



PROGRAMMING METHODOLOGY (PHƯƠNG PHÁP LẬP TRÌNH)

UNIT 16 (extra): Characters and Strings

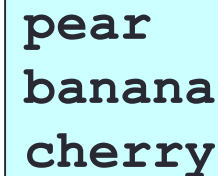
1. Array of Pointers to Strings (1/2)

- Declaration

```
char *fruits[3];
```

- Assignment

```
fruits[0] = "apple";  
fruits[1] = "banana";  
fruits[2] = "cherry";
```



```
pear  
banana  
cherry
```

- Declare and initialize

```
char *fruits[] = {"apple", "banana", "cherry"};  
fruits[0] = "pear";           // new assignment
```

- Output

```
for (i=0; i<3; i++)  
    printf("%s\n", fruits[i]);
```

1. Array of Pointers to Strings (2/2)

Unit16_ArrayOfPointersToStrings.c

```
#include <stdio.h>

int main(void) {
    char *fruits[] = { "apple", "banana", "cherry" };
    int i;

    fruits[0] = "pear";
    for (i=0; i<3; i++) {
        printf("%s\n", fruits[i]);
    }

    return 0;
}
```

2. Command-line Arguments (1/2)

- So far, our main function header looks like this:

```
int main(void)
```

- We can pass arguments to a program when we run it:

```
a.out water "ice cream" 34+7
```

- Add two parameters in the main function header:

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

- Parameter `argc` stands for “argument count”
- Parameter `argv` stands for “argument vector”. It is an array of pointers to strings.
- `argv[0]` is the name of the executable file (sometimes also called the command)
- You can name these two parameters anything, but the names `argc` and `argv` are widely used.

2. Command-line Arguments (2/2)

Unit16_CommandLineArgs.c

```
#include <stdio.h>
int main(int argc, char *argv[]) {
    int count;

    printf ("This program was called with \"%s\"\n", argv[0]);
    if (argc > 1)
        for (count = 1; count < argc; count++)
            printf("argv[%d] = %s\n", count, argv[count]);
    else
        printf("The command had no argument.\n");

    return 0;
}
```

```
gcc -Wall Unit16_CommandLineArgs.c
```

```
a.out water "ice cream" 34+7
```

This program was called with "a.out"

argv[1] = water

argv[2] = ice cream

argv[3] = 34+7