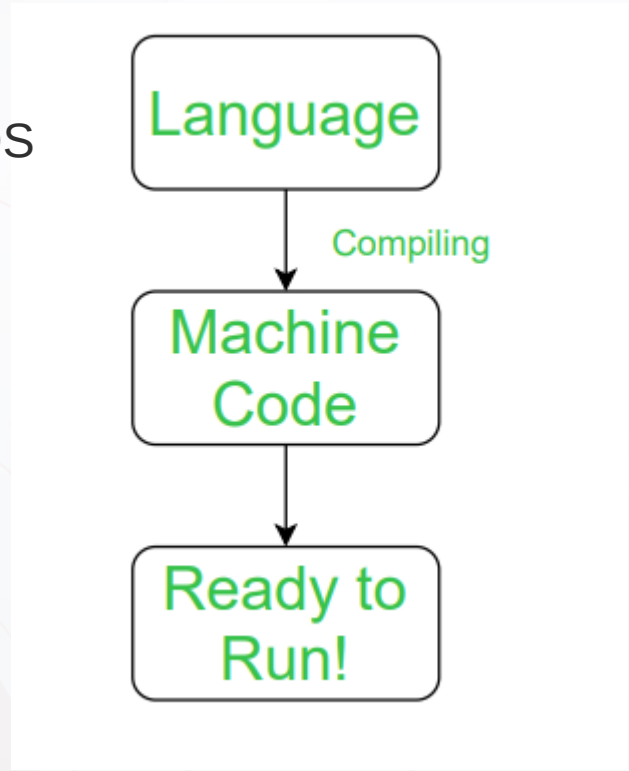
Compiled Language

- Faster
- Closer to OS

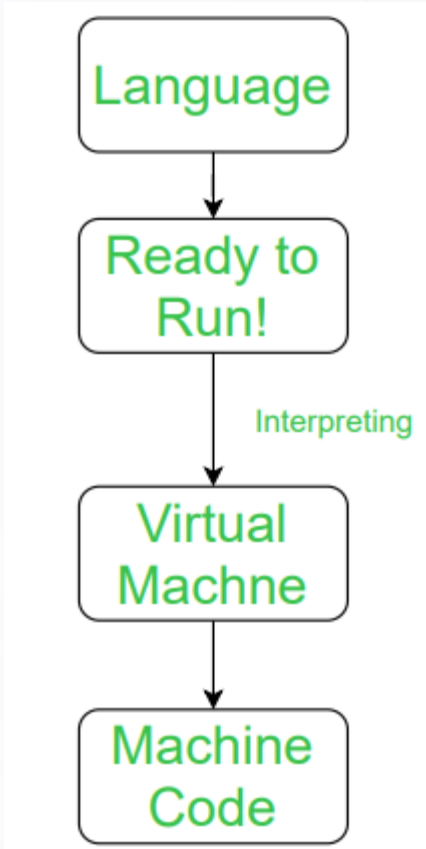
- Platform dependent
- Steeper learning curves
- Difficult to debug / modify



e.g. Fortran, C, C++, C#, COBOL, Rust…

VS.

Interpreted Language



- Cross-platform
- Run directly
- Usually easier
- Run-time debugging

- Slower
- Source code visible to user

e.g. Python, matlab, Java script, PHP, BASIC…

Strengths of C

- *Efficiency*
- *Portability*
- *Power*
- *Flexibility*
- *Standard library*
- *Integration with UNIX*

Weaknesses of C

- *Programs can be error-prone.*
- *Programs can be difficult to understand.*
- *Programs can be difficult to modify.*

When size or speed matters, go with C!

