Oregon State University

School of Electrical Engineering and Computer Science

CS 261 – Recitation 2

Spring 2016

Outline

- Programming in C
  - Headers
  - Structures
  - Preprocessor
  - Pointers
- Programming Assignment 1

Programming in C

- Headers

  - To include a standard library in C, use "<>".

  E.g: #include <stdio.h>

  - To include a header file, use quotation marks "".

  E.g: #include "sort.h"

  - In fact, when using angle brackets, the preprocessor only searches for it in certain directories.
  - When using quotation marks, the preprocessor first looks for the file in the current working directory.

Programming in C

- **Special operators:** \`++\` and \`--\` operators
- \`++\` : x is incremented (x=x+1)
  - **++x** : increments x *before* it is evaluated
  - **x++** : increments x*after* it is evaluated
- \`--\` : x is decremented (x=x-1)
  - **--x** : decrements x *before* it is evaluated
  - **x--** : decrements x *after* it is evaluated

Usually best to use **x++** or **x--**

Programming in C

- **The `struct` type:**
  - Like class in Java with no method.
- **Declare a struct data type:**

```
struct  student        /*student is the name of this struct*/
{
  char   name[40];
  int   id;
  double gpa;
};
```

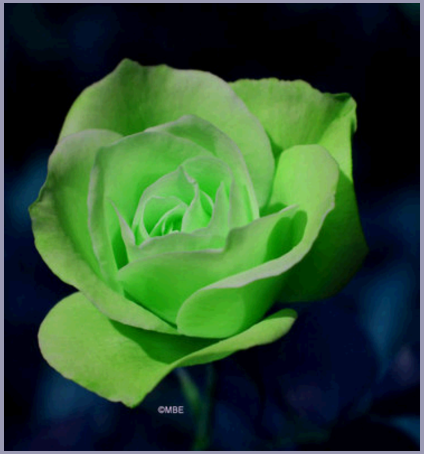
- **Declare variables with the structure type:**

```
struct student  a1, a2;
struct student *pointer_to_a1;
struct student entireClass[100];
```

Programming in C

- **Initialize structure variables:**

```
struct article {char name[15]; char color[14]; double price;};
struct article flower=
            {"rose", "green", 2.49};
struct article bouquet[10] ;
bouquet [0] = flower;
struct article *pArticle = (struct article *) malloc (sizeof(struct article));
```



Programming in C

- **Access structure members:**
  - **Using the dot operator**

```
flower.name     // The array 'name'
flower.price    // The double variable 'price'
```

  - **Using pointers**

```
pArticle = &flower;     // Let pArticle point to flower
pArticle->color         // Access members of flower
pArticle->price         // using the pointer pArticle
```

Programming in C

- **C Preprocessor**
  - The C compiler preprocesses every source file before performing the actual translation. The preprocessor removes comments and replaces macros with their definitions.
  - Every preprocessing directive appears on a line by itself. If the directive is long, it can be continued on the next line by inserting a backslash (\) as the last character before the line break.

- **`#define` directive:**
  - Used to define macros
  - Syntax:

```
#define name  [replacement_text]
```

  - Example:

```
#define  BUF_SIZE  512        // Symbolic constant
#define MAX(a,b)  ((a) > (b) ? (a) : (b))
```

Programming in C

- **#ifdef and #ifndef**
  - The #ifdef and #ifndef directives are used to test if a certain directive has been defined or not defined respectively.
  - Syntax

```
#ifdef _WIN32  //Compiling under a windows environment
 ...
#endif

#ifndef _WIN32
 ...
#endif
```

Whats the difference between "if" and "ifdef" ?

Pointers

- A pointer represents the *address* and *type* of a variable or a function. In other words, for a variable x, &x is a pointer to x.

- Two fundamental operators:
  - &: address-of operator – to get a pointer to (address of) a variable
  - *: dereference operator - get the thing the pointer points to.

- * is also used to declare a pointer variable

```
int i=5,
int *p;
```

- **Note**:
  - The name of an array is automatically converted to a pointer to the array's first element.
  - The value of a null pointer is 0.

