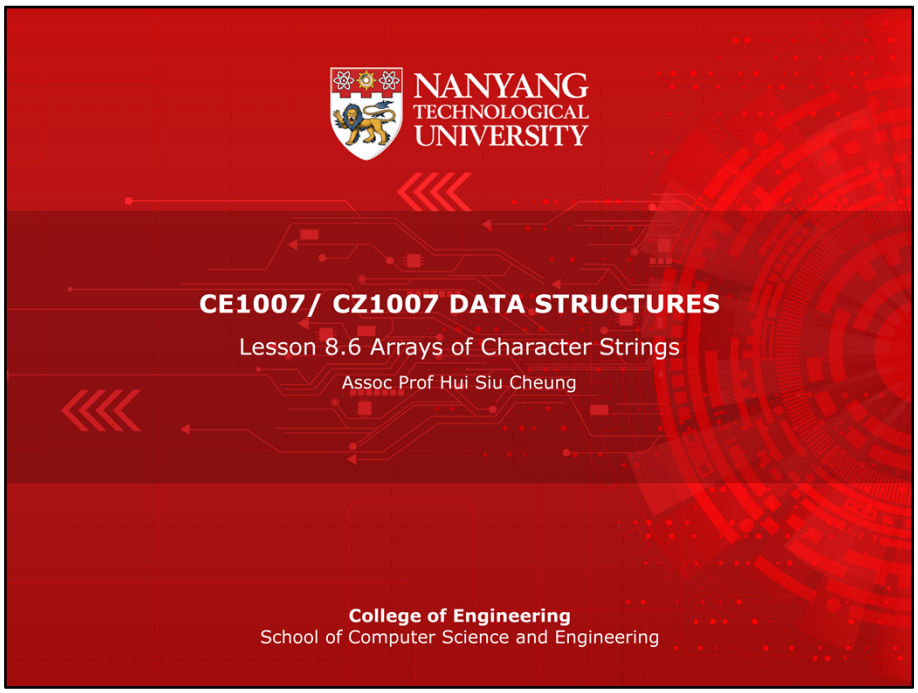
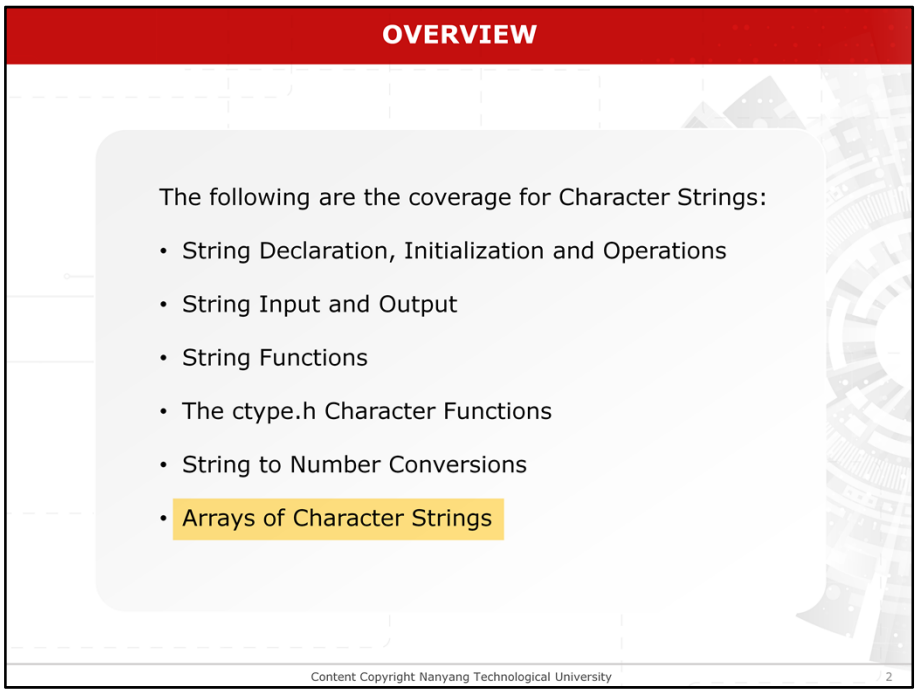
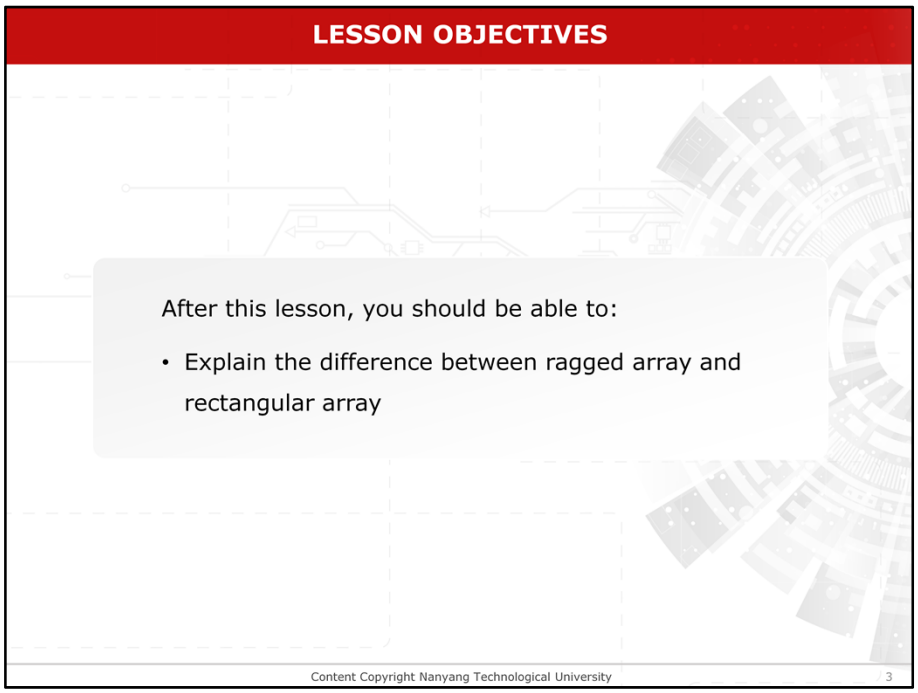


Lesson 8.6 Arrays of Character Strings

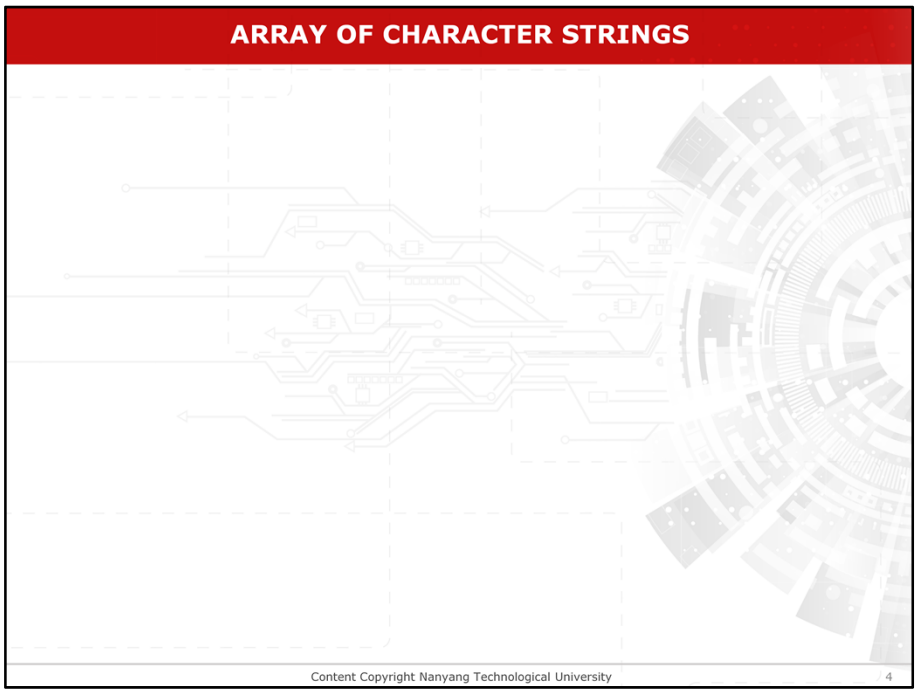


There are 6 main sections to cover for Character Strings as shown. This video lesson focuses on the last part on Arrays of Character Strings.



