String

Department of Information System

SoICT, HUST

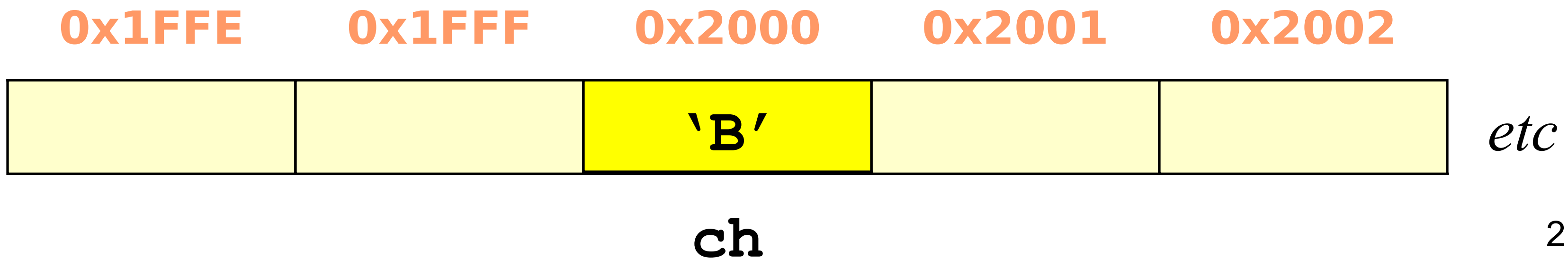
Store in the memory

- *Remind:*
  - Each cell in the memory is addressed.
  - Each variable declaration takes one cell to store value.

Example:

```
char  ch;
```

```
ch = 'B';
```

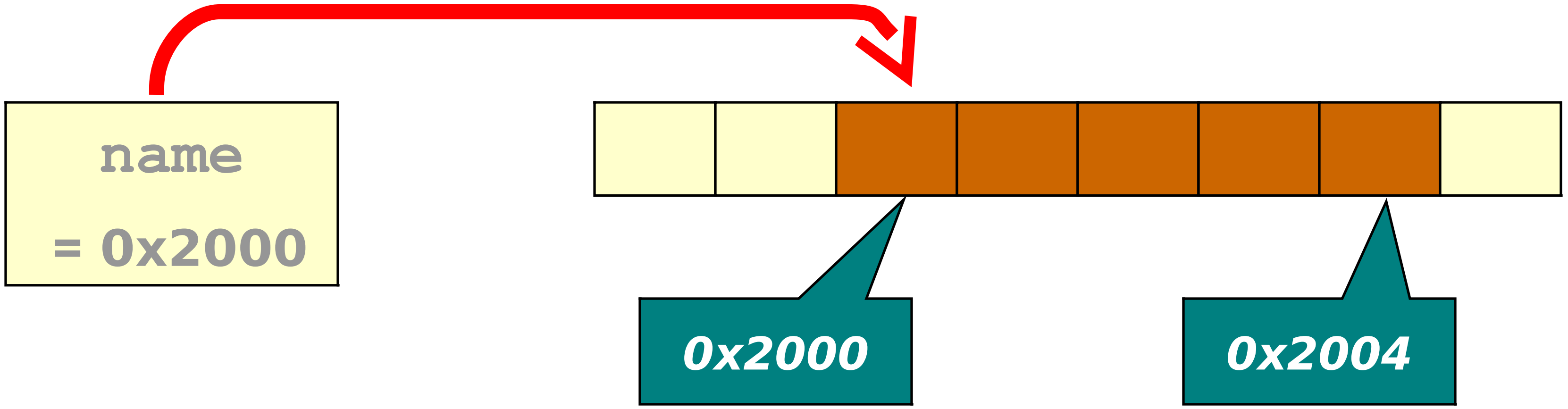


Representing a string

- A character string is a char array
- Each cell in the array contains a char
- The character string *must* have the terminating character (**’\0’**), aka null character. The address of the first character is the string’s address

Example:

