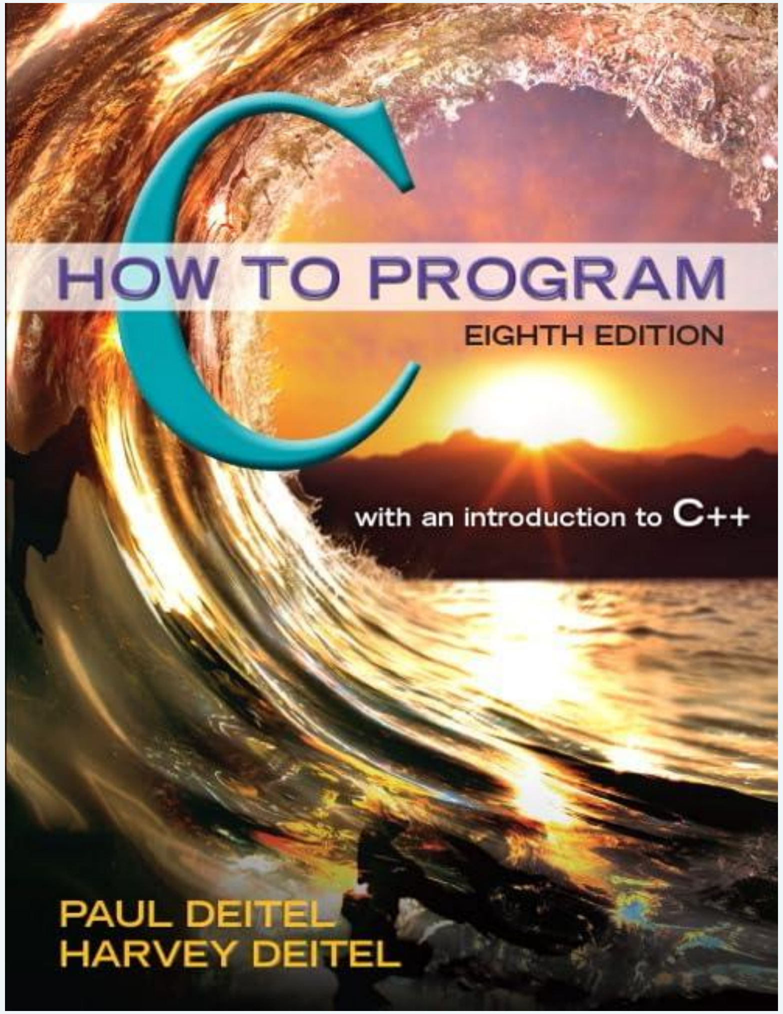
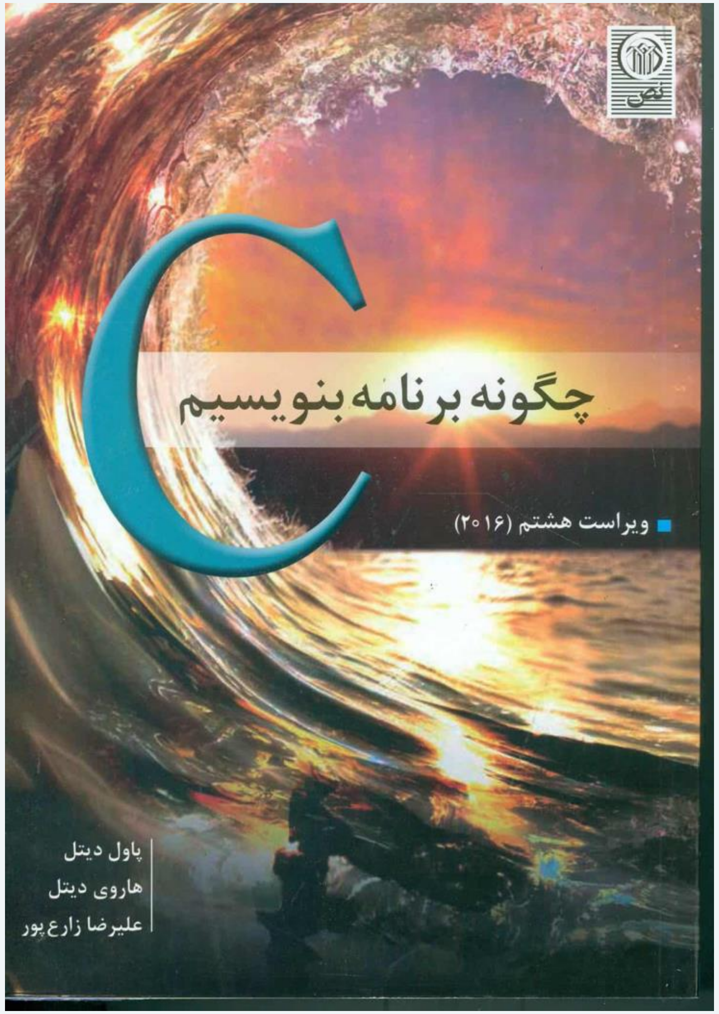
1

INTRO. TO PROGRAMMING

مراجع


C
HOW TO PROGRAM
EIGHTH EDITION
with an introduction to C++
PAUL DEITEL
HARVEY DEITEL


C
چگونه برنامه بنویسیم
ویراست هشتم (۲۰۱۶)
پاول دیتل
هاروی دیتل
علیرضا زارع پور
نص