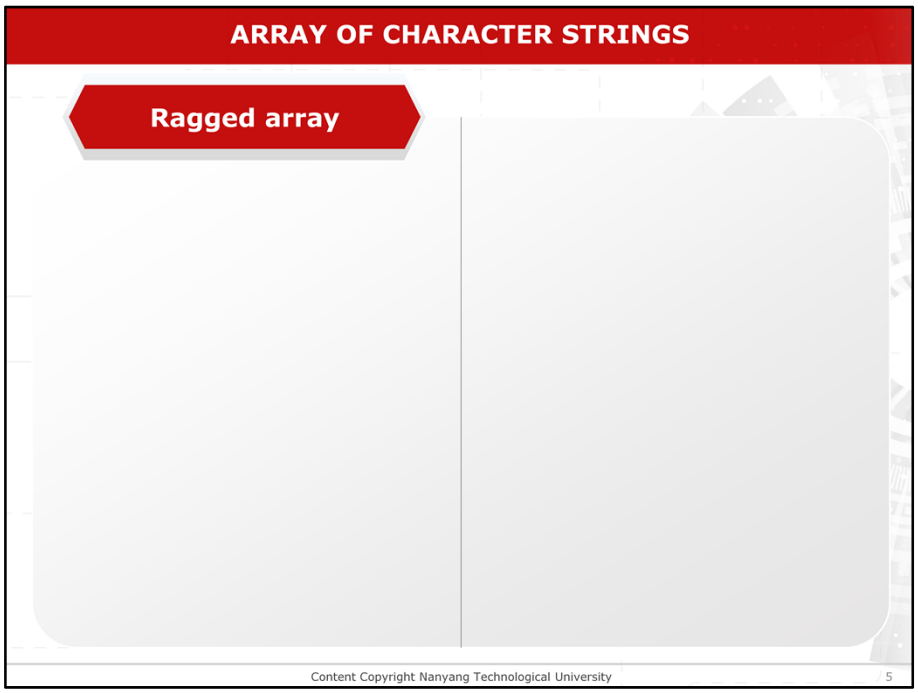
After this lesson, you should be able to:

Explain the difference between ragged array and rectangular array

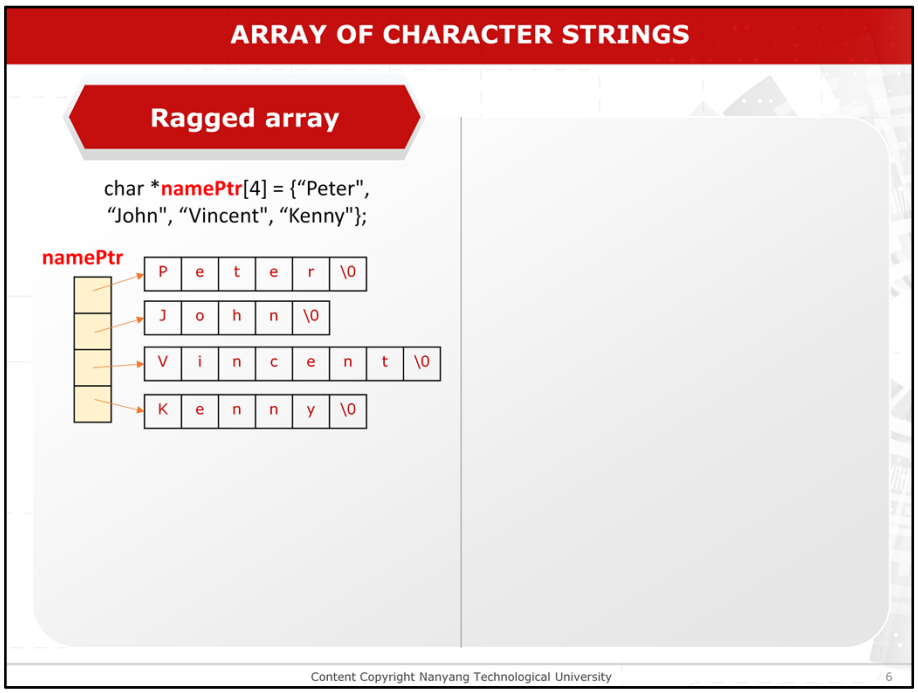


Array of character Strings

It is possible to define an array of character strings using ragged array and rectangular array. Let's compare both of them,



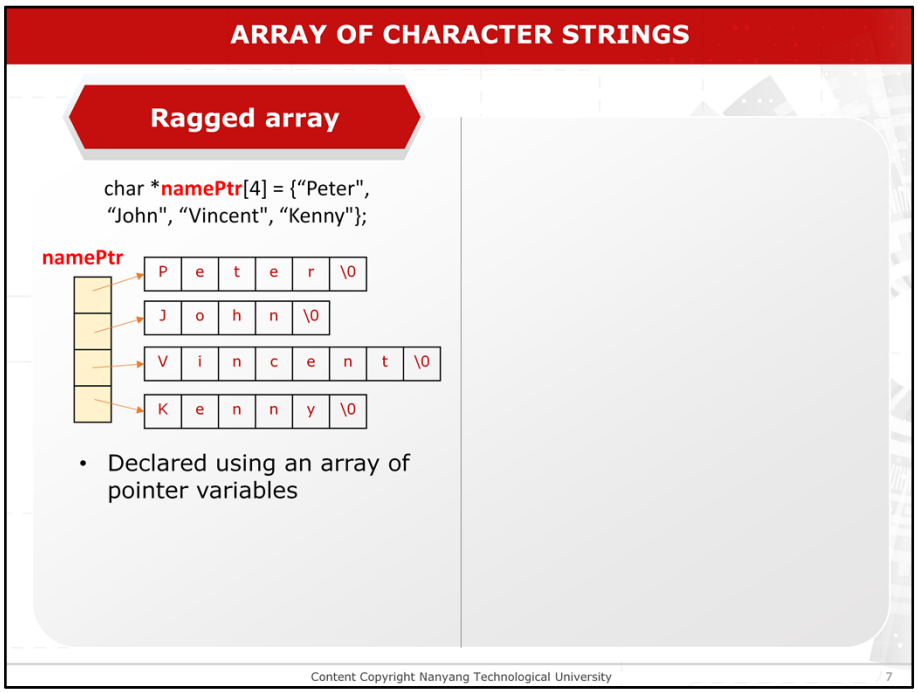
Ragged Array



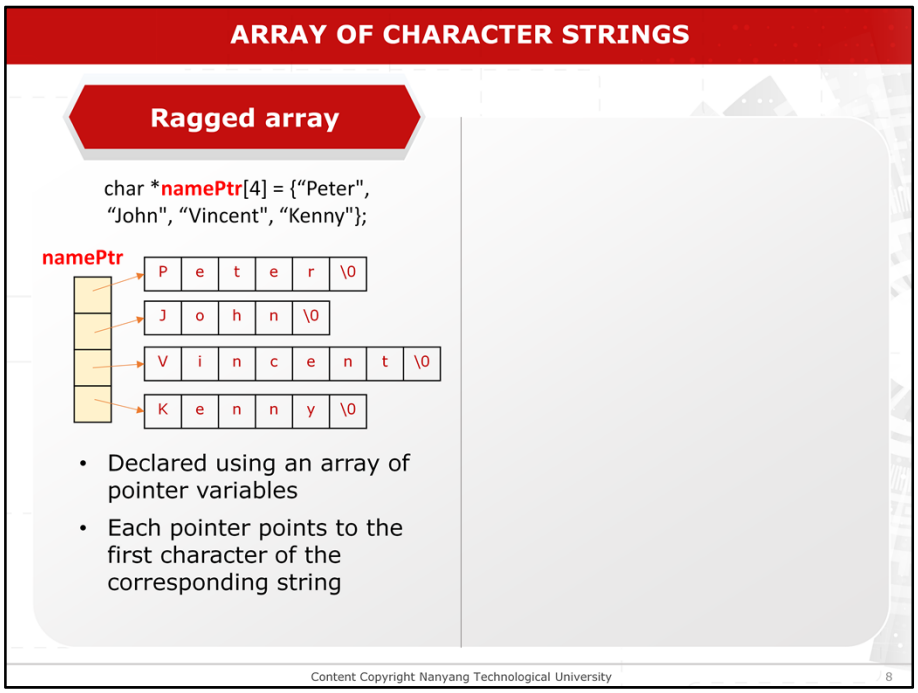
For example, we can declare

character asterisk name Pointer [4] = {"Peter", "John", "Vincent", "Kenny"};

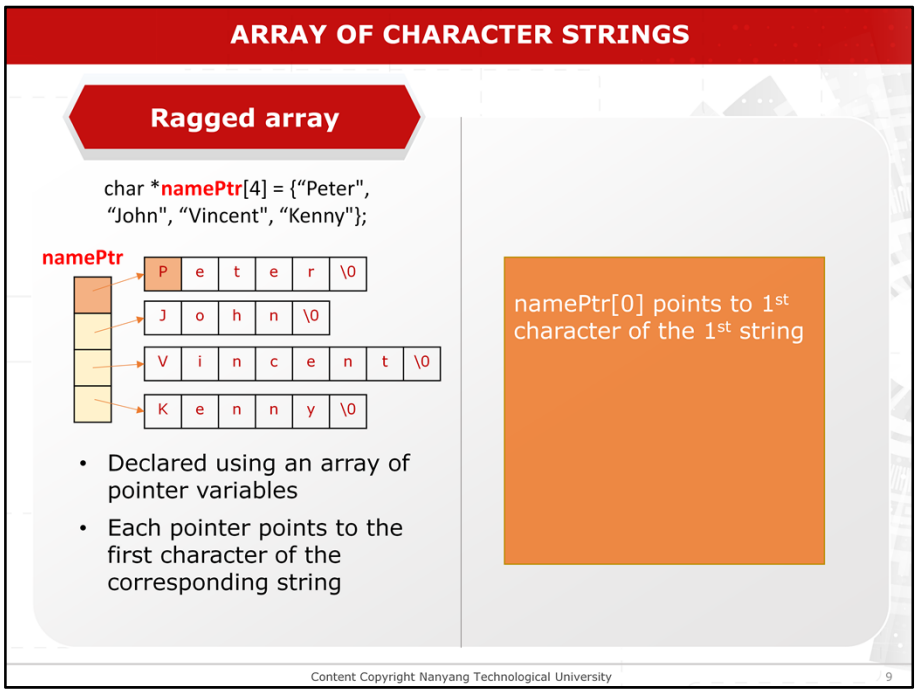
where **name Pointer** is a one-dimensional array of four pointers to type **character**.