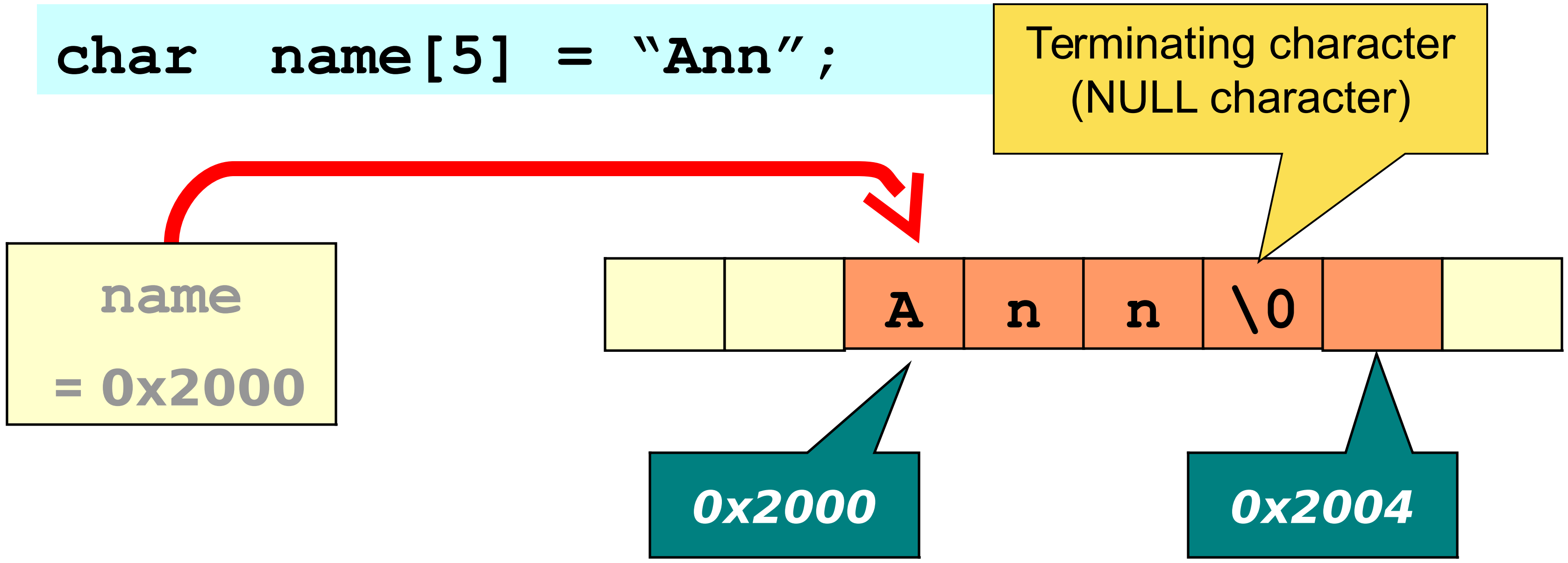
```
char  name[5];
```



String declaration

Declare 1:

```
char  name[5] = "Ann";
```



Equal declare:

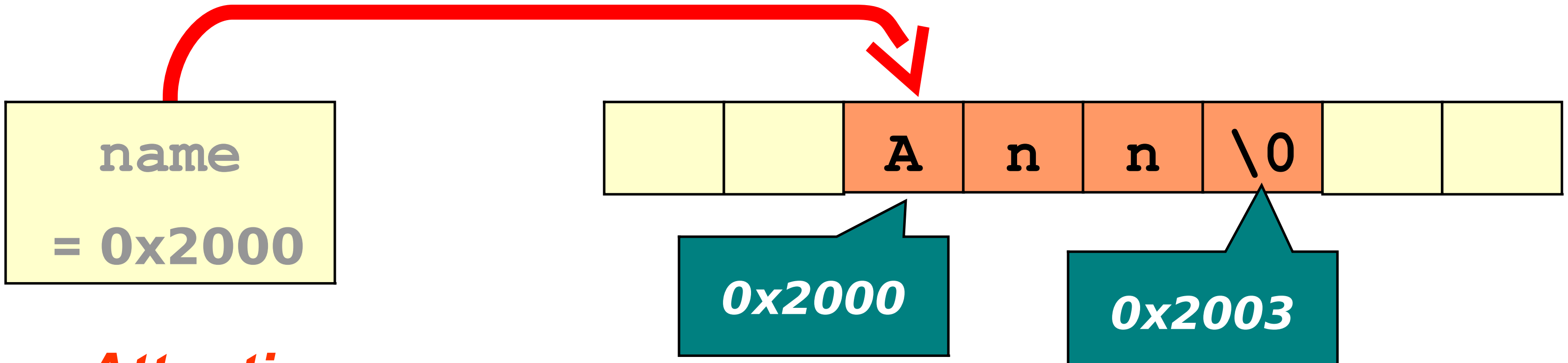
```
char  name[5] = {'A','n','n','\0'};
```