کلیات

- بارم بندی:

5 نمره پایان ترم

4 میان ترم

6 پروژه

2.5 تمرینها

2.5 کوییزها

2. راه ارتباطی Yasamansabahi@gmail.com

3. برای دریافت اسلایدها و نمونه کدها :

https://github.com/Ysabahi/Computer-Programming-C

Outline

آشنایی کلی با برنامه نویسی

آشنایی با برنامه نویسی C

مفاهیم اولیه و پایه، مقدمات برنامه نویسی

فرمت بندی ورودی و خروجی

آشنایی با الگوریتم، فلوچارت و شبه کد

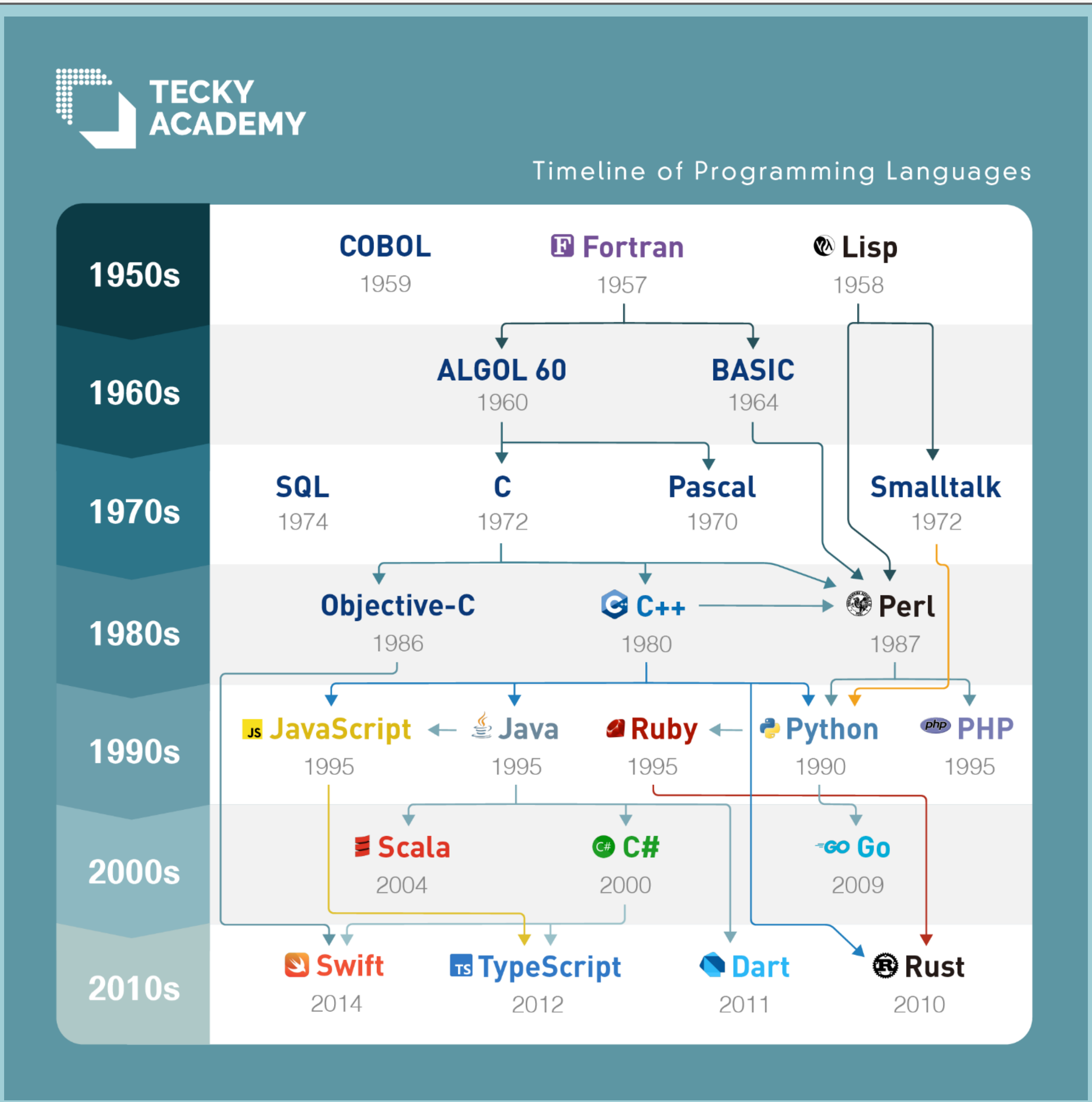
عبارات و دستورالعمل ها

توابع

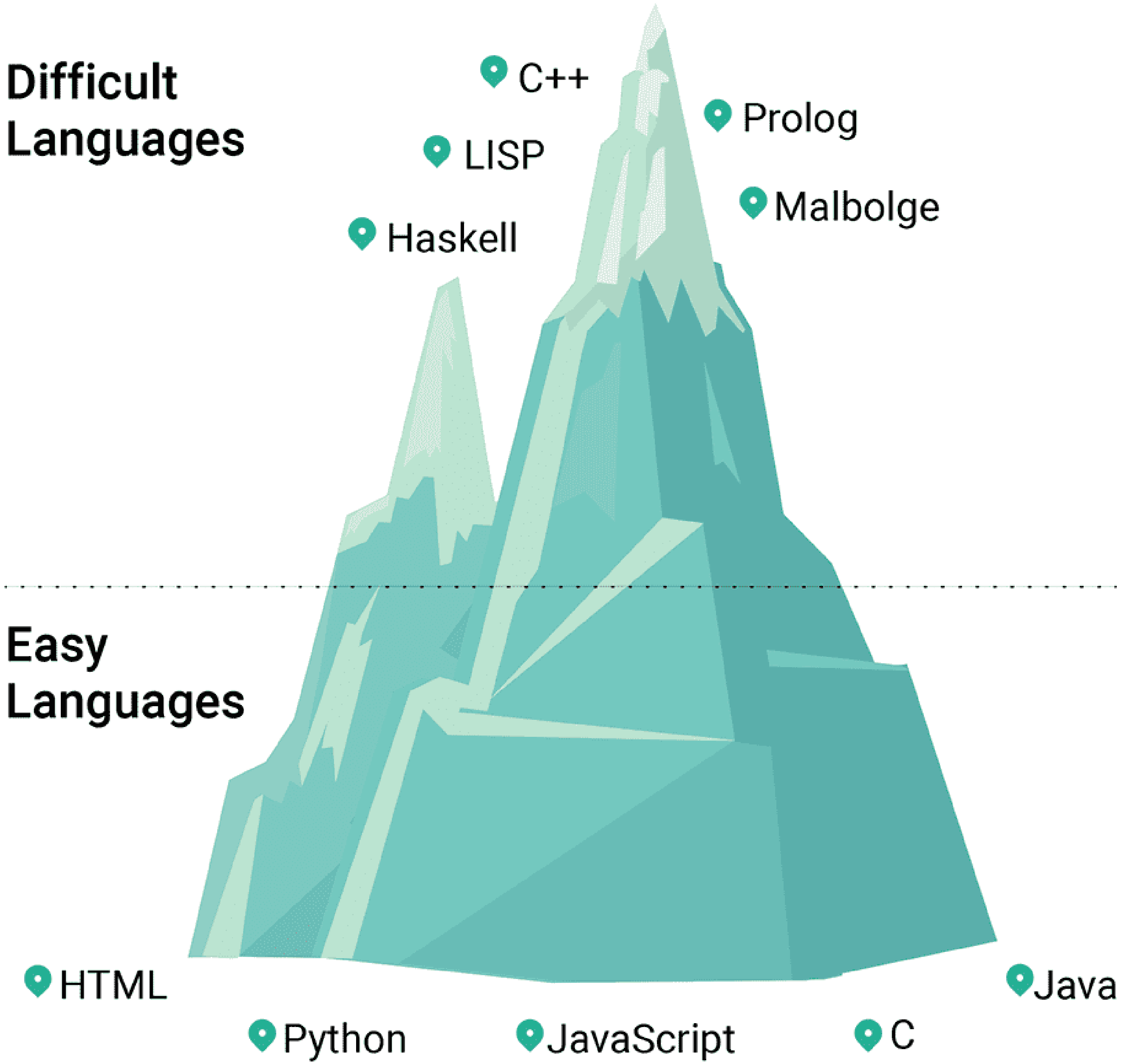
عیب یابی

آرایه ها، اشاره گرها و ...

ورودی و خروجی با فایل ها


TECKY
ACADEMY
Timeline of Programming Languages
1950s
COBOL
1959
Fortran
1957
Lisp
1958
1960s
ALGOL 60
1960
BASIC
1964
1970s
SQL
1974
C
1972
Pascal
1970
Smalltalk
1972
1980s
Objective-C
1986
C++
1980
Perl
1987
1990s
JavaScript
1995
Java
1995
Ruby
1995
Python
1990
PHP
1995
2000s
Scala
2004
C#
2000
Go
2009
2010s
Swift
2014
TypeScript
2012
Dart
2011
Rust
2010

◀ ▶



Difficult Languages

C++

Prolog

LISP

Malbolge

Haskell

Easy Languages

HTML

Java

Python

JavaScript

C

C

Pros:

Performance: C is compiled to machine code, making it very fast and efficient for system-level programming.

Control: Provides low-level access to memory and system resources, allowing for fine-tuned performance.

Portability: Code can be compiled on different platforms with minimal changes.

Foundation for Other Languages: Many modern languages (like C++, Java, and Python) are influenced by C, making it essential for understanding programming concepts.

Large Community: A long-standing community with extensive libraries and tools.

Cons:

Complex Syntax: More difficult for beginners due to its strict syntax and manual memory management.

Error-Prone: Manual memory management can lead to memory leaks and buffer overflows.

Less Flexibility: Requires more code for simple tasks compared to higher-level languages.

mid-level

Python

Pros:

Ease of Use: Simple and readable syntax makes it beginner-friendly and promotes rapid development.

Dynamic Typing: No need to declare variable types explicitly, allowing for more flexibility.

Rich Libraries: Extensive standard library and third-party packages for various applications (e.g., web development, data science).

Automatic Memory Management: Built-in garbage collection simplifies memory management.

Versatile: Widely used in many fields, including web development, data analysis, and artificial intelligence.

Cons:

Performance: Generally slower than C due to being an interpreted language.

Less Control: Higher-level abstractions can limit control over system resources and performance.

Dynamic Typing Risks: Flexibility can lead to runtime errors that are harder to debug.