Why pointers

- A simple explanation found on Web
- To impress friends *wink*
- Pointers can give performance gains
- New data structure possibilities

Pointer arithmetic

- Two arithmetic operations can be performed on pointers:
  - An integer can be added to or subtracted from a pointer.
  - One pointer can be subtracted from another of the same type.

- In arithmetic operations on pointers, the size of the objects pointed to is automatically taken into account.

```
int a[3] = { 0, 10, 20 };
int *ptr_a = a;

&a[i] , a+i , ptr_a+i                  // pointers to the i-th array element

a[i] , *(a+i) , *(ptr_a+i) , ptr_a[i]  // the i-th array element

ptr_a = a+2;
int n = ptr_a - a;
```

printf("%d", &a[2]); //Ans : memory location pointed by 20

printf("%d", *(a+1)); //Ans : 10

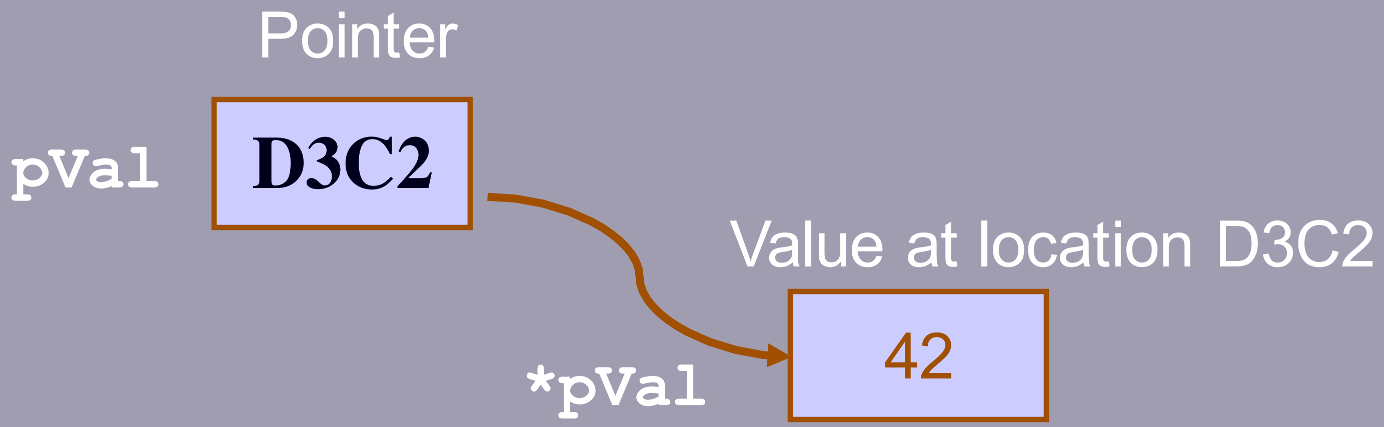
More on pointers!

Pointer Value vs. Thing Pointed At

the value of the pointer

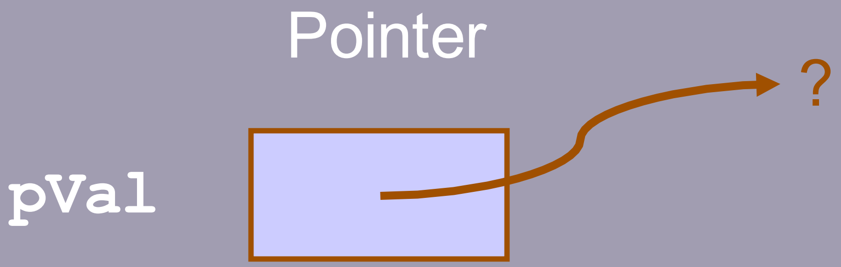
vs.

the value of the thing the pointer points to:



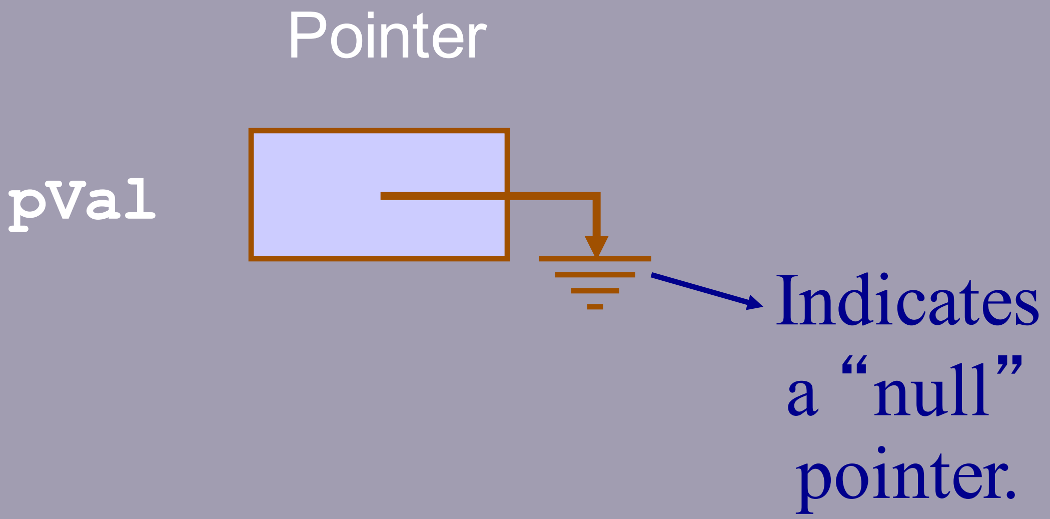
Pointers

```
int *pVal;    /* Pointer uninitialized to
                 unallocated integer value. */
```



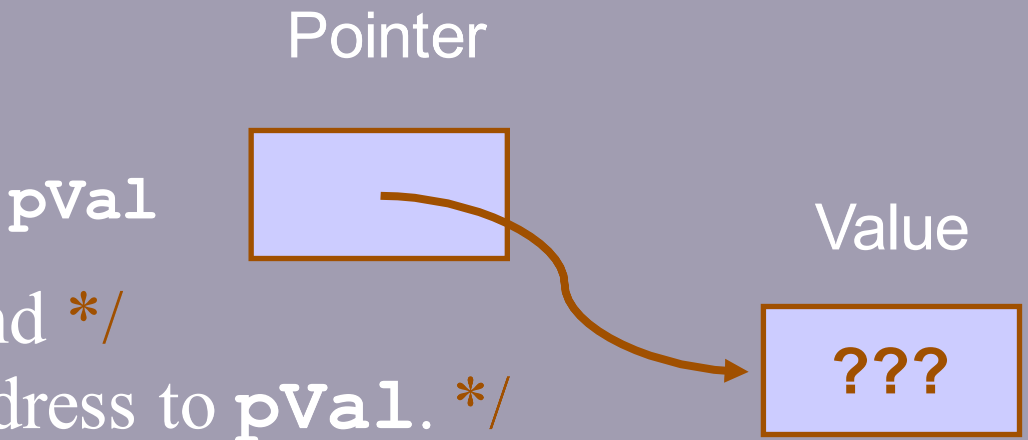
Pointers

```
int *pVal;     /* Pointer uninitialized to
                  unallocated integer value. */

pVal = 0;      /* Initialize pointer to indicate that
                  it is not allocated.  */
```