String declaration

Declare 2:

```
char  name[] = "Ann";
```

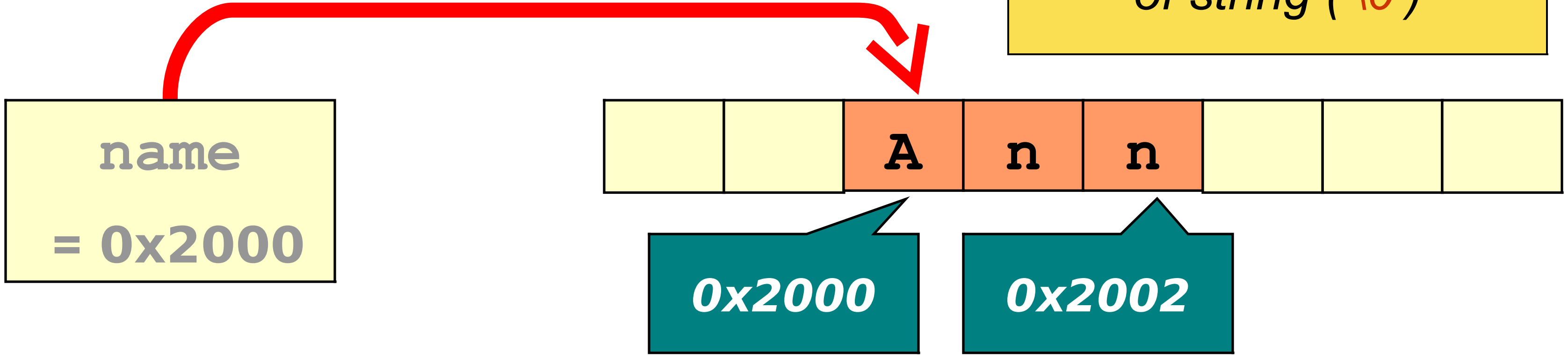
An additional character for end of string '\0'



Attention:

```
char  name[] = 'Ann';
```

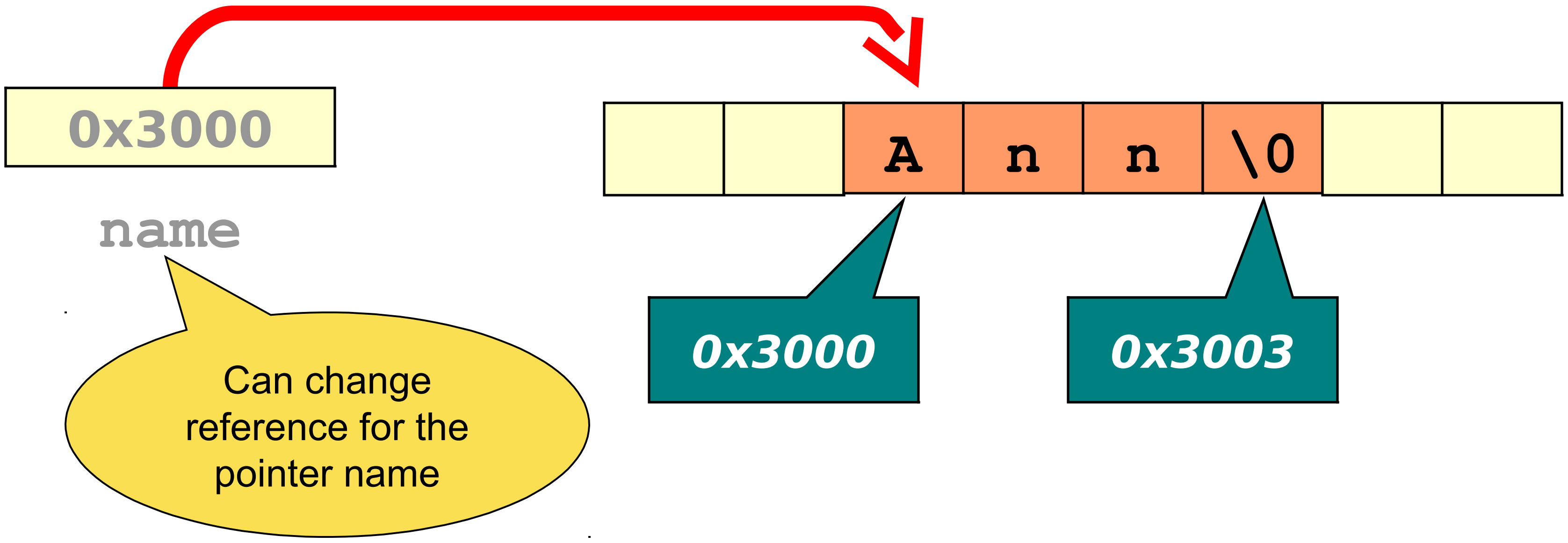
No character for end of string ('\0')