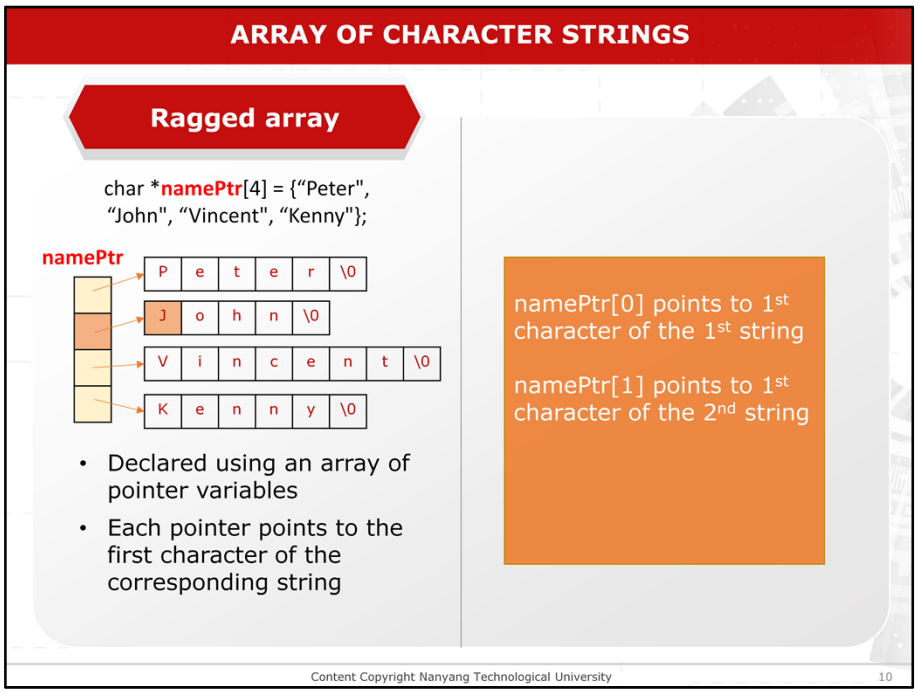
Ragged array is declared using an array of pointer variables



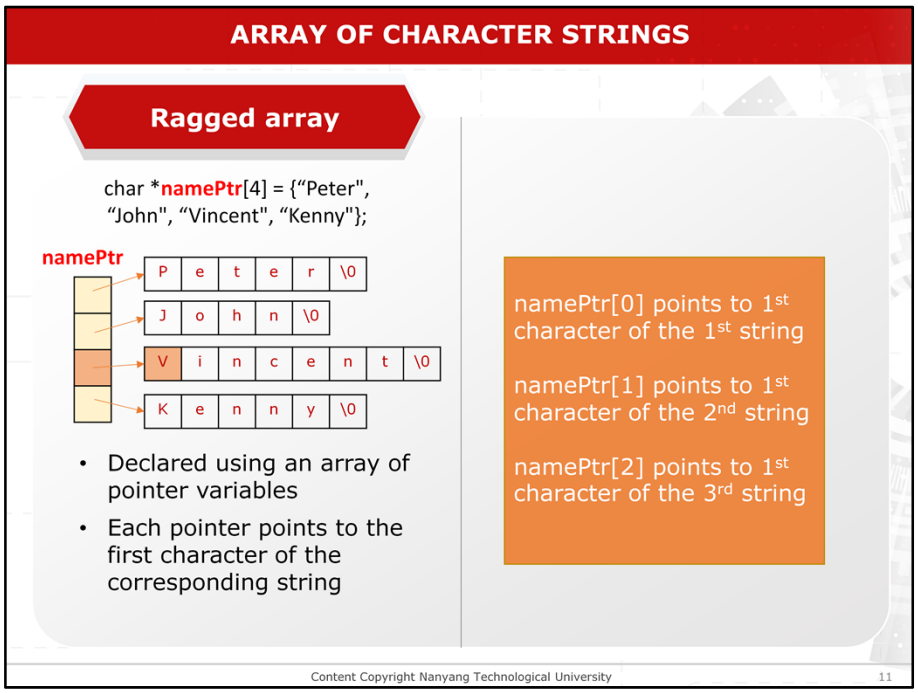
Each pointer points to the first character of the corresponding string.



Name Pointer[0] points to 1st character of the 1st string



Name Pointer[1] points to 1st character of the 2nd string.



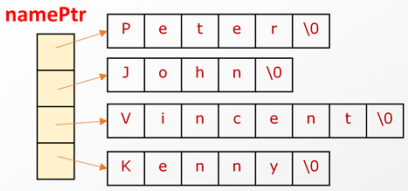
Name Pointer[2] points to 1st character of the 3rd string and so on.

ARRAY OF CHARACTER STRINGS

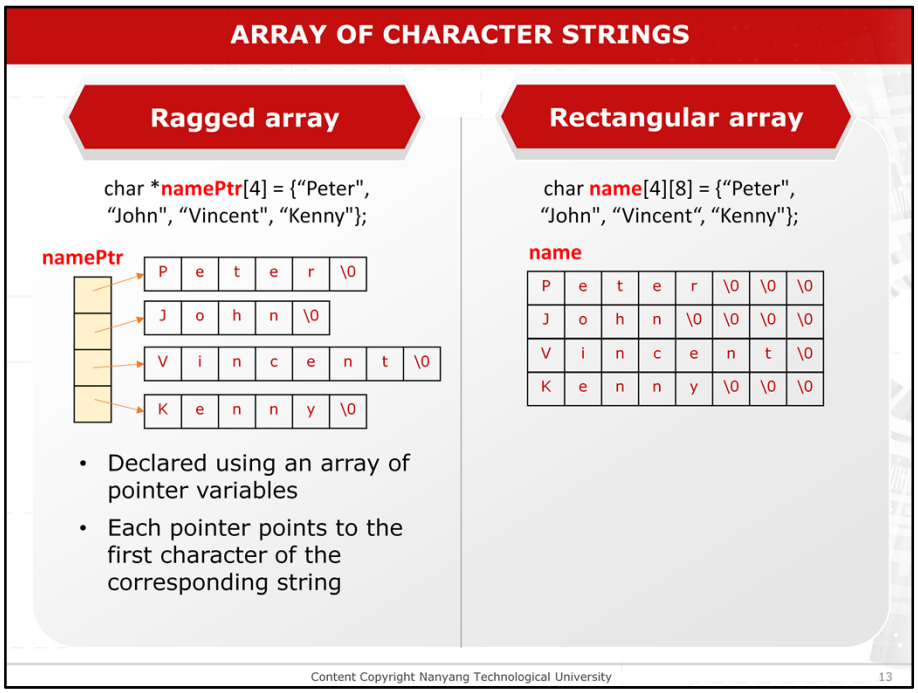
Ragged array

Rectangular array

char ***namePtr**[4] = {“Peter", “John", “Vincent", “Kenny"};

