C Pointers

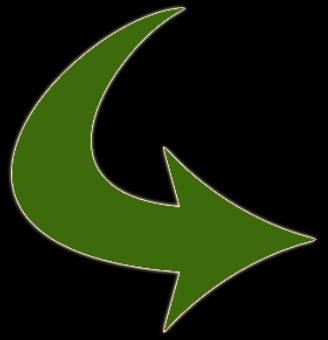
Pointers to Data, Pointer to Pointer, Passing Pointers, Function Pointers

SoftUni Team

Technical Trainers

Software University

http://softuni.bg



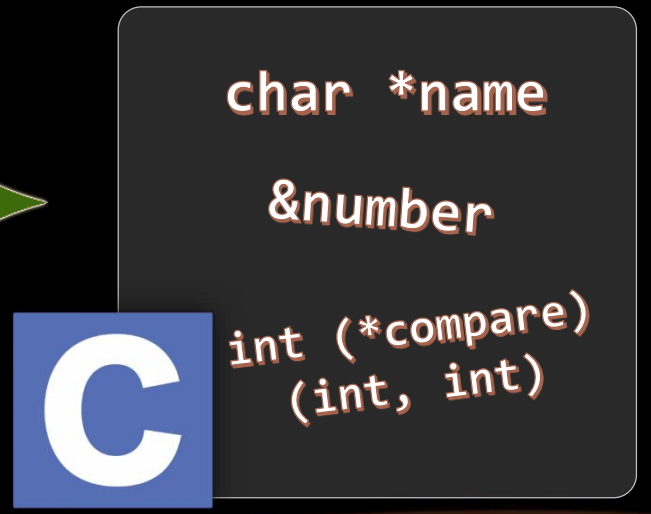


Table of Contents

char *name

0	1	2	3	4	5
'P'	'e'	's'	'h'	'o'	'\0'

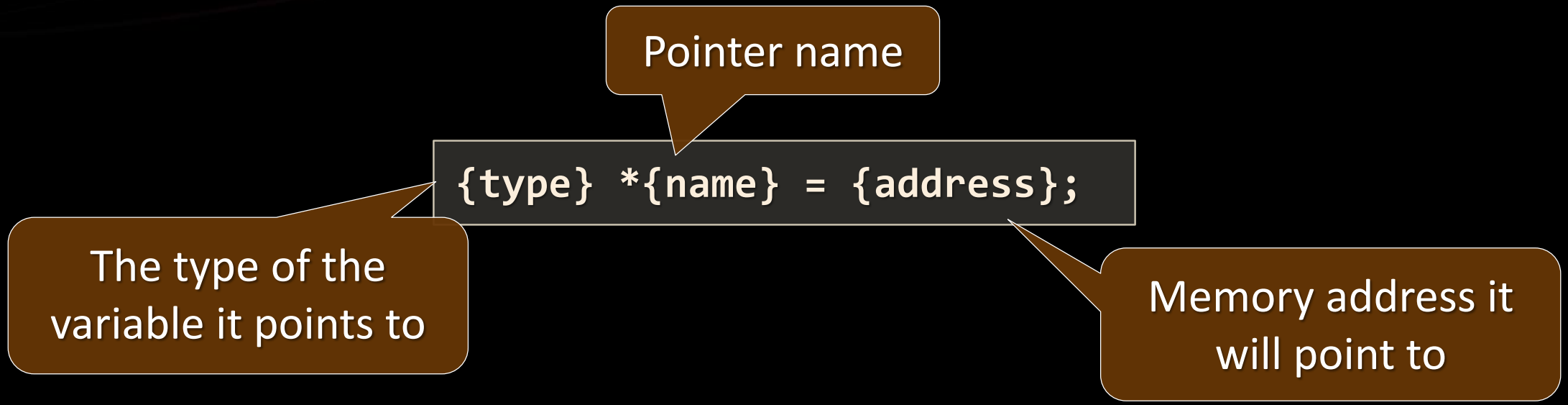
Pointers

Pointers

- Pointers are special types that point to an address in memory
  - Do not hold actual data (i.e. `'a'`, `3.14`, etc.)
  - Hold the address of another piece of memory

Declaring Pointers

- Pointers are declared with the * operator:



- Examples:

```
int *intPtr;
float *floatPtr;
char *name;
```

Initializing Pointers

- Pointers are initialized with the **&** operator
  - **&{identifier}** returns a pointer to the variable

```
int a = 5;
int *aPtr = &a; // aPtr now points to the memory of a

float f = 3.14;
float *fPtr = &f; // fPtr points to the memory of f
float *ptrPtr = fpr; // ptrPtr points to where fPtr points
```

  - By default uninitialized pointers point to junk memory

Dereferencing Pointers

- Pointers can be dereferenced
  - i.e. return the memory they point to
  - Done with the * operator

```
int a = 5;
int *aPtr = &a;
int value = *aPtr; // Dereference the pointer
printf("%d == %d", a, value);

*aPtr += 10; // Modify memory where it points to
printf("%d", a); // 15
```

Pointer Address

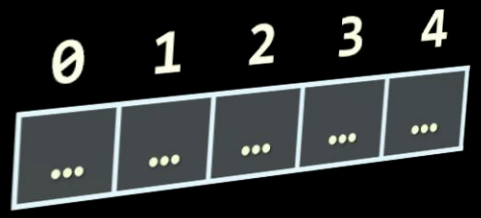
- A Pointer holds the address it points to
  - Address is **unsigned long** on 64-bit hardware
  - The pointer itself also takes up memory
    - 4 bytes on 32-bit and 8 bytes on 64-bit systems

```
char sign = '+';
char *signPtr = &sign;

printf("%lu\n", signPtr); // 140732033241279
printf("%d\n", sizeof(signPtr)); // 4 or 8
printf("%c\n", *signPtr); // +
```

Memory address varies every time

Pointers

Live Demo

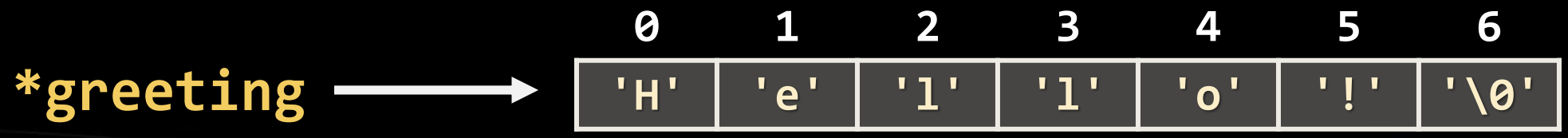


Arrays are Pointers?

Arrays are Pointers

- All arrays are pointers, in a sense
  - `char*`, `char[]`, `int[]`, `float[][]`, etc.
  - They do not hold the data
  - They point to a memory address where the elements reside
  - Example:

```
char greeting[] = "Hello!";
```



Dereferencing Arrays

- Dereferencing arrays will return the memory at the address
  - It will return bytes equal to the size of the pointed type
    - E.g. for `int` 4 bytes, `char` 1 byte, etc.

```
char greeting[] = "Hello!";

char letter1 = *greeting;          // same as greeting[0]
char letter2 = *(greeting + 1)); // same as greeting[1]
```

    - The pointer can be incremented to access consequent elements
    - Similar to using the `[]` operator

Pointer Arithmetic

- `pointer+1` does not always increment the pointer by 1
  - The actual result is `pointer+(1*sizeof(type))`

```
int array[] = { 5, 13, 2035 };
```

	5				13				2035			
bytes	0000 0000	0000 0000	0000 0000	0000 0101	0000 0000	0000 0000	0000 0000	0000 1101	0000 0000	0000 0000	0000 0111	1111 0011