Pointers

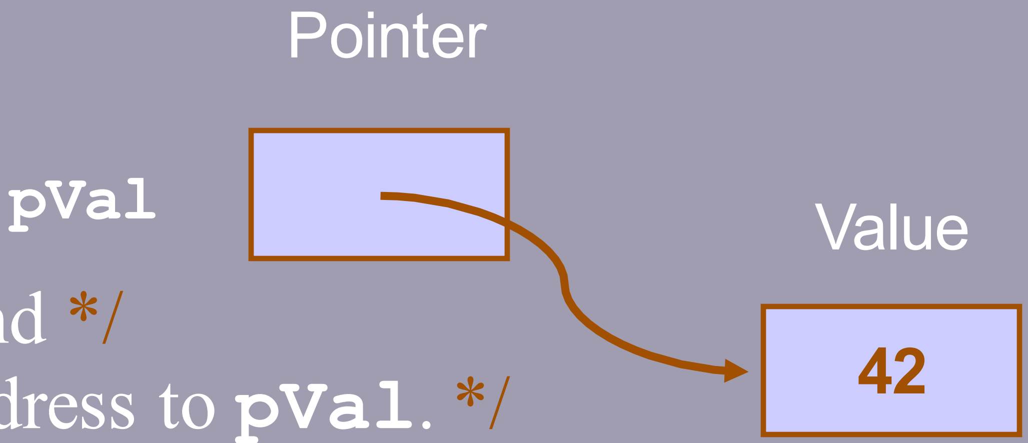
```
int *pVal;      /* Pointer uninitialized to
                   unallocated integer value. */

pVal = 0;       /* Initialize pointer to indicate that
                   it is not allocated.   */
.
.
.
```



```
/* Allocate integer and */
/* assign memory address to pVal. */

pVal = (int *) malloc(sizeof(int));
```

Pointers

```
int *pVal;      /* Pointer uninitialized to
                   unallocated integer value. */

pVal = 0;       /* Initialize pointer to indicate that
                   it is not allocated.   */
.
.
.
/* Allocate integer and */
/* assign memory address to pVal. */

pVal = (int *) malloc(sizeof(int));

*pVal = 42;
```



Pointer Syntax

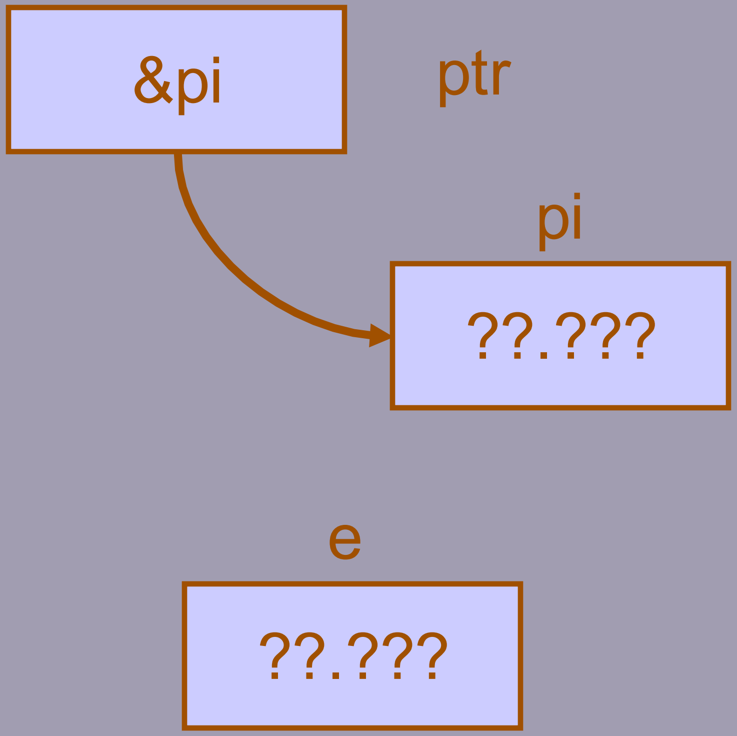
- Use `*` to
  - declare a pointer,
  - get value of pointer