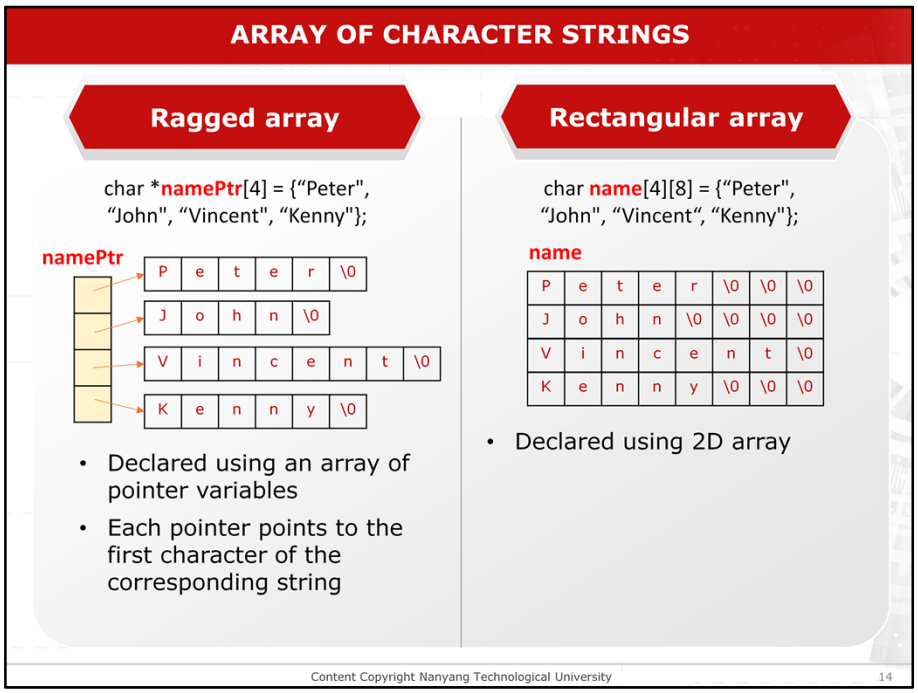


- Declared using an array of pointer variables
- Each pointer points to the first character of the corresponding string
- Help save storage space

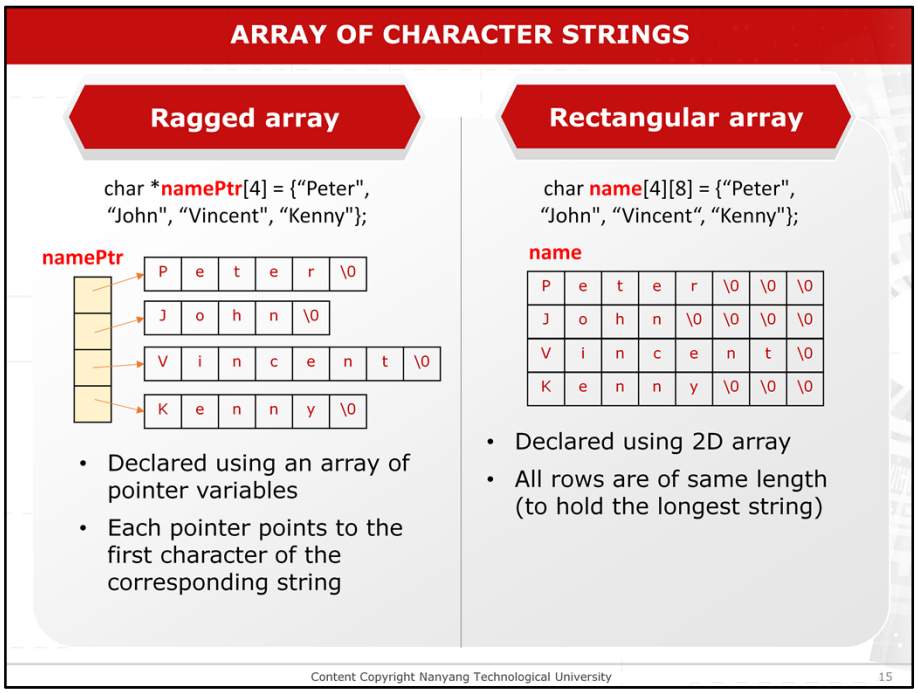


For example, we can declare **name** as

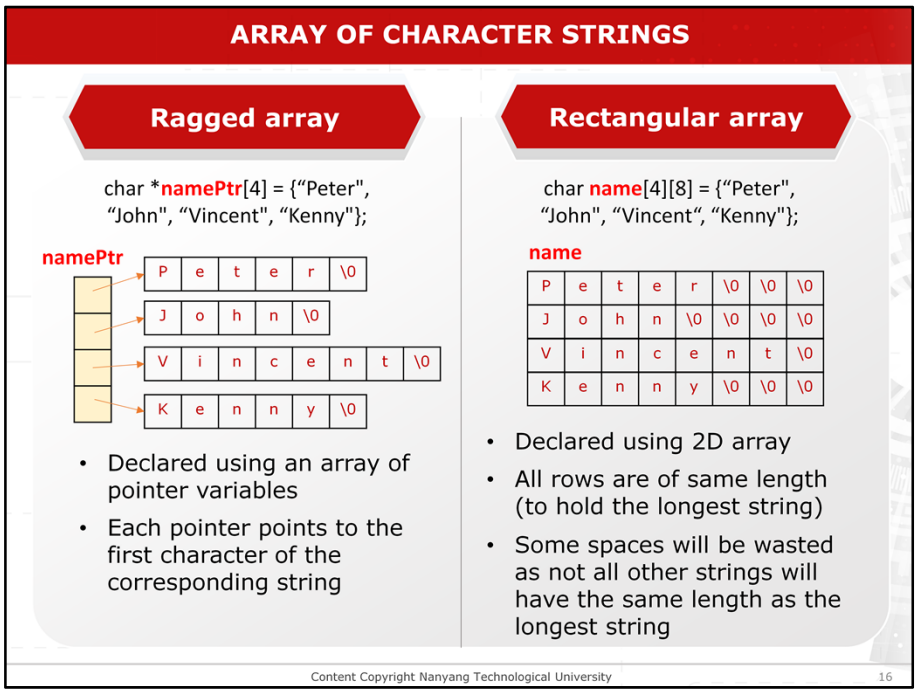
2D array character name of size [4] [8] = {"Peter", "John", "Vincent", "Kenny"};



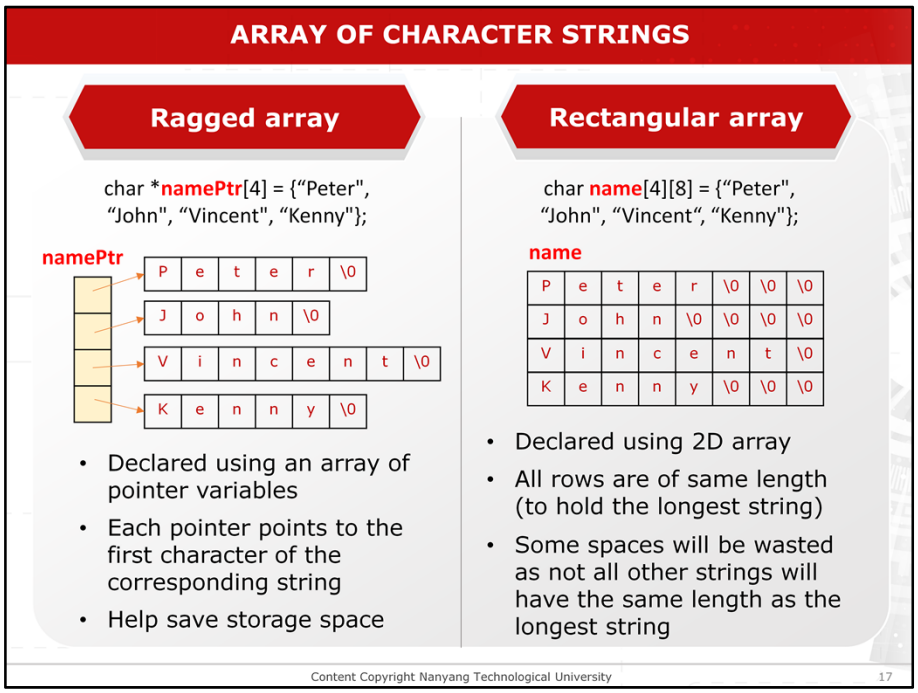
Rectangular array is declared using a two-dimensional array.



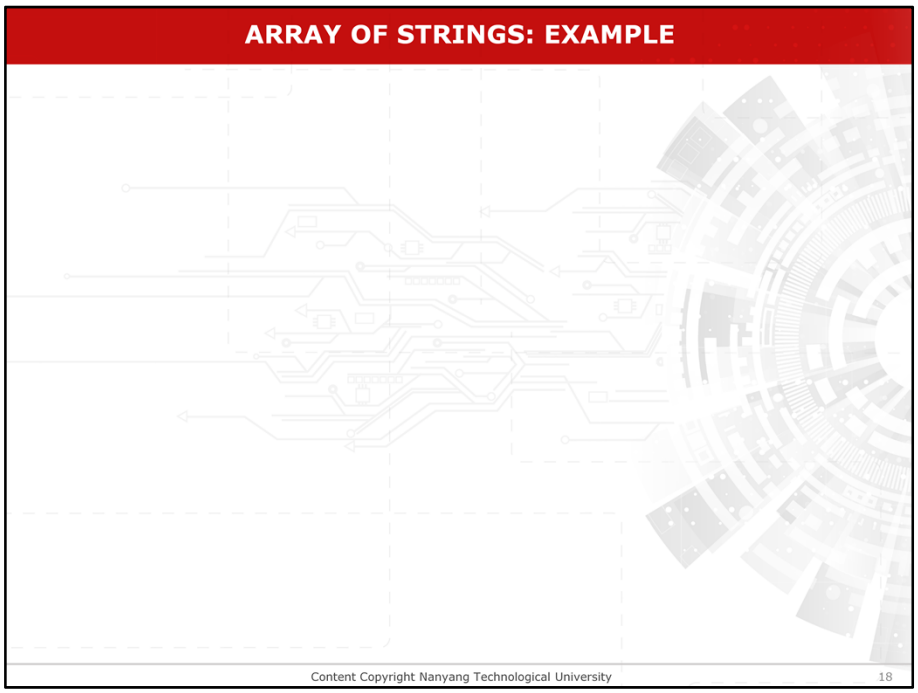
All the rows are of the same length. The two-dimensional array **name** needs to be defined with enough storage space to hold the longest string.