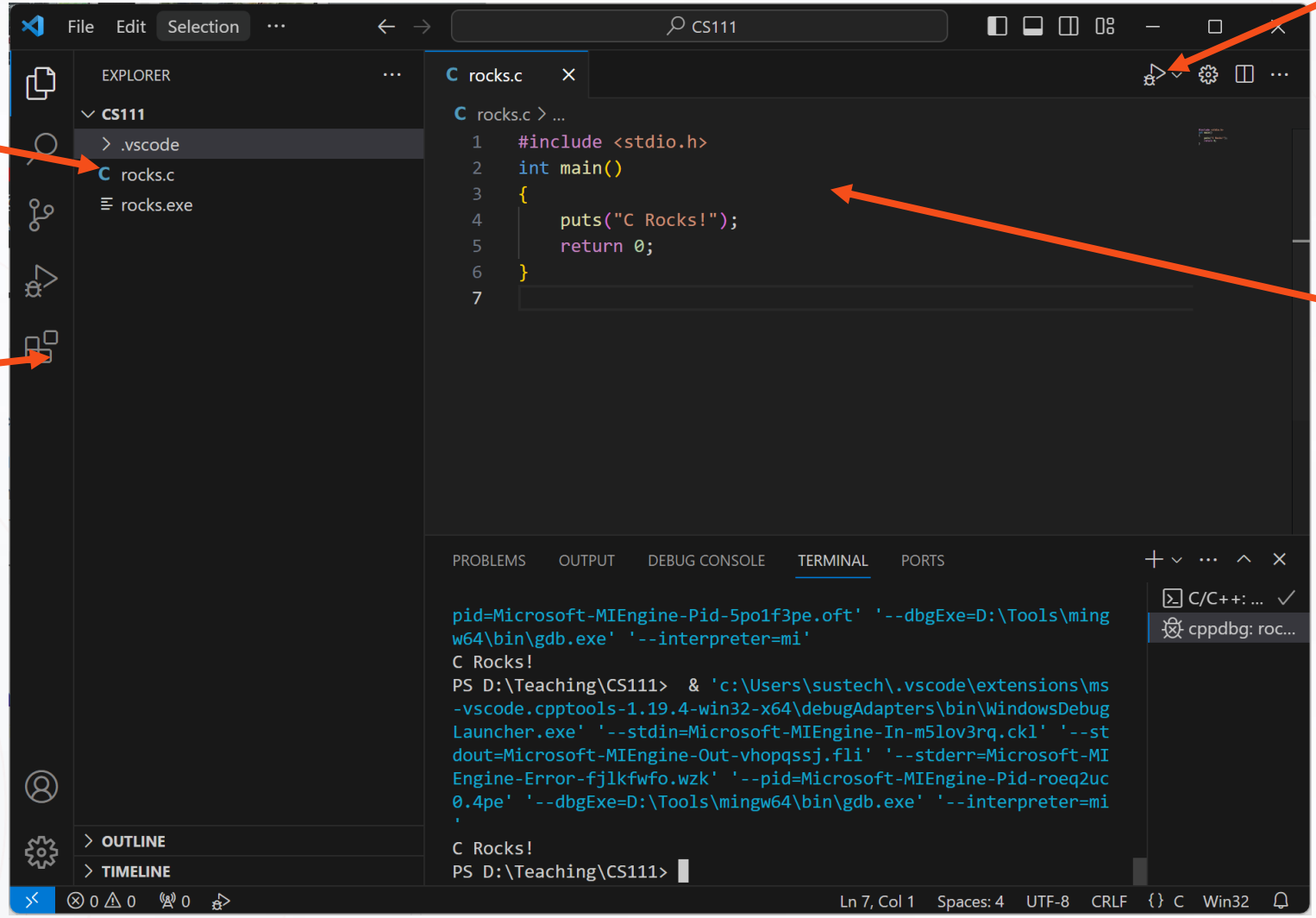
IDE
(Integrated Development Environment)

Easy compiling & debugging

Project management

Source file editor with (customizable) color theme

Open extensions for virtually all languages



MS Visual Studio Code

Lecture 2 Diving into C

Sample code fragments

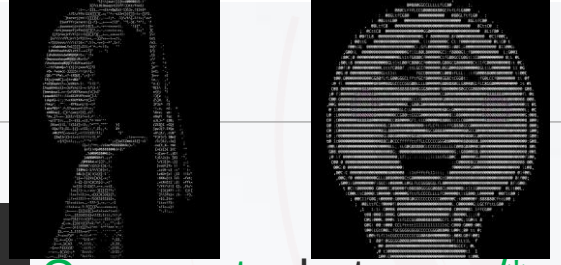
```
int card_count = 11;
if (card_count > 10)
puts("The deck is hot. Increase bet.");
int c = 10;
while (c > 0) {
        puts("I must not write code in class");
        c = c - 1;
}

/* Assume name shorter than 20 chars. */
char ex[20];
puts("Enter boyfriend's name: ");
scanf("%19s", ex);
printf("Dear %s.\n\n\tYou're history.\n", ex);
char suit = 'H';
```

```
switch(suit) {
case 'C':
        puts("Clubs");
        break;
case 'D':
        puts("Diamonds");
        break;
case 'H':
        puts("Hearts");
        break;
default:
        puts("Spades");
}
```

Guess what they mean!

A complete C program



These * are for decoration

```c
/*
* Program to calculate the number of cards in the shoe.
* This code is released under the Vegas Public License.
* (c)2014, The College Blackjack Team.
*/
#include <stdio.h>

int main()
{
    int decks;
    puts("Enter a number of decks");
    scanf("%i", &decks);
    if (decks < 1)
    {
        puts("That is not a valid number of decks");
        return 1;
    }
    printf("There are %i cards\n", (decks * 52));
    return 0;
}
```

Comments: between /* and */
Can cross multiple lines
Does not affect running

Header files for external libraries

C programs consist of **functions**
The main function is where a program starts

The function **puts** prints a string on the screen.

Formatted input & output functions.