- Use `&` to get address of a variable

```
double *ptr;
double pi, e;
```

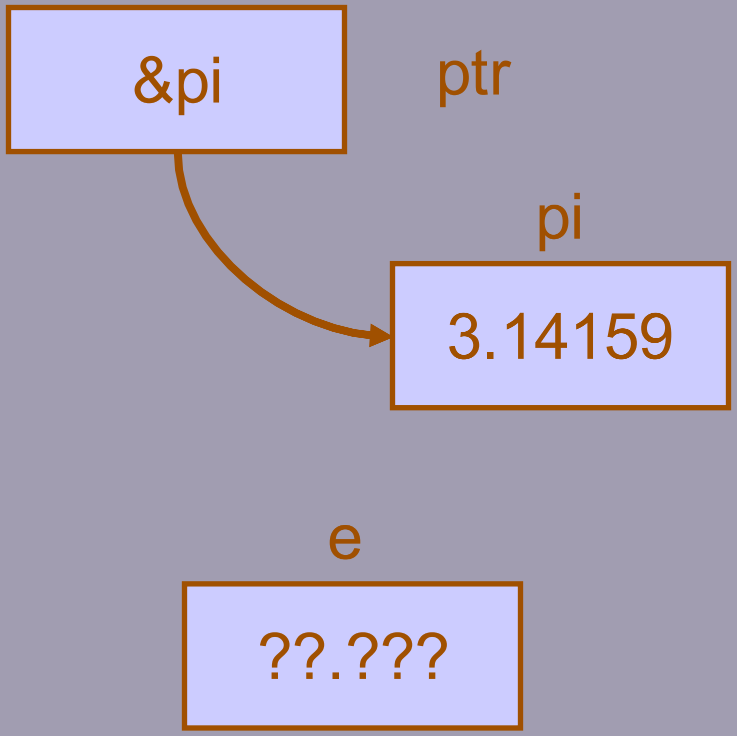
Pointer Syntax

```
double *ptr;
double pi, e;

ptr = &pi;
*ptr = 3.14159;
ptr = &e;
*ptr = 2.71828;
printf("Values: %p %g %g %g\n",
 ptr, *ptr, pi, e);
```



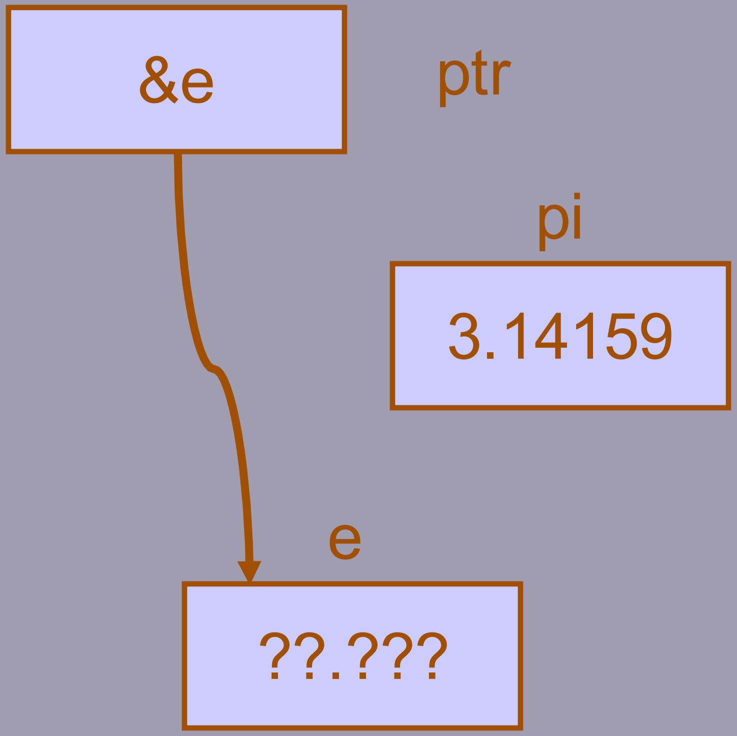
Pointer Syntax

```
double *ptr;
double pi, e;

ptr = &pi;
*ptr = 3.14159;
ptr = &e;
*ptr = 2.71828;
printf("Values: %p %g %g %g\n",
  ptr, *ptr, pi, e);
```