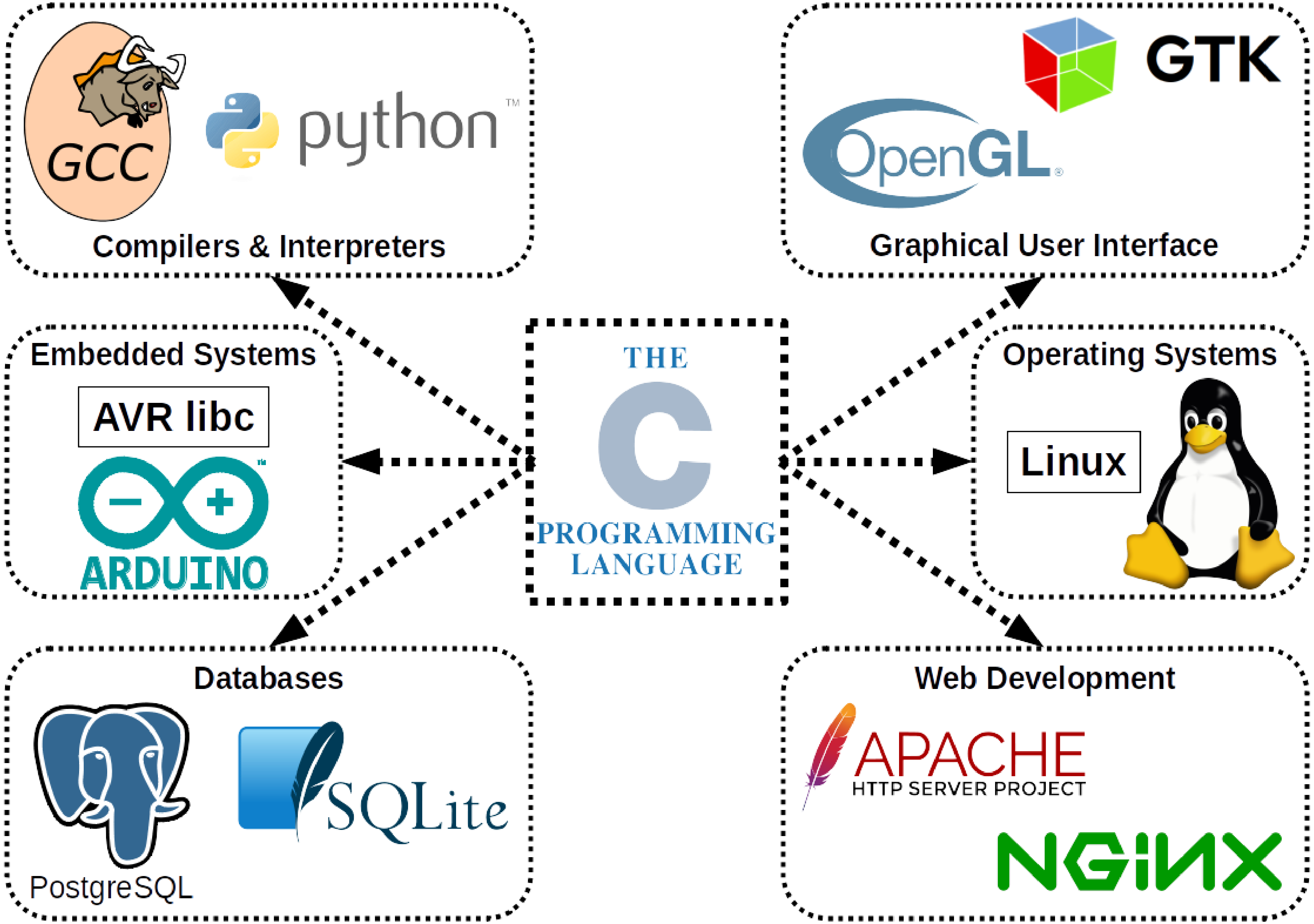
high-level

چرا زبان C؟

- تعریف C :یک زبان برنامه‌نویسی همه‌منظوره و رویه‌ای است.

- موارد استفاده :برنامه‌نویسی سیستم‌ها، سیستم‌های تعبیه‌شده، توسعه بازی و غیره.

- اهمیت :پایه‌ای برای بسیاری از زبان‌های مدرن مانندC++ ، جاوا و پایتون .



Compilers & Interpreters

Graphical User Interface

Embedded Systems

AVR libc

Operating Systems

Linux

Databases

Web Development

تاریخچه زبان C

- توسعه‌دهنده :دنیس ریچی در آزمایشگاه‌های بل.
- سال :۱۹۷۲.
- هدف :ایجاد برای توسعه سیستم‌عامل.UNIX
- تکامل :تأثیرگذار بر بسیاری از زبان‌های بعدی .

ویژگی‌های C

- قابل حمل بودن: می‌تواند روی ماشین‌های مختلف با تغییرات کم اجرا شود.
- بهره‌وری: کنترل بر روی سخت‌افزار و منابع سیستم را فراهم می‌کند.
- انعطاف‌پذیری: از برنامه‌نویسی سطح بالا و پایین پشتیبانی می‌کند.
- ماژولار بودن: امکان سازماندهی کد در توابع و ماژول‌ها را فراهم می‌کند.
- کتابخانه غنی: دارای مجموعه‌ای قوی از توابع داخلی است.