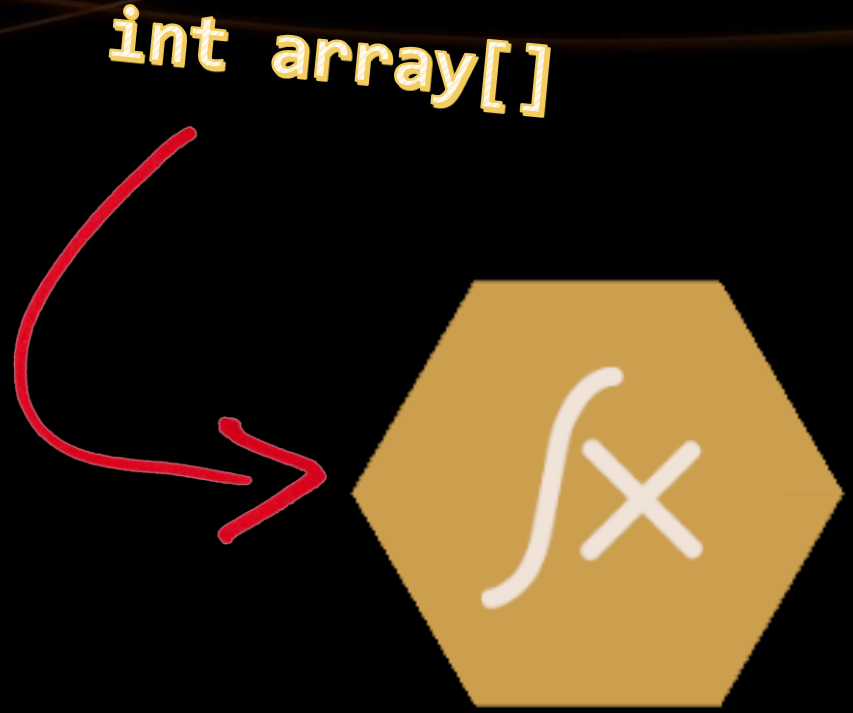
```
printf("Address of *arr: %d\n", arr);          // 1758749584
printf("Value of arr[0]: %d\n", *arr);         // 5
printf("Address of arr[1]: %d\n", arr + 1);    // 1758749588
printf("Value of arr[0]: %d\n", *(arr + 1));   // 13
```

Printing Array without Indexer – Example


```
#define TABLE_HEADER_FORMAT "%-10s|%15s\n"
#define TABLE_ROW_FOMAT "%-10d|%15lu\n"

int main()
{
    int arr[] = { 5, 3, 8, 7, 3, 2 };
    int length = sizeof(arr) / sizeof(int);

    printf(TABLE_HEADER_FORMAT, "Value", "Address");
    for (int i = 0; i < length; i++)
    {
        int *address = arr + i;
        int value = *(arr + i);
        printf(TABLE_ROW_FORMAT, value, address);
    }

    return 0;
}
```

Passing Pointers to Functions – Example


```
#include <ctype.h>
void convert_to_uppercase(char *stringPtr)
{
    while (*stringPtr != '\0')
    {
        *stringPtr = toupper(*stringPtr);
        *stringPtr++;
    }
}

int main()
{
    char text[] = "HeLLo 123!";
    convert_to_uppercase(text);
    printf("%s\n", text);

    return 0;
}
```