Thus, some space will be wasted as not all the other strings will have the same length as the longest string



However, the ragged array declaration **name Pointer** can help save storage space when compared with the rectangular array declaration.



Array of Strings: Example

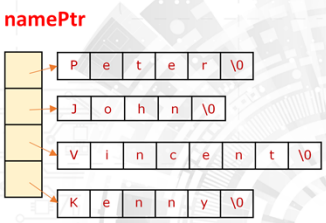
ARRAY OF STRINGS: EXAMPLE

```
#include <stdio.h>
int main()
{
    char *nameptr[4] = {"Peter", "John", "Vincent", "Kenny"};
    char name[4][8] = {"Mary", "Jane", "Susan", "May"};
    int  i, j;

    printf("Ragged Array: \n");
    for (i=0; i<4; i++)
      printf("nameptr[%d] = %s\n", i,
            nameptr[i]);

    printf("Rectangular Array: \n");
    for (j=0; j<4; j++)
      printf("name[%d] = %s\n", j,
          name[j]);
    return 0;
}
```


In the program, we can process array of strings declared using ragged array or rectangular array.

ARRAY OF STRINGS: EXAMPLE

```
#include <stdio.h>
int main()
{
   char *nameptr[4] = {"Peter", "John", "Vincent", "Kenny"};
   char name[4][8] = {"Mary", "Jane", "Susan", "May"};
   int  i, j;

   printf("Ragged Array: \n");
   for (i=0; i<4; i++)
     printf("nameptr[%d] = %s\n", i,
            nameptr[i]);

   printf("Rectangular Array: \n");
   for (j=0; j<4; j++)
     printf("name[%d] = %s\n", j,
          name[j]);
   return 0;
 }
```

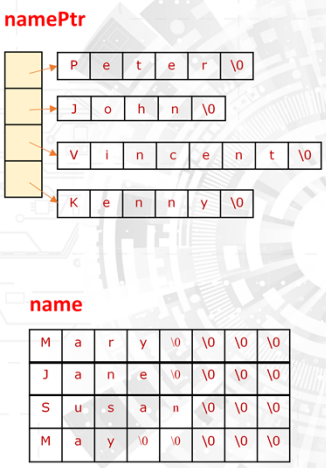


We can declare an array of strings using ragged array

character asterisk name Pointer [4] = {"Peter", "John", "Vincent", "Kenny"};

ARRAY OF STRINGS: EXAMPLE

```
#include <stdio.h>
int main()
{
    char *nameptr[4] = {"Peter", "John", "Vincent", "Kenny"};
    char name[4][8] = {"Mary", "Jane", "Susan", "May"};
    int  i, j;

    printf("Ragged Array: \n");
    for (i=0; i<4; i++)
      printf("nameptr[%d] = %s\n", i,
            nameptr[i]);

    printf("Rectangular Array: \n");
    for (j=0; j<4; j++)
      printf("name[%d] = %s\n", j,
          name[j]);
    return 0;
}
```



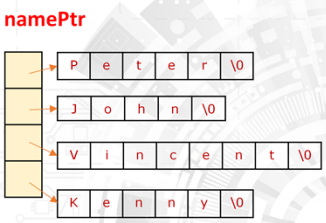
To declare an array of strings using rectangular array, we use
2D array character **name of size** [4] [8] = {"Mary", "Jane", "Susan", "May"};

ARRAY OF STRINGS: EXAMPLE

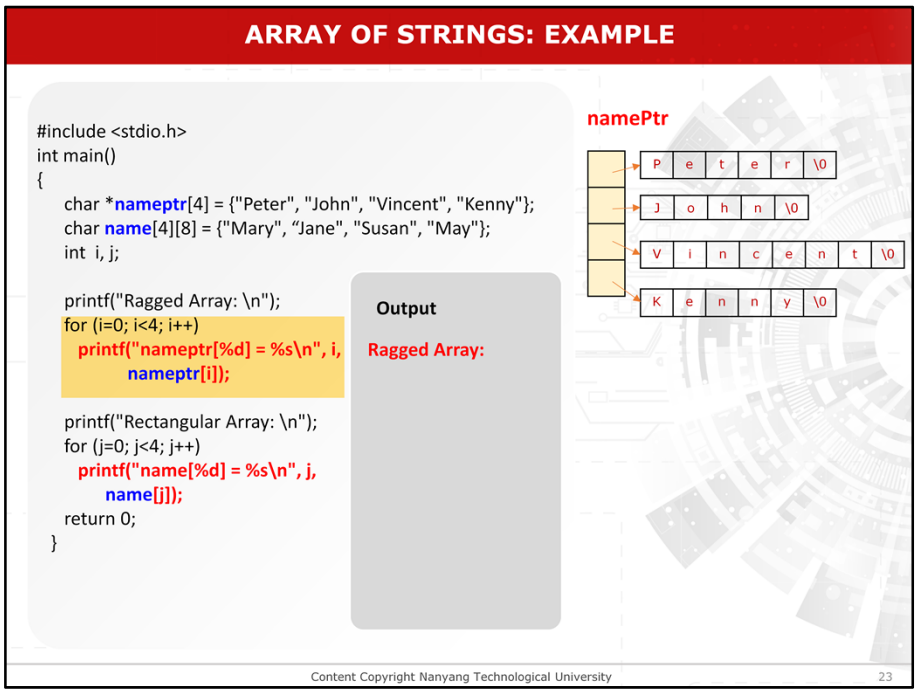
```
#include <stdio.h>
int main()
{
    char *nameptr[4] = {"Peter", "John", "Vincent", "Kenny"};
    char name[4][8] = {"Mary", "Jane", "Susan", "May"};
    int  i, j;

    printf("Ragged Array: \n");
    for (i=0; i<4; i++)
      printf("nameptr[%d] = %s\n", i,
            nameptr[i]);

    printf("Rectangular Array: \n");
    for (j=0; j<4; j++)
      printf("name[%d] = %s\n", j,
          name[j]);
    return 0;
}
```