مفاهیم اولیه

همگردان یا کامپایلر برنامه یا مجموعه‌ای از برنامه‌های کامپیوتری است که متنی از زبان برنامه نویسی سطح بالا (زبان مبدا) را به زبانی سطح پایین (زبان مقصد)، مثل اسمبلی یا زبان ماشین، تبدیل می‌کند. خروجی این برنامه ممکن است برای پردازش شدن توسط برنامه دیگری مثل پیوند دهنده مناسب باشد یا فایل متنی باشد که انسان نیز بتواند آن را بخواند.

یک IDEیا به طور کامل محیط توسعه یکپارچه که مخففی از integrated development environmentمی-باشد. برنامه نرم افزاری است که برای کمک به برنامه نویسان و توسعه دهندگان جهت ساخت نرم افزار طراحی شده است. اکثر IDEها شامل یک ویرایشگر کد منبع , یک یا چند کامپایلر و یک اصلاح کننده خطا میباشند .

کامپایلر وظیفه ی تبدیل کد های برنامه نویسی به زبان قابل فهم ماشین را برعهده دارد اما IDEیک نرم افزار کمکی برای راحتتر شدن برنامه نویسی است . بی شک زبان Cو C++جزء قدرتمندترین و مشهورترین زبان های برنامه نویسی جهان هستند و کامپایلر ها و IDEهای بسیاری برای آن ها عرضه شده است . که تعداد محدودی از آن ها دارای محبوبیت و قدرت کافی هستند

کامپایلر های محبوب زبان C و ++C

1. **MinGw:** نرم افزار MinGwکامپایلر مخصوص مایکروسافت میباشد که فقط از ویندوز پشتیبانی میکند و برای C RunTimeو برخی دیگر از زبان های RunTimeمیباشد .
2. **GCC:**نرم افزار GCCیک کامپایلر رایگان زیر نظر GNUمیباشد که نه تنها کد های C – C++را کامپایل میکند بلکه از زبان های Ada , Objective_C , Javaو ...... نیز پشتیبانی میکند .
3. **Tiny C Compiler:** نرم افزار TCCیکی از بهترین کامپایلر های Cمیباشد که از تمامی پیش پردازنده ها پشتیبانی میکند و در آن از اسمبلر GNUاستفاده شده است . لازم به ذکر است که اسمبلر GNUیکی از بهترین اسمبلر های جهان است .
4. **Ideone:** نرم افزار Ideoneیک IDEو کامپایلر آنلاین میباشد که از C – C++و 60 زبان دیگر پشتیبانی میکند .

معرفی انواع IDEکاربردی در زبان Cو C++

Visual studio: از قابلیت های VSمیتوان به برنامه نویسی برای موبایل , وب و دکستاپ اشاره کرد و پشتیبانی از زبان های بسیاری هم چون , Asp.net , Basic , C# , C++ , C , Css , Xml , Ruby , JavaScript , Pythonو ..... و هم چنین قابلیت های بیشمار دیگر اما از بدی های آن میتوان پشتیبانی نکردن از دیگرسیستم عامل ها و کامپایلر ها , حجم بسیار زیاد و قیمت سرسام آور آن اشاره کرد .

Code: ادیتور C::Bیک ادیتور مخصوص C-C++است که البته در نگارش جدید آن Fortran نیز اضافه شده است سرعت بالا پشتیبانی از تمام سیستم عامل ها , کامپایلرها , حجم بسیار کم و همچنین رایگان و متن باز (Open Surce) بودن آن , آن را در بین برنامه نویسان بسیار محبوب کرده است .

Kdevelop: ادیتور Kdیک ادیتور C - C++رایگان متن باز و کم حجم برای سیستم عامل های خانواده ی لینوکس و Macمیباشد . این ادیتور از فریم ورک قدرتمند Qtنیز پشتیبانی میکند و البته نسخه های مختلفی از آن برای پشتیبانی از زبان های Phpو Pythonنیز ارائه شده است . از بدی های این ادیتور میتوان پشتیبانی نکردن از سیستم عامل محبوب ویندوز نام برد .

IDEs!

- ACK
- Borland Turbo C
- Clang
- GCC
- ICC
- LCC
- Norcroft C
- PCC
- SDCC
- TCC
- Visual Studio,

1. Code::Blocks

- Platforms: Windows, Mac, Linux
- Features: Customizable interface, support for multiple compilers, and a powerful debugger.

2. Eclipse CDT

- Platforms: Windows, Mac, Linux
- Features: Extensive plugin support, integrated debugging, and project management tools.

3. CLion

- Platforms: Windows, Mac, Linux
- Features: Smart code completion, powerful refactoring tools, and integrated debugger. (Paid)

4. Visual Studio

- Platforms: Windows
- Features: Excellent debugging tools, GUI design capabilities, and a rich set of extensions. (Community version available for free)

5. Xcode

- Platforms: Mac
- Features: Integrated development environment for macOS with a suite of software development tools.

6.NetBeans

- Platforms: Windows, Mac, Linux
- Features: Supports various languages, including C, with a simple interface and good debugging tools.

7. Atom

- Platforms: Windows, Mac, Linux
- Features: Highly customizable with packages for C programming, lightweight, and user-friendly.

8. Geany

- Platforms: Windows, Mac, Linux
- Features: Lightweight and fast, with basic IDE features and support for multiple programming languages.

2

INTRO. TO PROGRAMMING

Programming Languages

- There are three types of programming Languages

1) Machine Languages (machine codes):

- Strings of 1s and 0s.
- Only unterstood by integrated circuits, such as microprocessors.