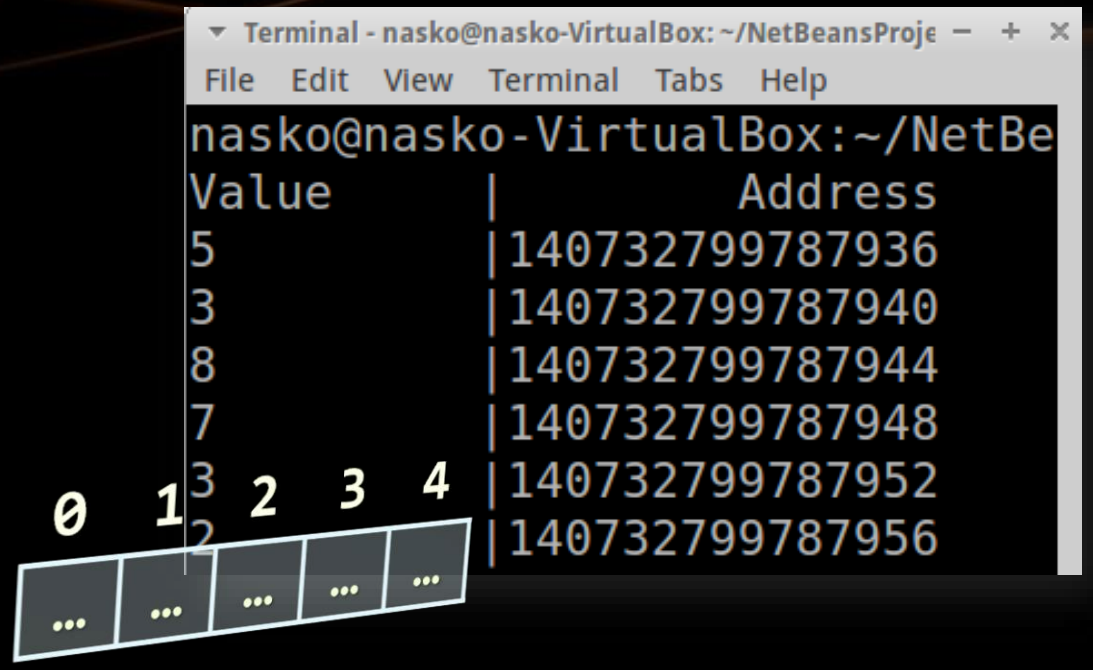
Passing Pointers to Functions

Live Demo

SOFTWARE UNIVERSITY FOUNDATION

Exercises in Class

Bubble Sort with Pointers



Printing Array Without Indexer

Live Demo

SOFTWARE UNIVERSITY FOUNDATION

void Pointers

void Pointers

- Void pointers point to a memory address holding data of unknown type
  - All pointer types can be assigned a void pointer
  - A void pointer can be assigned a pointer of any type

```
char *text = "These pointers are very strange";
void *voidPtr = text;
char *charPtr = voidPtr;
```

  - Note: Void pointers cannot be dereferenced

```
char letter = *voidPtr; // Error, cannot dereference void pointer
```

Generic Swap Function – Example


```
void swap(void *a, void *b, size_t size)
{
    char *aPtr = a;
    char *bPtr = b;
    for (int i = 0; i < size; i++)
    {
        char tempByte = aPtr[i];
        aPtr[i] = bPtr[i];
        bPtr[i] = tempByte;
    }
}
int main()
{
    char name[] = "Pesho";
    int array[] = { 1, 3, 10, 15, 2 };
    swap(&array[1], &array[3], sizeof(int));
    swap(&name[0], &name[4], sizeof(char));
}
```