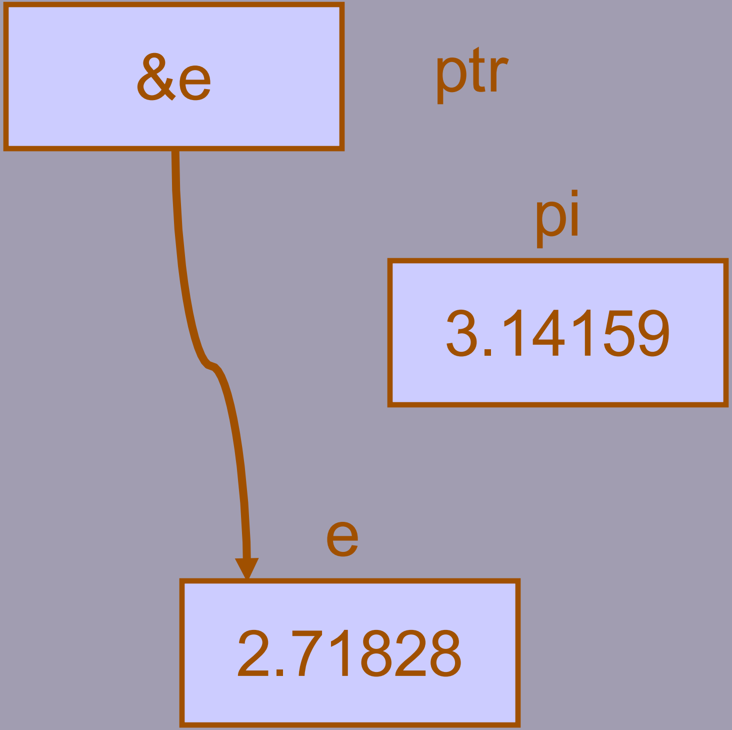
Pointer Syntax

```
double *ptr;
double pi, e;

ptr = &pi;
*ptr = 3.14159;
ptr = &e;
*ptr = 2.71828;
printf("Values: %p %g %g %g\n",
 ptr, *ptr, pi, e);
```



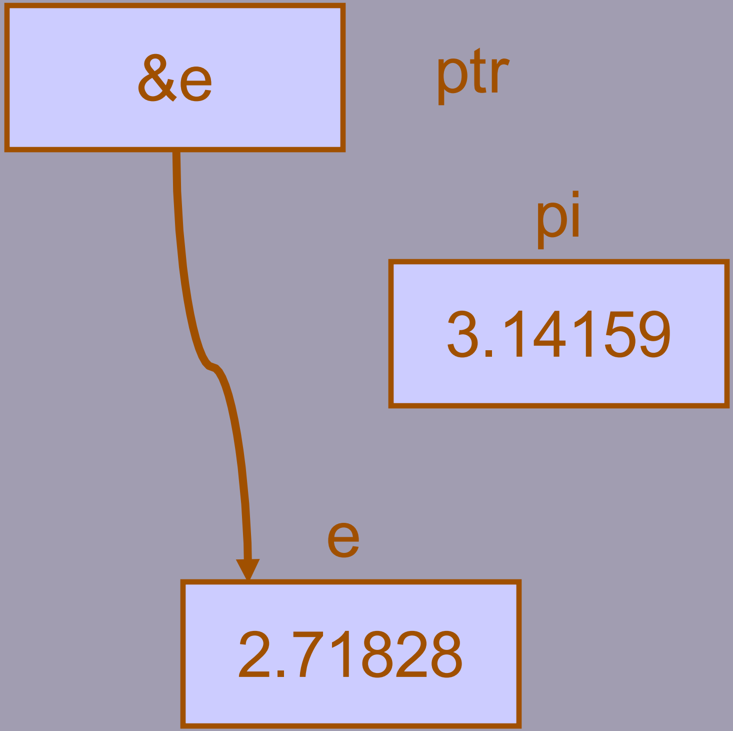
Pointer Syntax

```
double *ptr;
double pi, e;

ptr = &pi;
*ptr = 3.14159;
ptr = &e;
*ptr = 2.71828;
printf("Values: %p %g %g %g\n",
 ptr, *ptr, pi, e);
```