Example:

```
10100010
01011011
10101010
```

2) Assembly Languages:

- English-like abbreviations representing elementary computer operations.
- translated to machine code by using assemblers.

Example:

```
MOV AL,3BH
ADD AL, AH
SUB AL,AH
MOV [SI]
```

Programming Languages

- There are three types of programming Languages

1) Machine Language
2) Assembly Languages

3) High-level Languages:
   - Codes similar to everyday English
   - Use mathematical notations
   - translated to machine code by using compilers.
   - C, C++, PASCAL, FORTRAN, BASIC are high-level languages.

Example:

```
c=a+b; if(a<b)
    printf(“a is less than b\n”);
    else
    printf(“a is NOT less than b\n”);
```

Structured programming

- Disciplined approach to writing programs
  - Using flowcharts (graphical representation)
  - Using pseudocodes or step by step algorithms.
- Clear, easy to test and debug and easy to modify
  - Using functions for efficient programming.

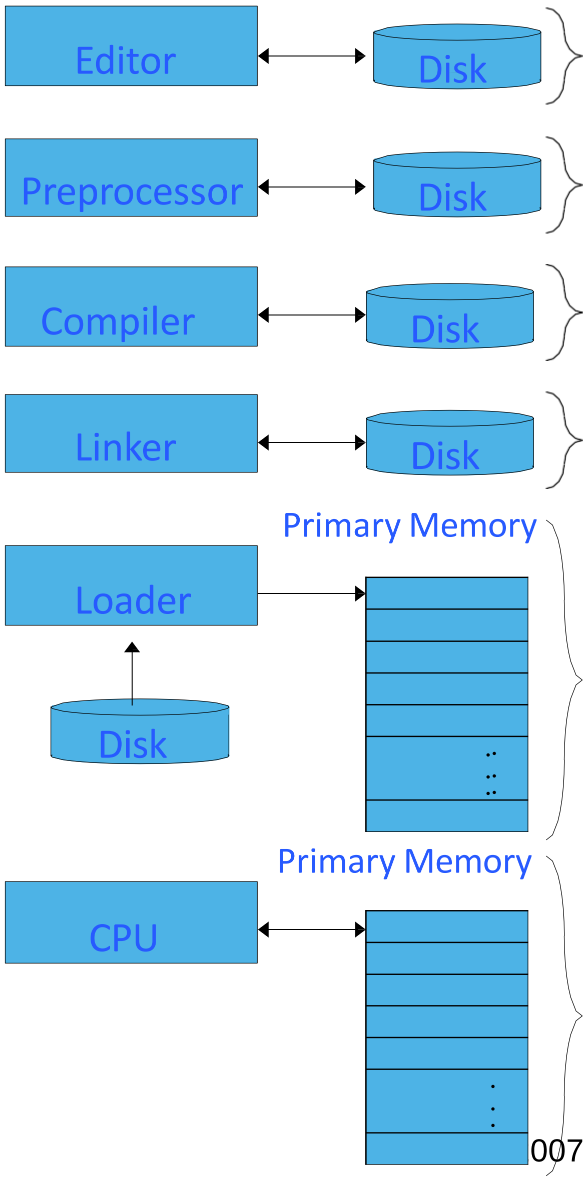
Multitasking

Specifying that many activities run in parallel.

Basics of a Typical C Program Development Process

- **Phases of C Programs:**

1. **Edit**
2. **Preprocess**
3. **Compile**
4. **Link**
5. **Load**
6. **Execute**



Program is created in the editor and stored on disk.

Preprocessor program processes the code.

Compiler creates object code and storesi it on disk.

Linker links the object code with the libraries

Loader puts program in memory.

CPU takes each instruction and executes it, possibly storing new data values as the program executes.

Simple C Program:

- The following program displays "Hello World" on the computer screen (monitor).

```
/* This is our first program in C Language */
#include <stdio.h>
int main()
{
   printf("Hello World\n");
return 0;
}
```

- The program output

```
Hello World
```

Simple C Program:

```
/* This is our first program in C Language */
#include <stdio.h>
int main()
{
   printf("Hello World\n");
return 0;
}
```

Comments:

- Text surrounded by `/*` and `*/` is ignored by computer.
- Used to describe program.

`#include <stdio.h>`

Preprocessor directive:

- Tells computer to load contents of a header file `<stdio.h>`,
- which includes standard input/output functions.
- For example `printf()` is one of the standard input/output functions.

Simple C Program:

```
/* This is our first program in C Language */
#include <stdio.h>
int main()
{
   printf("Hello World\n");
return 0;
}
```

`int main()`

- C programs contain one or more functions,
- One of the functions must be `main().`
- Parenthesis used to indicate a function
- `int` means that `main` "returns" an integer value
- Braces `(`{ and `})` indicate a block
- The bodies of all functions must be contained in braces.

Simple C Program:

```
/* This is our first program in C Language */
#include <stdio.h>
int main()
{
   printf("Hello World\n");
return 0;
}
```

`printf("Hello World\n");`

- `printf()` function print the string of characters within quotes `(" ")`
- All statements must end with a semicolon `(;)`
- `\n` is the newline character.

`return 0;`

- A way to exit a function.
- `return 0`, in this case, means that the program terminated normally.

`Right brace }`

- Indicates end of main has been reached.

Simple C Program:

- **Example 1:** Write a C program which displays your name and surname in two consecutive lines .