Generic Swap Function

Live Demo

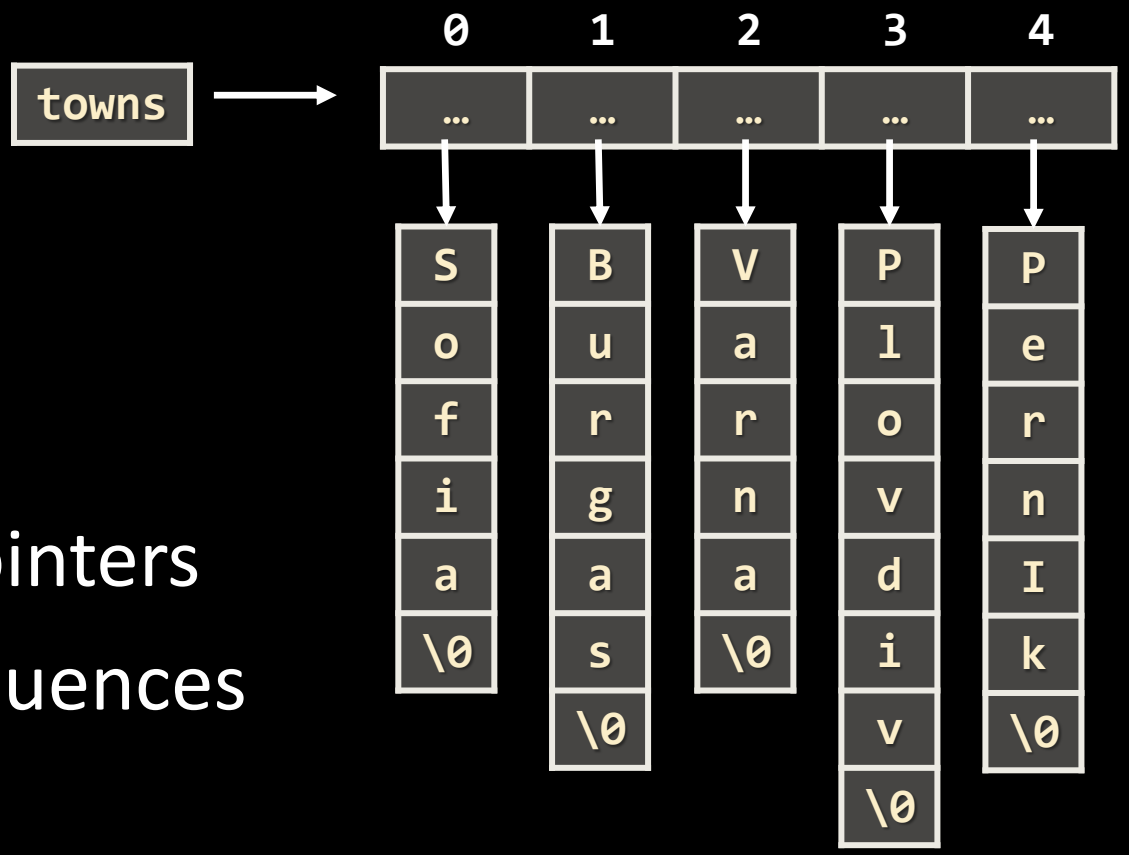
SOFTWARE UNIVERSITY FOUNDATION

Pointers to Pointers

Pointers to Pointers

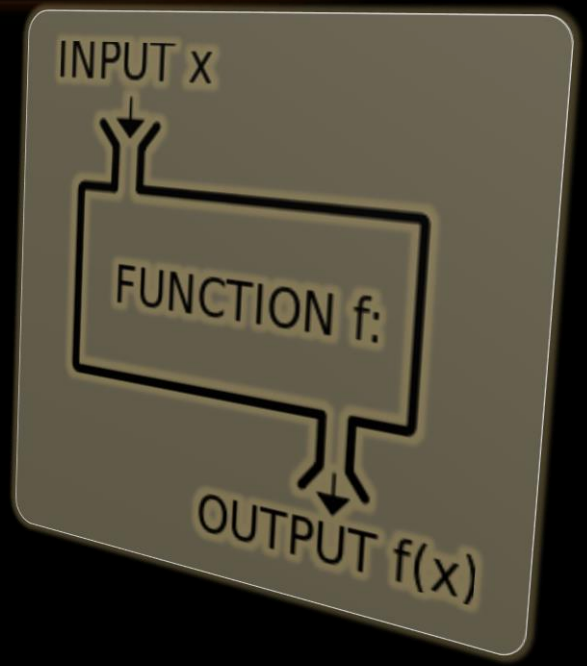
- Pointers can point to other pointers

```
char *towns[5];
towns[0] = "Sofia";
towns[1] = "Burgas";
towns[2] = "Varna";
towns[3] = "Plovdiv";
towns[4] = "Pernik";
```



- **towns** is a pointer to 5 **char** pointers
- Each **char** pointer points to sequences of characters

```
int sort(int (*order) (int, int))
{
   ...
}
```



Function Pointers

Passing Functions as Arguments

Function Pointers

- Function pointers point to a specific function
  - Declared in the following format:

```
{return-type} (*{name}) ({argType1}, {argType2}, ...)
```

  - Example:

```
int multiply(int a, int b) { return a * b; }

int main()
{
    int (*functionPtr)(int, int) = &multiply;
    return 0;
}
```