Output

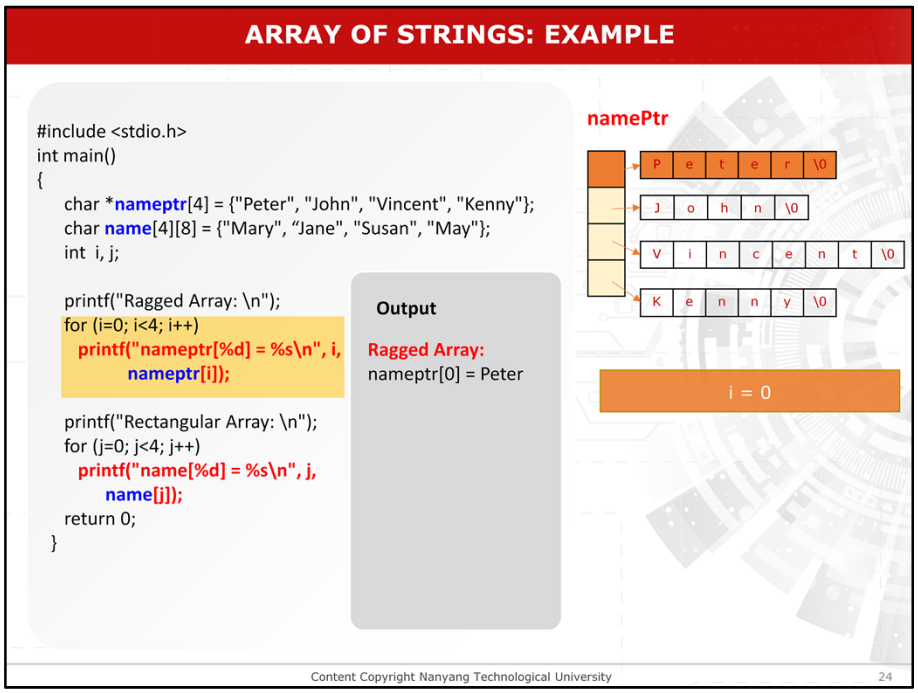
```
Ragged Array:
```



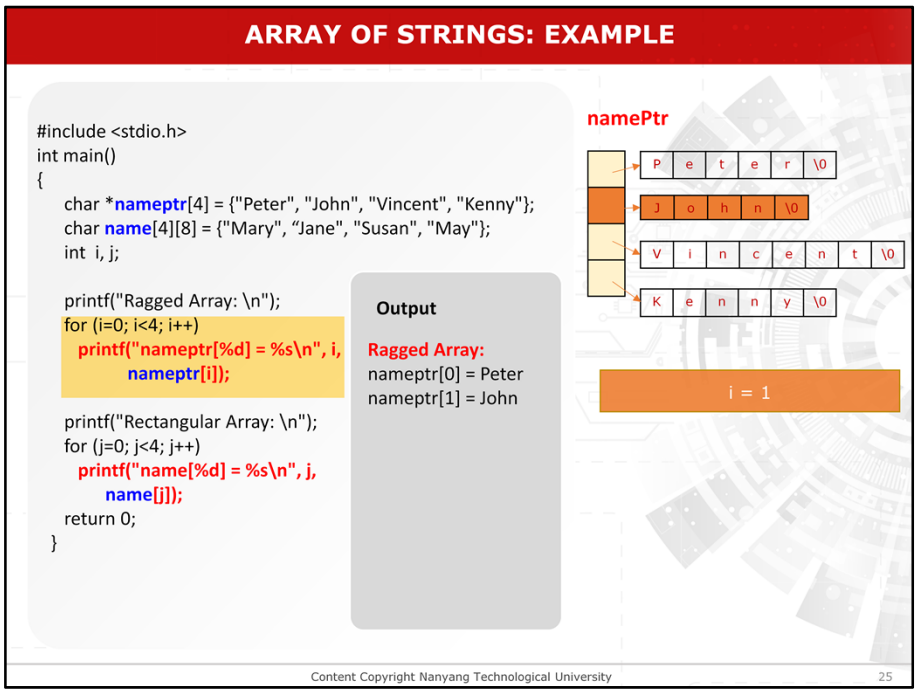
Let's look at the ragged array first.



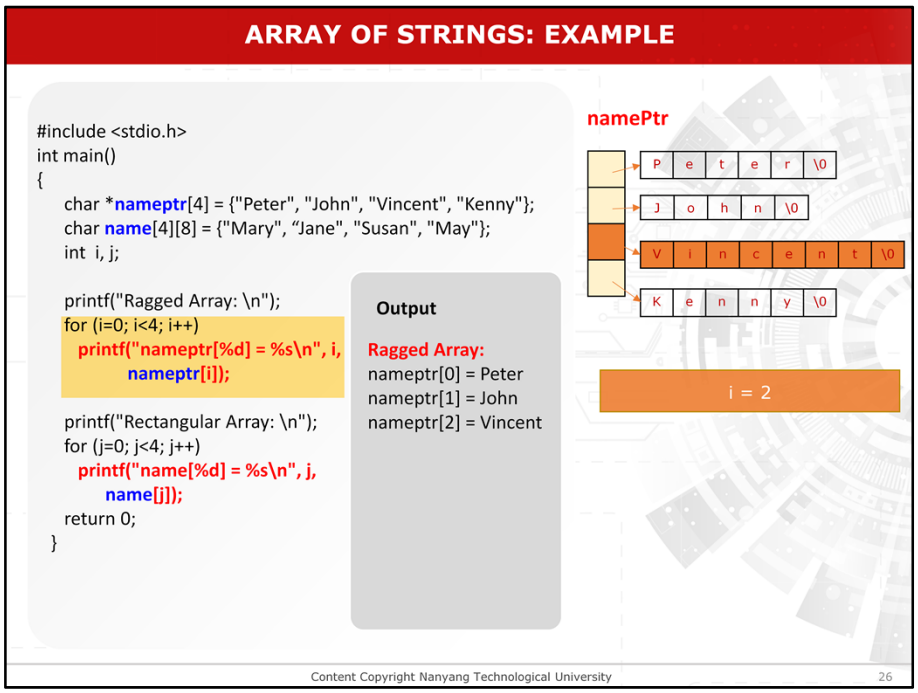
we can use an index (or subscript) to access each element of the array for the character strings


ARRAY OF STRINGS: EXAMPLE
#include <stdio.h>
int main()
{
char *nameptr[4] = {"Peter", "John", "Vincent", "Kenny"};
char name[4][8] = {"Mary", "Jane", "Susan", "May"};
int i, j;
printf("Ragged Array: \n");
for (i=0; i<4; i++)
printf("nameptr[%d] = %s\n", i,
nameptr[i]);
printf("Rectangular Array: \n");
for (j=0; j<4; j++)
printf("name[%d] = %s\n", j,
name[j]);
return 0;
}
Output
Ragged Array:
nameptr[0] = Peter
namePtr
P e t e r \0
J o h n \0
V i n c e n t \0
K e n n y \0
i = 0
Content Copyright Nanyang Technological University
24

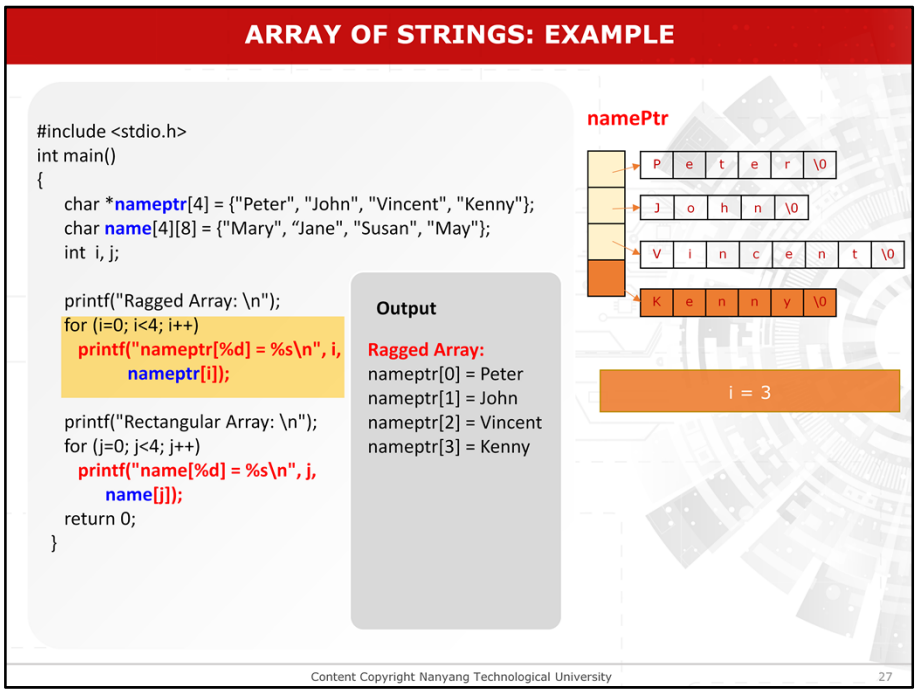
When I = 0


ARRAY OF STRINGS: EXAMPLE
#include <stdio.h>
int main()
{
char *nameptr[4] = {"Peter", "John", "Vincent", "Kenny"};
char name[4][8] = {"Mary", "Jane", "Susan", "May"};
int i, j;
printf("Ragged Array: \n");
for (i=0; i<4; i++)
printf("nameptr[%d] = %s\n", i,
nameptr[i]);
printf("Rectangular Array: \n");
for (j=0; j<4; j++)
printf("name[%d] = %s\n", j,
name[j]);
return 0;
}
Output
Ragged Array:
nameptr[0] = Peter
nameptr[1] = John
namePtr
P e t e r \0
J o h n \0
V i n c e n t \0
K e n n y \0
i = 1
Content Copyright Nanyang Technological University
25