Pointer Syntax

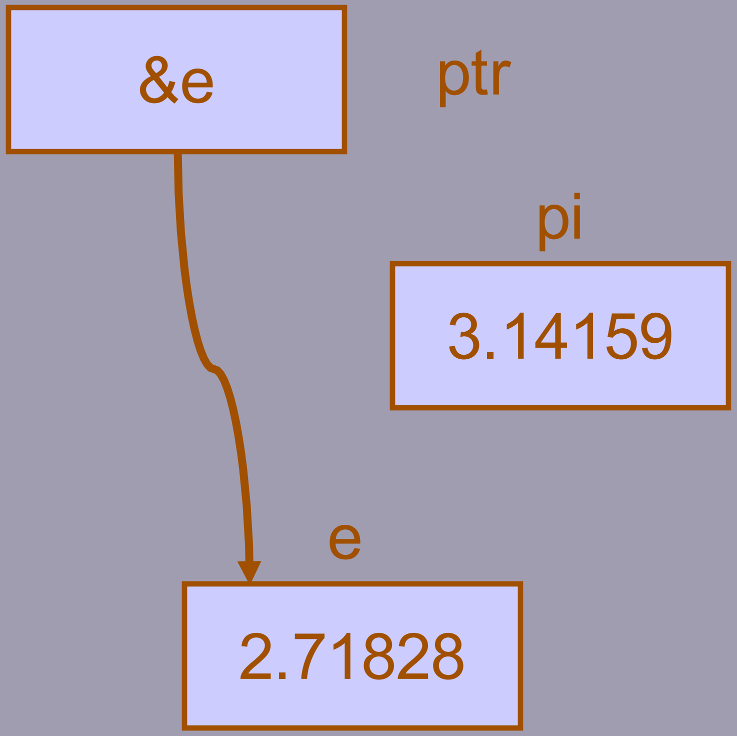
```
double *ptr;
double pi, e;

ptr = &pi;
*ptr = 3.14159;
ptr = &e;
*ptr = 2.71828;
printf("%p %g %g %g\n",
       ptr,  *ptr,  pi,   e);
```



Output: ?

Pointer Syntax

```
double *ptr;
double pi, e;

ptr = &pi;
*ptr = 3.14159;
ptr = &e;
*ptr = 2.71828;
printf("%p %g %g %g\n",
       ptr,  *ptr,  pi,  e);
```



Output: `ffbff958 2.71828 3.14159 2.71828`

Alternative Pointer Syntax

- Use `[]` to declare a pointer
- Use `[0]` to get the value of pointer

```
double data[]; /*pointer*/
double value = 5.3; /*variable*/

data = & value;
printf("%g",data[0]);
```

Output: ?

Programming Assignment 1 – Helpful Hints

- **warning: implicit declaration of function …**
  - Probable reason: Function prototype not declared
  - Fix: 2 choices
    - Insert the function prototype before the main function in C
    - Use a header file (myFunction.h) to declare the function prototype and include this header file in main.c

- **<filename>.h: No such file or directory**
  - Probable reason: wrong "include" definition

- **warning: implicit declaration of function `malloc' ('free') or 'assert'**
  - Probable reason: stdlib.h or assert.h library not included
  - Fix: To call `malloc` and `free` functions, you need to include stdlib.h library at the beginning of source files. To call 'assert' function, you need to include assert.h.