- **Example 2:** Write a C program which displays the following lines.

```
Today
is a
nice


day
```

C Program: Addition of two integer numbers

```
/* This program adds two integer numbers */
#include <stdio.h>
int main()
{
  int a, b, sum;         /* variable declarations */
  printf("Enter first integer\n"); /* prompt the user */
  scanf( "%d", &a);           /* read first integer */
  printf("Enter second integer\n"); /* prompt the user */
  scanf( "%d", &b);           /* read second integer */
  sum = a + b;           /* calculate the sum */
  printf( "Sum = %d\n", sum );   /* print the calculated sum*/
return 0; /* indicate that program ended successfully */
}
```

```
Enter first integer
15
Enter second integer
26
Sum = 41
```

Program Output

C Program: Addition of two integer numbers

```
int a, b, sum;
```

- Declaration of variables

  - Variables: locations in memory where a value can be stored

- `int` means the variables can hold integer numbers (-1, 3, 0, 47)

- Variable names (identifiers)

  - `a, b, sum;`

  - Identifiers: consist of letters, digits (cannot begin with a digit) and underscores(_). They are Case sensitive

- Declarations appear before executable statements

  - If an executable statement references and undeclared variable it will produce a syntax (compiler) error.

C Program: Addition of two integer numbers

```
scanf( "%d", &a );
```

- Obtains(reads/inputs) a value from the user
    - `scanf` uses standard input (usually keyboard)

- This `scanf` statement has two arguments

    `%d` - indicates data should be a decimal integer

    `&a` – location (address) in memory to store variable `a`.

        `&` is confusing in beginning – for now, just remember to include it with the variable name in `scanf` statements.

- When executing the program the user responds to the `scanf` statement by typing in a number, then pressing the *enter* (return) key.

C Program: Addition of two integer numbers

`=` (assignment operator)

- Assigns a value to a variable
- Is a binary operator (has two operands)

```
sum = a + b;
```

sum gets a + b;

- Variable receiving value on left

```
printf( "Sum is %d\n", sum );
```

- Similar to `scanf`
  - `%d` means decimal integer will be printed
  - `sum` specifies what integer will be printed
- Calculations can be performed inside `printf` statements

```
printf( "Sum is %d\n", a + b );
```

C Program: Addition of two integer numbers

- **Example 3:** Write a C program which calculates and displays the addition of integers 7, 8 and 14..

- **Example 4:** Write a C program which asks the user to enter 3 integer numbers and outputs the sum of these three numbers.

◀ ▶

Arithmetic Operations in C

- **Arithmetic Calculations:**
  - Use `*` for multiplication and `/` for division
  - Integer division truncates remainder

    `7 / 5` evaluates to `1`
  - Modulus operator(%) returns the remainder of modular division.

    `7 % 5` evaluates to `2`

- **Operator precedence:**
  - Some arithmetic operators act before others (i.e., multiplication before addition)
  - Use parenthesis when needed
  - Example: Find the average of three variables a, b and c

    Do not use: `a + b + c / 3`

    Use: `(a + b + c ) / 3`

Arithmetic Operations in C

- **Arithmetic operators:**

C operation	Arithmetic operator	Algebraic expression	C expression
Addition	`+`	***f + 7***	`f + 7`
Subtraction	`-`	***p – c***	`p - c`
Multiplication	`*`	***bm***	`b * m`
Division	`/`	***x / y***	`x / y`
Modulus	`%`	***r mod s***	`r % s`

Arithmetic Operations in C

- **Rules of Operator Presidence:**

Operator(s)	Operation(s)	Order of evaluation (precedence)
`()`	**Parentheses**	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
`*`, `/`, or `%`	**Multiplication,Division, Modulus**	Evaluated second. If there are several, they are evaluated left to right.
+ or **−**	**Addition Subtraction**	Evaluated last. If there are several, they are evaluated left to right.

Decision Making: Equality and Relational Operators

- **Executable statements**
  - Perform actions (calculations, input/output of data)
  - Perform decisions
    - May want to print `"pass"` or `"fail"` given the value of a test grade
- **if control structure**
  - Simple version in this section, more detail later
  - If a condition is `true`, then the body of the `if` statement executed
    - 0 is `false`, non-zero is `true`
  - Control always resumes after the `if` structure
- **Keywords**
  - Special words reserved for C
  - Cannot be used as identifiers or variable names