String declaration

Declaration 3: ⚠ ⚠

```
char  *name = "Ann";
```



Input/output a string

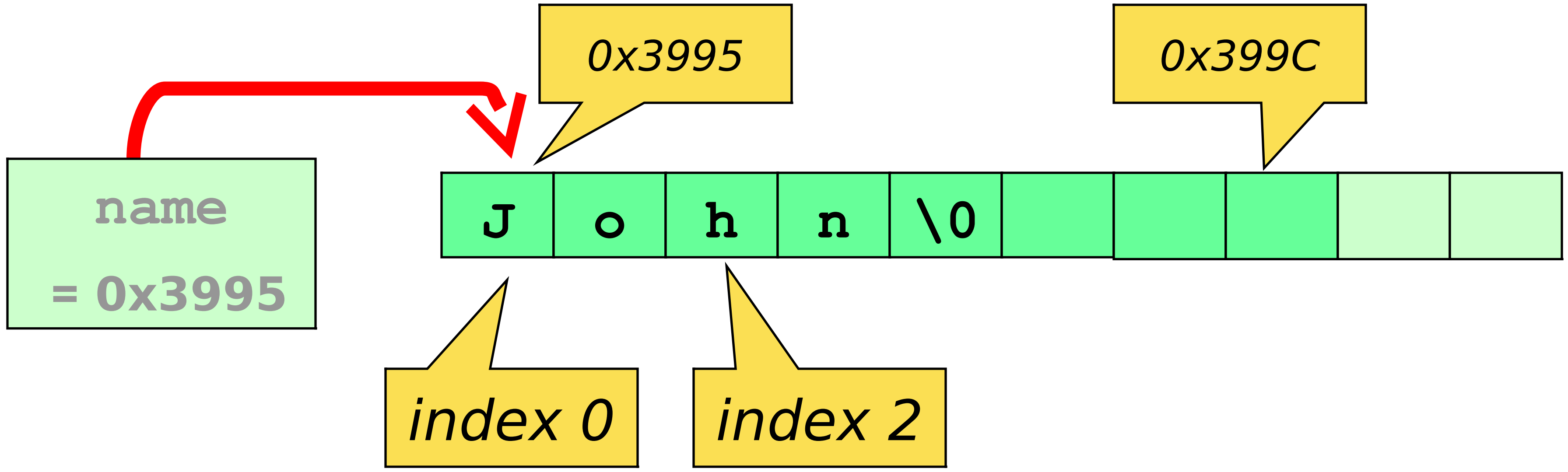
```
#include <stdio.h>

#define MAXLENGTH 15

int main()
{
   char str1[MAXLENGTH];
   char str2[MAXLENGTH];


   scanf("%s", str1);
   gets(str2);
   printf("%s\n%s\n", str1, str2);

   return 0;
}
```

Declare a constant

No operant & here

gets() allows input a string with spaces

Character in string



```
char name[8] = “John”;
int  i = 2;

printf(“Char at index %d is %c.\n”, i, name[i]);
```

```
output: Char at index 2 is h.
```

Program to calculating the number of character

- Calculate the characters that are not spaces in the input string

```
#include <stdio.h>

int main()
{
   char str[80];
   int dem, i;

   printf("Nhap xau bat ki: ");
   gets(str);

   dem = 0; i = 0;
   while ( str[i] != '\0' ) {
       if ( str[i] != ' ' ) dem++;
       i++;
   }
   printf("So ki tu khac trang trong xau la %d", dem);

   return 0;
}
```

String operations

- `#include <string.h>`
- Use library functions declared in <string.h>
  - Assignment : `strcpy()`

```
char s1[25];
char s2[25];
strcpy(s1, “Hello”);
strcpy(s2, s1);
```

String assignment

```
#include <stdio.h>
#include <string.h>

#define MAXLENGTH 100

int main()
{
   char string1[MAXLENGTH];
   char string2[MAXLENGTH];

   strcpy(string1, “Hello World!”);
   strcpy(string2, string1);

   return 0;
}
```

```
string1: “Hello World!”
string2: “Hello World!”
```

Common mistakes

```
s1 = “Hello”;
s2 = s1;
s1 = s1 + “Anna”;
s2 = s2 + “World”;

char s[4];
strcpy(s, “Hello”);
```

String operations

- Concatenation : `strcat()`

```
strcat(s1, " Anna");
strcat(s2, " World");
```

- Memory leak maybe occurs when copying/concatenating strings.
- Common errors:

```
char  name[5];
strcpy(name, "Ann");
strcat(name, " Smith");
```