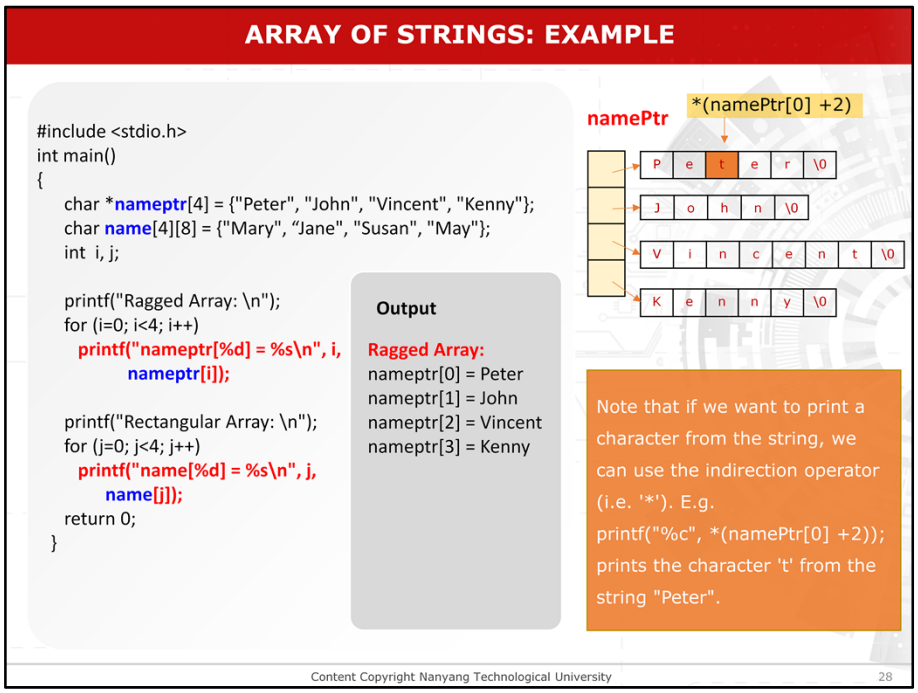
When i=1


ARRAY OF STRINGS: EXAMPLE
#include <stdio.h>
int main()
{
char *nameptr[4] = {"Peter", "John", "Vincent", "Kenny"};
char name[4][8] = {"Mary", "Jane", "Susan", "May"};
int i, j;
printf("Ragged Array: \n");
for (i=0; i<4; i++)
printf("nameptr[%d] = %s\n", i,
nameptr[i]);
printf("Rectangular Array: \n");
for (j=0; j<4; j++)
printf("name[%d] = %s\n", j,
name[j]);
return 0;
}
Output
Ragged Array:
nameptr[0] = Peter
nameptr[1] = John
nameptr[2] = Vincent
namePtr
P e t e r \0
J o h n \0
V i n c e n t \0
K e n n y \0
i = 2
Content Copyright Nanyang Technological University
26

When i=2


ARRAY OF STRINGS: EXAMPLE
#include <stdio.h>
int main()
{
char *nameptr[4] = {"Peter", "John", "Vincent", "Kenny"};
char name[4][8] = {"Mary", "Jane", "Susan", "May"};
int i, j;
printf("Ragged Array: \n");
for (i=0; i<4; i++)
printf("nameptr[%d] = %s\n", i,
nameptr[i]);
printf("Rectangular Array: \n");
for (j=0; j<4; j++)
printf("name[%d] = %s\n", j,
name[j]);
return 0;
}
Output
Ragged Array:
nameptr[0] = Peter
nameptr[1] = John
nameptr[2] = Vincent
nameptr[3] = Kenny
namePtr
P e t e r \0
J o h n \0
V i n c e n t \0
K e n n y \0
i = 3
Content Copyright Nanyang Technological University
27

When i=3



Note that if we want to print a character from the string, we can use the indirection operator (i.e. '**asterisk**'). For example, the statement

print f (percentage c, asterisk (name Pointer [0] + 2));

prints the character '**t**' from the string "**Peter**".

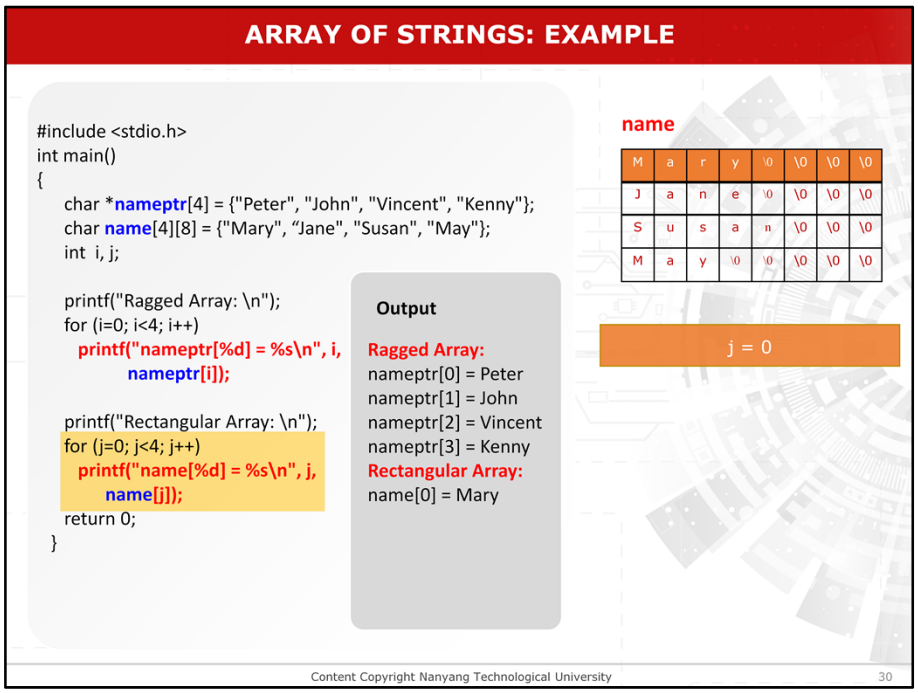
ARRAY OF STRINGS: EXAMPLE

```
#include <stdio.h>
int main()
{
    char *nameptr[4] = {"Peter", "John", "Vincent", "Kenny"};
    char name[4][8] = {"Mary", "Jane", "Susan", "May"};
    int  i, j;

    printf("Ragged Array: \n");
    for (i=0; i<4; i++)
      printf("nameptr[%d] = %s\n", i,
             nameptr[i]);

    printf("Rectangular Array: \n");
    for (j=0; j<4; j++)
      printf("name[%d] = %s\n", j,
          name[j]);
    return 0;
  }
```

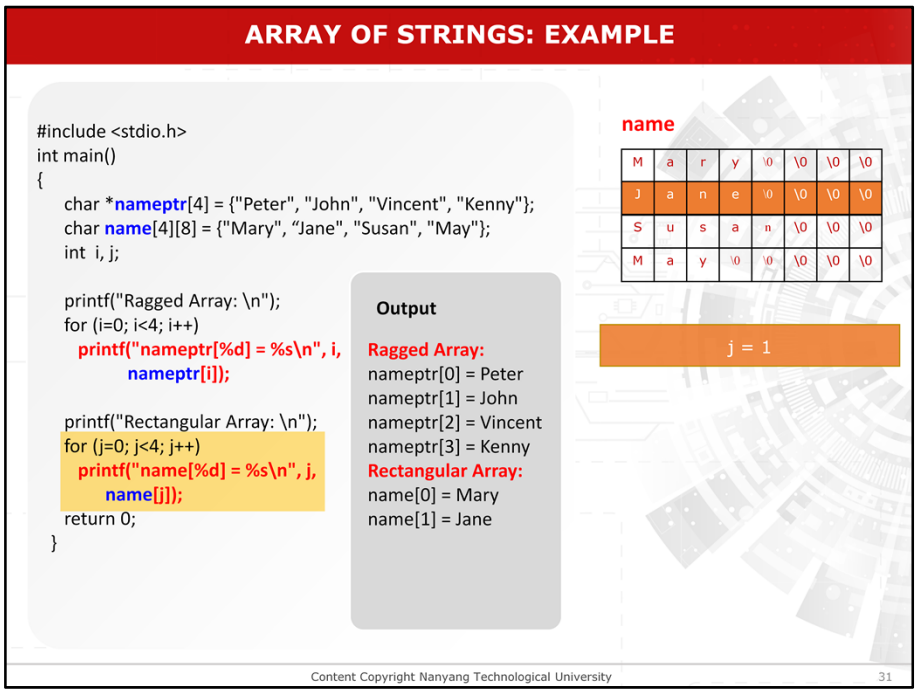
Output

```
Ragged Array:
nameptr[0] = Peter
nameptr[1] = John
nameptr[2] = Vincent
nameptr[3] = Kenny
Rectangular Array:
```

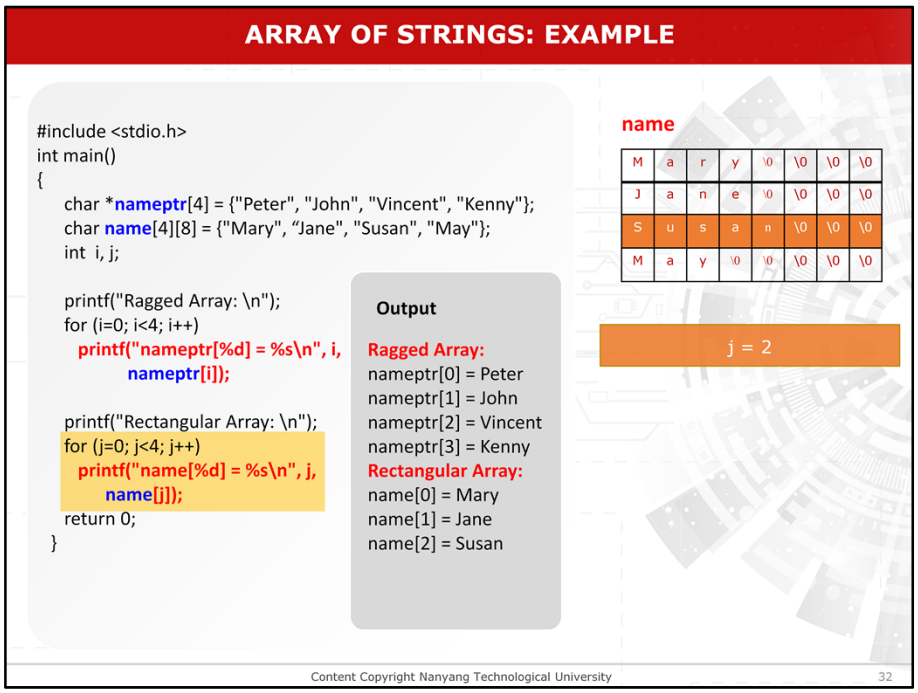
Let's take a look at the rectangular array.


