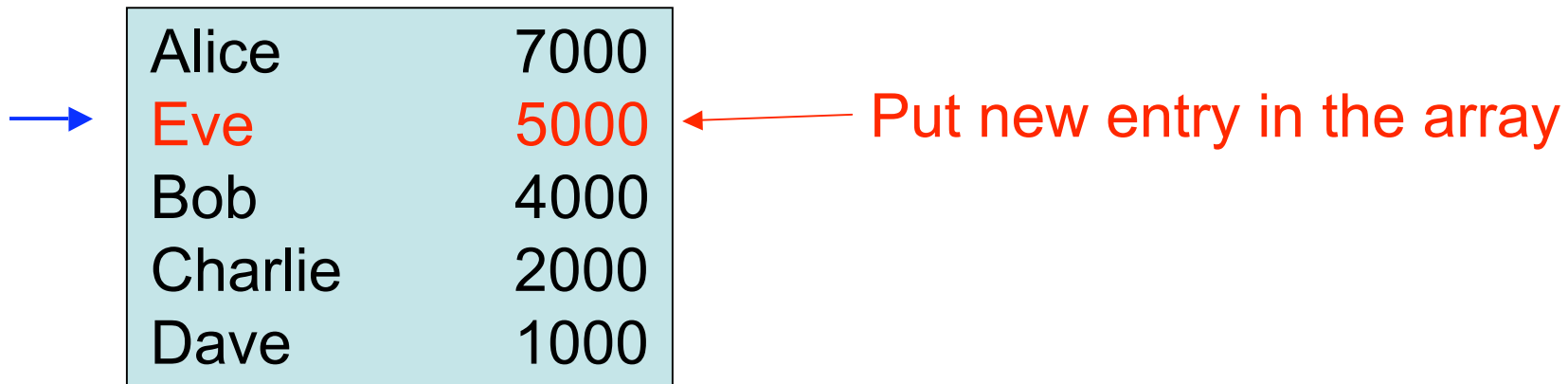


And eventually move the entry
identified 1 *down* in the array

Now add this new entry:
Eve 5000

High Score System

- Assume the name and score arrays hold the following values?



A diagram illustrating a high score system. A light blue rectangular box contains a table of names and scores. The table has five rows: Alice (7000), Eve (5000), Bob (4000), Charlie (2000), and Dave (1000). The name 'Eve' and her score '5000' are highlighted in red. A blue arrow points to the left of the table, and a red arrow points from the text 'Put new entry in the array' to the red '5000'.

| | |
|---------|------|
| Alice | 7000 |
| Eve | 5000 |
| Bob | 4000 |
| Charlie | 2000 |
| Dave | 1000 |

Now add this new entry:
Eve 5000

High Score System

■ Algorithm

- Determine where the new score belongs.
 - Start looking from the top until we find a lower score and mark that index.
- Each score from that index to the end is shifted down one element to make room. It is important to start this at the end of the array. The bottom score “falls off” if the array is already full.
- The new score is inserted at the identified index.

High Score System

- Add a new entry - newScore, newName:
 - Find where it fits
 - mark=0
 - while (mark \leq lastElement and score_{mark} \geq newScore)
 - ↑ mark = mark+1
 - If at end of list don't do anything, otherwise ...
 - if (mark \leq lastElement)
 - ↑ for j = lastElement *downto* mark+1
 - ↑ score_j = score_{j-1}
 - ↑ name_j = name_{j-1}
 - score_{mark} = newScore
 - name_{mark} = newName

High Score System

```
void add_score(int newScore, string newName)
{
    int mark;           //index of "marked" entry
    int j;              //index used to move entries down
    mark = 0;
    while (mark <= score.Length-1 &&
           score[mark] >= newScore) mark++;
    if (mark <= score.Length-1) //only if new is better
    {
        for (j = score.Length-1; j > mark; j--)
        {
            score[j] = score[j-1];
            name[j] = name[j-1];
        }
        score[mark] = newScore;
        name[mark] = newName;
    }
}
```

High Score System

```
void add_score(int newScore, string newName)
{
    int mark;           //index of "marked" entry
    int j;              //index used to move entries down
    mark = 0;
    while (mark <= score.Length-1 &&
           score[mark] >= newScore) mark++;
    if (mark <= score.Length-1) //only if new is better
    {
        for (j = score.Length-1; j > mark; j--)
        {
            score[j] = score[j-1];
            name[j] = name[j-1];
        }
        score[mark] = newScore;
        name[mark] = newName;
    }
}
```

Use print_scores and add_score in a program that will store a name and score on a form, and then add the name / score if a button is clicked, and show current scores if another button is clicked