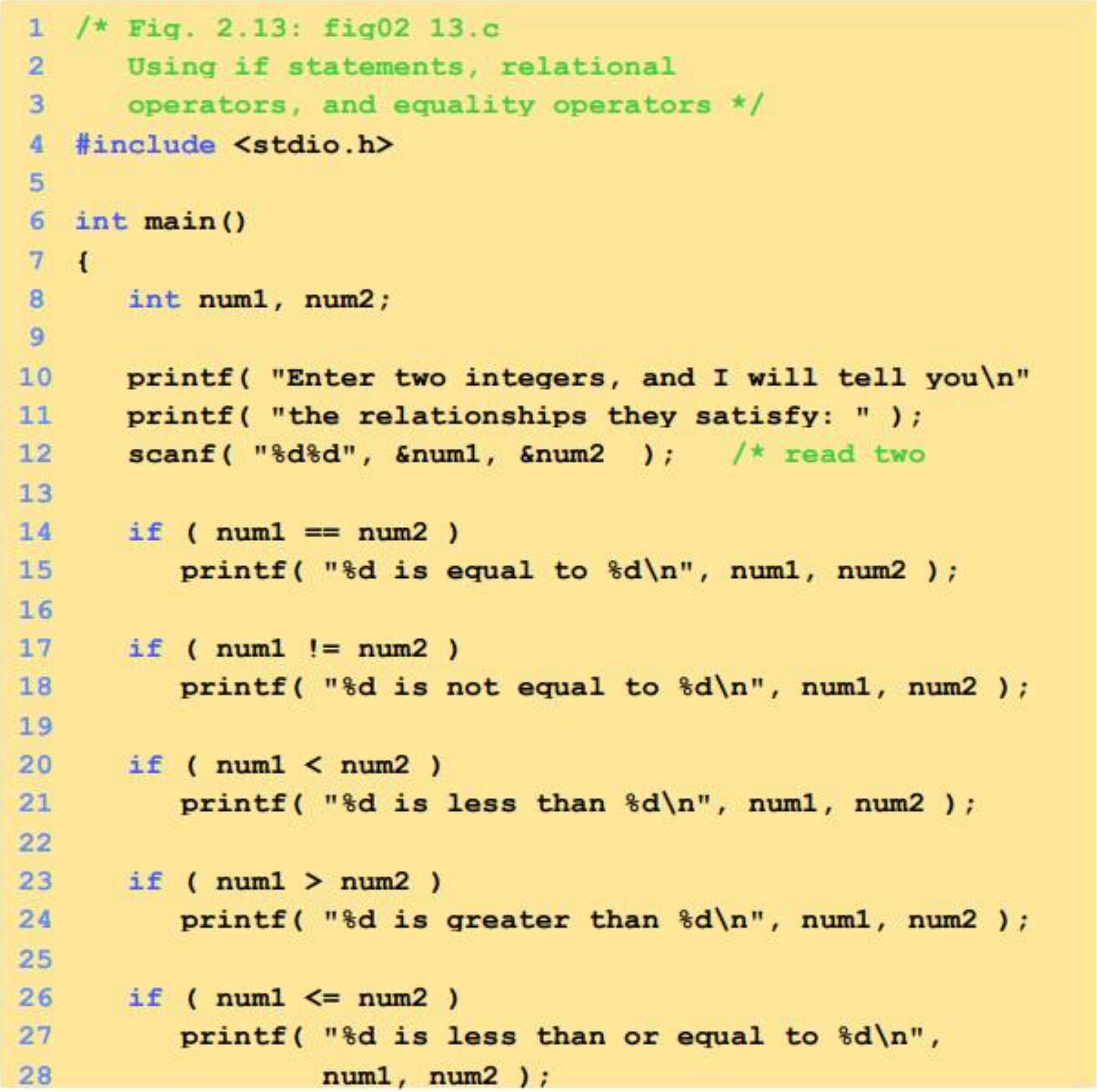
Decision Making: Equality and Relational Operators

Standard algebraic equality operator or relational operator	C equality or relational operator	Example of C condition	Meaning of C condition
Equality Operators			
=	`==`	`x == y`	**x** is equal to **y**
not =	`!=`	`x != y`	**x** is not equal to **y**
Relational Operators			
>	`>`	`x > y`	**x** is greater than **y**
<	`<`	`x < y`	**x** is less than **y**
>=	`>=`	`x >= y`	**x** is greater than or equal to **y**
<=	`<=`	`x <= y`	**x** is less than or equal to **y**

Decision Making: Equality and Relational Operators

Keywords			
`auto`	`double`	`int`	`struct`
`break`	`else`	`long`	`switch`
`case`	`enum`	`register`	`typedef`
`char`	`extern`	`return`	`union`
`const`	`float`	`short`	`unsigned`
`continue`	`for`	`signed`	`void`
`default`	`goto`	`sizeof`	`volatile`
`do`	`if`	`static`	`while`

Decision Making: Equality and Relational Operators



```c
/* Fig. 2.13: fig02 13.c
   Using if statements, relational
   operators, and equality operators */
#include <stdio.h>

int main()
{
   int num1, num2;

   printf( "Enter two integers, and I will tell you\n"
   printf( "the relationships they satisfy: " );
   scanf( "%d%d", &num1, &num2  );    /* read two

   if ( num1 == num2 )
      printf( "%d is equal to %d\n", num1, num2 );

   if ( num1 != num2 )
      printf( "%d is not equal to %d\n", num1, num2 );

   if ( num1 < num2 )
      printf( "%d is less than %d\n", num1, num2 );

   if ( num1 > num2 )
      printf( "%d is greater than %d\n", num1, num2 );

   if ( num1 <= num2 )
      printf( "%d is less than or equal to %d\n",
              num1, num2 );
```

Program Outline

1. Declare variables

2. Input

2.1 `if` statements

3. Print

Decision Making: Equality and Relational Operators

Program Outline

```
   if ( num1 >= num2 )
      printf( "%d is greater than or equal to %d\n",
              num1, num2 );

   return 0;   /* indicate program ended successfully */

}
```

3.1 Exit `main`

Program Output

```
Enter two integers, and I will tell you
the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

Decision Making: Equality and Relational Operators

- **Example 5:** Write a C program which asks the user to enter two integers, compare them and perform the following actions:

  - if the first value is greater -> add the two numbers,

  - if the second value is greater -> multiply the integers

  - if they are equal -> divide their multiplication with their sum.