ARRAY OF STRINGS: EXAMPLE
#include <stdio.h>
int main()
{
char *nameptr[4] = {"Peter", "John", "Vincent", "Kenny"};
char name[4][8] = {"Mary", "Jane", "Susan", "May"};
int i, j;
printf("Ragged Array: \n");
for (i=0; i<4; i++)
printf("nameptr[%d] = %s\n", i,
nameptr[i]);
printf("Rectangular Array: \n");
for (j=0; j<4; j++)
printf("name[%d] = %s\n", j,
name[j]);
return 0;
}
Output
Ragged Array:
nameptr[0] = Peter
nameptr[1] = John
nameptr[2] = Vincent
nameptr[3] = Kenny
Rectangular Array:
name[0] = Mary
name
M a r y \0 \0 \0 \0
J a n e \0 \0 \0 \0
S u s a n \0 \0 \0
M a y \0 \0 \0 \0 \0
j = 0
Content Copyright Nanyang Technological University
30

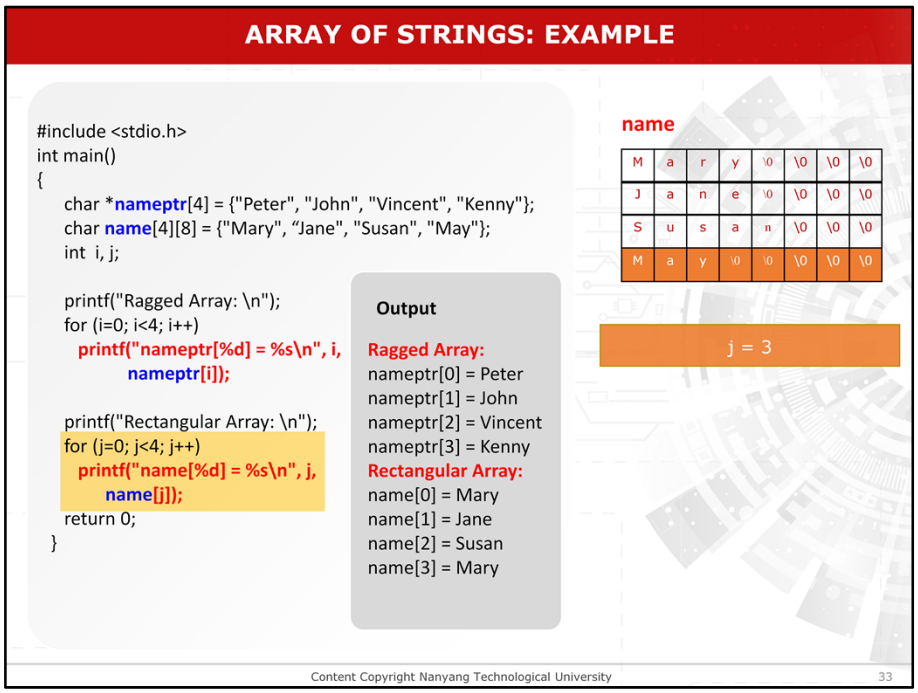
When j=0


ARRAY OF STRINGS: EXAMPLE
#include <stdio.h>
int main()
{
char *nameptr[4] = {"Peter", "John", "Vincent", "Kenny"};
char name[4][8] = {"Mary", "Jane", "Susan", "May"};
int i, j;
printf("Ragged Array: \n");
for (i=0; i<4; i++)
printf("nameptr[%d] = %s\n", i,
nameptr[i]);
printf("Rectangular Array: \n");
for (j=0; j<4; j++)
printf("name[%d] = %s\n", j,
name[j]);
return 0;
}
Output
Ragged Array:
nameptr[0] = Peter
nameptr[1] = John
nameptr[2] = Vincent
nameptr[3] = Kenny
Rectangular Array:
name[0] = Mary
name[1] = Jane
name
M a r y \0 \0 \0 \0
J a n e \0 \0 \0 \0
S u s a n \0 \0 \0
M a y \0 \0 \0 \0 \0
j = 1
Content Copyright Nanyang Technological University
31

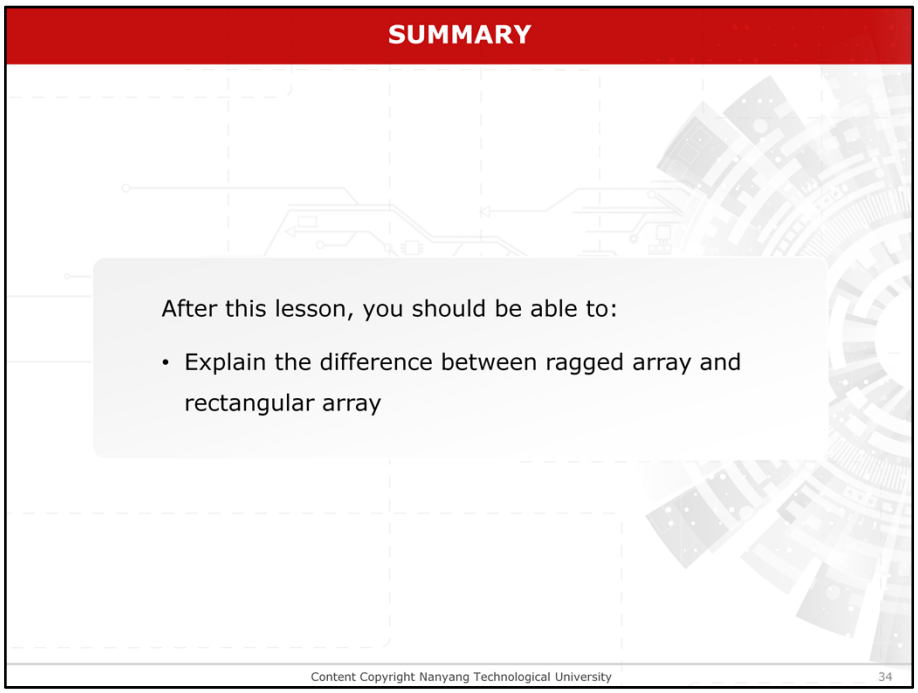
When j=1


ARRAY OF STRINGS: EXAMPLE
#include <stdio.h>
int main()
{
char *nameptr[4] = {"Peter", "John", "Vincent", "Kenny"};
char name[4][8] = {"Mary", "Jane", "Susan", "May"};
int i, j;
printf("Ragged Array: \n");
for (i=0; i<4; i++)
printf("nameptr[%d] = %s\n", i,
nameptr[i]);
printf("Rectangular Array: \n");
for (j=0; j<4; j++)
printf("name[%d] = %s\n", j,
name[j]);
return 0;
}
Output
Ragged Array:
nameptr[0] = Peter
nameptr[1] = John
nameptr[2] = Vincent
nameptr[3] = Kenny
Rectangular Array:
name[0] = Mary
name[1] = Jane
name[2] = Susan
name
M a r y \0 \0 \0 \0
J a n e \0 \0 \0 \0
S u s a n \0 \0 \0
M a y \0 \0 \0 \0 \0
j = 2
Content Copyright Nanyang Technological University
32

When j=2


ARRAY OF STRINGS: EXAMPLE
#include <stdio.h>
int main()
{
char *nameptr[4] = {"Peter", "John", "Vincent", "Kenny"};
char name[4][8] = {"Mary", "Jane", "Susan", "May"};
int i, j;
printf("Ragged Array: \n");
for (i=0; i<4; i++)
printf("nameptr[%d] = %s\n", i,
nameptr[i]);
printf("Rectangular Array: \n");
for (j=0; j<4; j++)
printf("name[%d] = %s\n", j,
name[j]);
return 0;
}
Output
Ragged Array:
nameptr[0] = Peter
nameptr[1] = John
nameptr[2] = Vincent
nameptr[3] = Kenny
Rectangular Array:
name[0] = Mary
name[1] = Jane
name[2] = Susan
name[3] = Mary
name
M a r y \0 \0 \0 \0
J a n e \0 \0 \0 \0
S u s a n \0 \0 \0
M a y \0 \0 \0 \0 \0
j = 3
Content Copyright Nanyang Technological University
33

When j=3



In summary, after viewing this video lesson, you should be able to do the points listed.