Return values of the main function

A function can have a return type. Always int for main

The body of a function (and other blocks) is always surrounded by curly braces

A program to show card value

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char card_name[3];
    puts("Enter the card_name: ");
    scanf("%2s", card_name);
    int val = 0;
    if (card_name[0] == 'A') {
        val = 1;
    } else if (card_name[0] == 'J') {
        val = 11;
    } else if (card_name[0] == 'Q') {
        val = 12;
    } else if (card_name[0] == 'K') {
        val = 13;
    } else {
        val = atoi(card_name);
    }
    printf("The card value is: %i\n", val);
    return 0;
}
```

No native support to strings

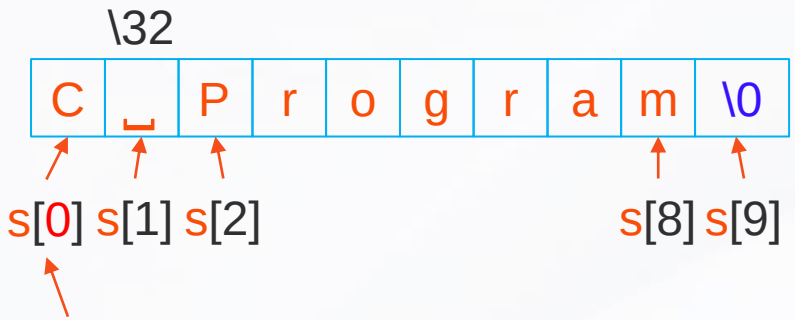
```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char card_name[3];
    puts("Enter the card_name: ");
    scanf("%2s", card_name);
    int val = 0;
    if (card_name[0] == 'A') {
        val = 1;
    } else if (card_name[0] == 'J') {
        val = 11;
    } else if (card_name[0] == 'Q') {
        val = 12;
    } else if (card_name[0] == 'K') {
        val = 13;
    } else {
        val = atoi(card_name);
    }
    printf("The card value is: %i\n", val);
    return 0;
}
```

A string is represented by an array of type **char**

Declaration of a character array of **3** elements.

All variables must be declared before usage!

Representation of a string s: “C Program”



*Array index starts from **0** in C*

The NULL character ‘\0’ (ASCII value 0) is used to indicated the termination of a string

We always need one extra element for a string!

Not all equal signs are equal

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char card_name[3];
    puts("Enter the card_name: ");
    scanf("%2s", card_name);
    int val = 0;
    if (card_name[0] == 'A') {
        val = 1;
    } else if (card_name[0] == 'J') {
        val = 11;
    } else if (card_name[0] == 'Q') {
        val = 12;
    } else if (card_name[0] == 'K') {
        val = 13;
    } else {
        val = atoi(card_name);
    }
    printf("The card value is: %i\n", val);
    return 0;
}
```

Assignment or initiation (*in declaration*)

```
val = 1;
```

Testing equality

```
val == 1
```

gives 1 if equal
(0 otherwise)

Other related assignment operators

```
val += 2;        /* val = val + 2; */
val -= 3;        /* val = val - 3; */

val++;           /* val = val + 1; */
val--;           /* val = val - 1; */
```

Control statement

```
if (value_of_hand <= 16)
    hit();
else
    stand();
```

The condition

Run this if the condition is met. (true)

Run this if the condition is not met. (false)

This part is optional

```
if (dealer_card == 6)
{
    double_down();
    hit();
}
```

Tips

- A control statement (actually any statement) can be written in one line or multiple lines. Separate blocks for clarity & readability.
- Keywords true (1) and false (0) available for C99 and after.
- Multiple statements will be treated as one when enclosed in a pair of curly brackets, { }.

More Boolean operators

- Logical AND && : check if both conditions are met

```
if ((dealer_up_card == 6) && (hand == 11))
    double_down();
```

- Logical OR || : check if one of two conditions are met

```
if (cupcakes_in_fridge || chips_on_table)
    eat_food();
```

- Logical NOT ! : flips the value of a condition

```
if (!brad_on_phone)
    answer_phone();
```

Yes, *not equal* writes

!=

Do we need the parentheses/brackets?

C Operator Precedence (incomplete)

Precedence	Operator	Description
1	++ -- () []	Suffix/postfix increment and decrement Function call Array subscripting
2	!	Logical NOT
3	* / %	Multiplication, division, and remainder
4	+ -	Addition and subtraction
5	< <= > >=	Relational operators < and ≤ respectively Relational operators > and ≥ respectively
6	== !=	Relational = and ≠ respectively
7	&&	Logical AND
8	||	Logical OR
9	= += -=	Simple assignment Assignment by sum and difference

It never hurts to add unnecessary parentheses!

```c
int main()
{
    char card_name[3];
    puts("Enter the card_name: ");
    scanf("%2s", card_name);
    int val = 0;
    if (card_name[0] == 'A') {
        val = 1;
    } else if (card_name[0] == 'J') {
        val = 11;
    } else if (card_name[0] == 'Q') {
        val = 12;
    } else if (card_name[0] == 'K') {
        val = 13;
    } else {
        val = atoi(card_name);
    }
    /* Check if the value is 2 to 6 */
    if ( (val > 1) && (val < 7)  )
        puts("Small Card!");
    /* Otherwise check if the card is 10, J, Q, K, or A */
    else if ( (val > 9) || (val == 1) )
        puts(“Large Card!");
    return 0;
}
```