Dereferencing Function Pointers

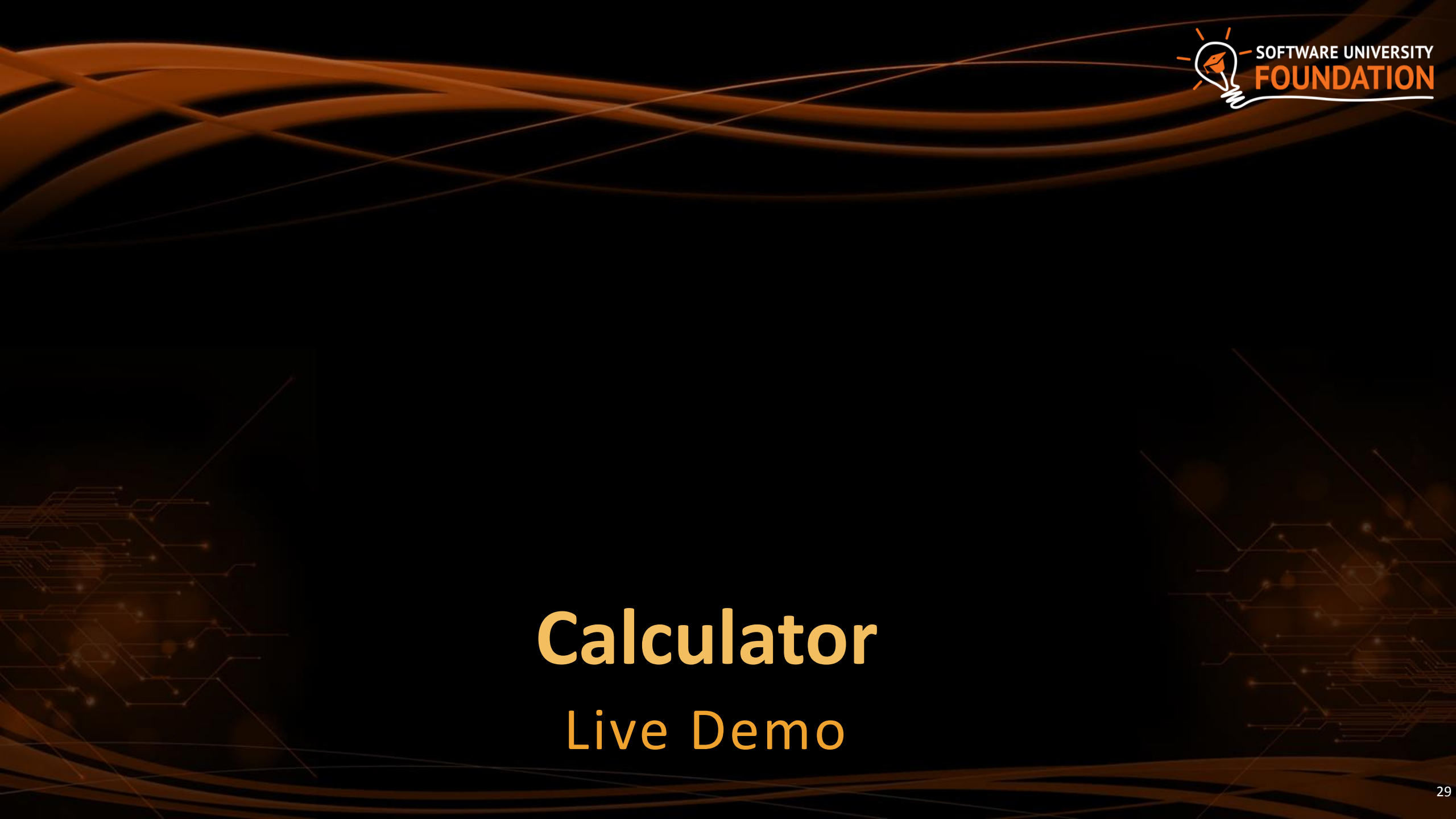
- The function pointer can be dereferenced
  - The pointed function can then be called

```
int (*functionPtr)(int, int) = &multiply;

int result1 = (*functionPtr)(5, 5); // 25
int result2 = functionPtr(10, 20);  // 200
```

    - Note: Explicit dereferencing can be omitted on some compilers

Calculator – Example

```
int multiply(int a, int b) { return a * b; }
int add(int a, int b) { return a + b; }
int subtract(int a, int b) { return a - b; }

void print_calculations(int a, int b, int (*calcFunc)(int, int))
{
    int result = calcFunc(a, b);
    printf("Result: %d\n", result);
}

int main()
{
    print_calculations(8, 7, &multiply);
    print_calculations(2500, 500, &add);
    print_calculations(3, 11, &subtract);
}
```



Calculator

Live Demo

Bubble Sort Comparator – Example


```
void bubble_sort(int array[], int length, int (*compare)(int, int))
{
    bool hasSwapped = true;
    while (hasSwapped)
    {
        hasSwapped = false;
        for (int i = 0; i < length - 1; i++)
        {
            if (compare(array[i], array[i + 1]))
            {
                swap(&array[i], &array[i + 1]);
                hasSwapped = true;
            }
        }
    }
}
```

Bubble Sort Comparator – Example (2)


```
int ascending(int a, int b)
{
    return a > b; // Evaluates to 1 if true
}

int descending(int a, int b)
{
    return a < b; // Evaluates to 1 if true
}

int main()
{
    int array[] = { 2, 10, 7, 3, 2, 1, 13 };
    int length = sizeof(array) / sizeof(int);
    bubble_sort(array, length, &ascending);   // 1 2 2 3 7 10 13
    bubble_sort(array, length, &descending);  // 13 10 7 3 2 2 1
}
```

Bubble Sort Comparator

Live Demo

SOFTWARE UNIVERSITY FOUNDATION

Filter Function

Live Demo

C Programming – Pointers

SOFTWARE UNIVERSITY FOUNDATION

Questions?

https://softuni.bg/trainings/1212/C-Programming-October-2015

License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

  - Attribution: this work may contain portions from
    - "Programming Basics" course by Software University under CC-BY-SA license

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University @ YouTube
  - youtube.com/SoftwareUniversity
- Software University Forums – forum.softuni.bg