String concatenation

```
char string1[80];
char string2[80];

strcpy(string1, “Goodbye”);
strcpy(string2, “, Cruel ”);

strcat(string1, string2);
strcat(string1, string2);
strcat(string1, “World!”);
```

```
string1: “Goodbye, Cruel , Cruel World!”
string2: “, Cruel ”
```

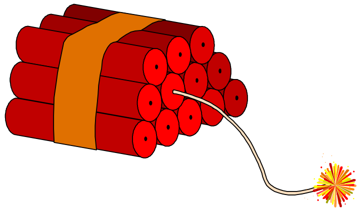
String operations

- Comparison : `strcmp()`
- strcmp returns 0 if str1 = str2

  <0 if str1 < str2

  >0 if str1 > str2

```
char str1[] = “Windows”;
char str2[] = “Unix”;
if (strcmp(str1, str2) < 0)
  printf(“%s %s\n”, str1, str2);
else  printf(“%s %s\n”, str2, str1);
```



Common errors

```
strcpy(string1, "Apple");
strcpy(string2, "Wax");

if (string1 < string2)
{
  printf("%s %s\n", string1, string2);
}
else
{
  printf("%s %s\n", string2, string1);
}
```

Compare addresses of the two strings

Strings as function parameters

- Declare as char* or char[]

```
void greeting (char* name)
void greeting (char name[])
```

- It points to the first character of the string
- Changes to the string inside the function affect the actual string
- It is not necessary to pass the length of the string to the function

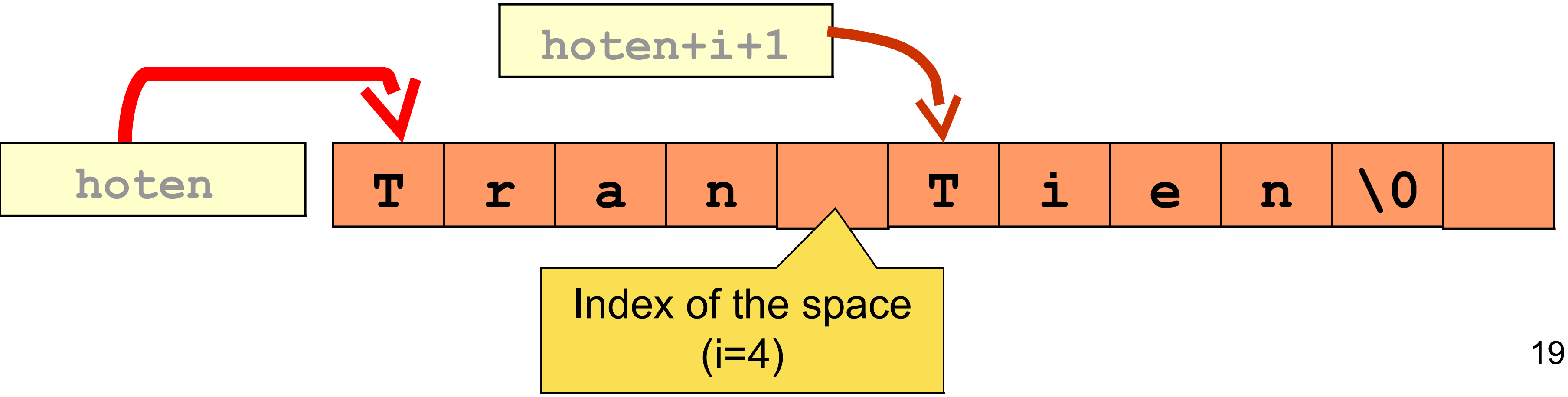
Example

```
char *capitalize(char * str)
{
    for (i=0; i<strlen(str); i++)
      if ( str[i]>='a'&& str[i]<='z')&&
            (i==0||str[i-1]==' ') )
          str[i] = 'A' + (str[i]-'a');
    return str;
}
```

Example

- Write a function that returns the name from a full name. The full name is not an empty string and does not have extra spaces

```
char * timten(const char[] hoten)
{
   int i;
   i = strlen(hoten)-1;
   /* Find the last space in the string */
   while (i >= 0 && hoten[i] != ' ') i--;
   return hoten + i + 1;
}
```



Example (con’t)

```
#include <stdio.h>
#include <string.h>

char * timten(const char[] hoten);

int main()
{
   char hoten[80];

   printf("Nhap mot xau ho va ten: ");
   gets(hoten);

   printf("Ten sau khi tach duoc: %s", timten(hoten));

   return 0;
}
```

Exercises

(i) Trim left blanks, right blanks, and redundant blanks in a string.

(ii) Inverse a string.

(iii) Copy first name or last name in a full name string.