Programming Assignment 1 – Helpful Hints

- Even after a successful compilation, I'm not allowed to execute the program
  - **Example:**

```
% gcc main.c sort.c -o myProg
% myProg
```

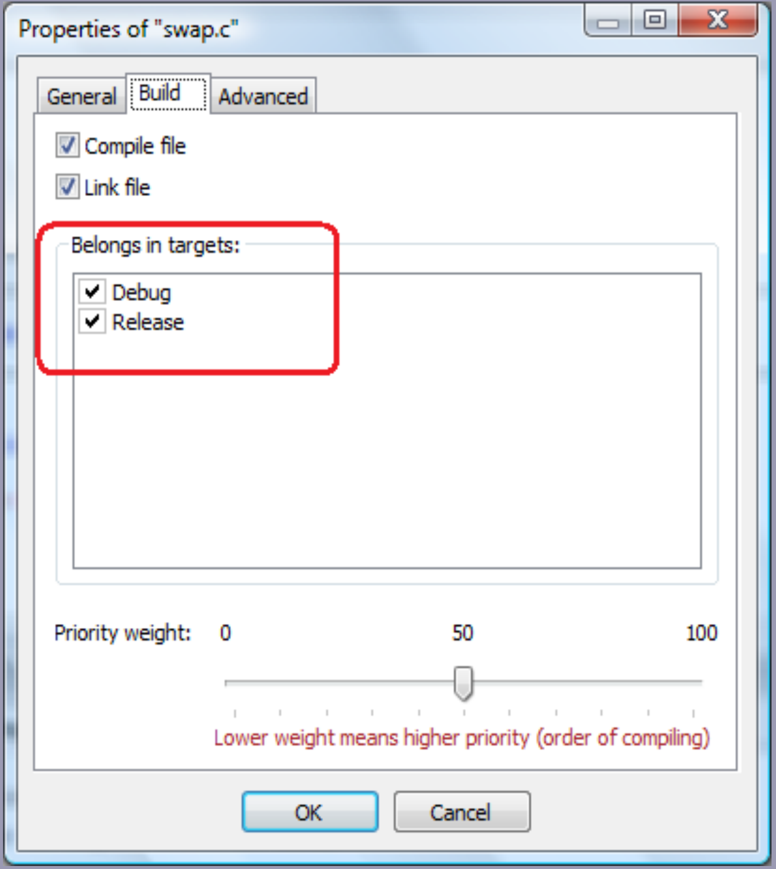
  - **Error message**:

```
myProg: Too few arguments.
```

  - **Reason**:
    - Path to the executable file not provided.
    - Executable file name is a UNIX keyword like cat, grep,
  - **Fix**: provide the path to the executable file:
    - `./myProg`
  - We can provide full path to the file, or just use `./` to indicate the current directory

Programming Assignment 1 – Helpful Hints

- **CodeBlocks errors:**
  - Message: Linking stage skipped (build target has no object files to link)
  - Fix: The source file must belong in `debug` and 'release` targets in order to be compiled.



Properties of "swap.c"

General | Build | Advanced

[✓] Compile file

[✓] Link file

Belongs in targets:

[✓] Debug

[✓] Release

Priority weight: 0 50 100

Lower weight means higher priority (order of compiling)

Test Assignment – Solution

- **Question:**
  - Code in C for printing prime numbers in an infinite loop.
  - Execution is stopped by the user
  - Comments based on guidelines

- **Helper function:**

```
int isPrime(int n) {

    for(int i = 2; i * i <= n; i++) {     /* for every possible number i */
        if (n % i == 0) return 0;         /* if i divides n then n is not a prime number */
    }
    return 1;                             /* if no number divides n from 2 to sqrt(n), n is prime */


}
```

Test Assignment – Solution

- **Question:**
  - Code in C for printing prime numbers in an infinite loop.
  - Execution is stopped by the user
  - Comments based on guidelines

- **Main function:**

```
int main() {
        int j = 2; /* first prime number */
        printf("Press enter for next prime number, give other character for termination\n");
        while(true) {
                if (isPrime(j)){                        /* if j is a prime number */
                        printf("%d is prime",j);  /* then print it */
                        char c = getchar();
                        if (c != '\n')                  /* if the user did not press enter alone */
                                return 0;               /* then break out of the loop by returning */
                }
                j++;                                    /* increase j to the next integer value */
        }
        return 0; /* purely cosmetic reasons, never executed */
}
```

More Questions ?