



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

CPCS 301 – Programming languages
2nd semester 2020
Summary for the final exam

هذا التلخيص هو جهد احد الطلاب فـ راح اوضح نقطتين

أولاً: هذا التلخيص لا يغني عن السلايدات او الكتاب بالعكس انا هنا ملخص تقريبا 90 بالمية من الافكار وشارحها بتفصيل, الافضل بعد ماتخلص قراءة الملخص تروح تمر على السلايدات وباذن الله تكون لميت المنهج كله

ثانياً: انا طالب مثلثي مثلكم وهذا مجرد اجتهاداً مني, وهذا مايعني انه كل الكلام الي شرحته صح ولكن انا بحثت ودورت ونقحت المعلومات وكتبت الاصح ولكن ان كان فيه خطأ او شيء كلموني ع الواتس او اي وسيلة تواصل اخرى وراح اعدله باذن الله

عبدالله الكثيري

Introduction

- A **data type** defines a collection of data objects and a set of predefined operations on those objects
- A **descriptor** is the collection of the attributes of a variable
- An **object** represents an instance of a user-defined (abstract data) type

	Data type descriptor
هيكلة من البيانات الوصفية التي تصف البيانات يعني مجموعة بيانات مع بعض هدفها انها تصف بيانات اخرى وهي تكون احد الاجزاء في ال simple table	
الـ object هو عباره عن data-type زي ال integer, string المستخدم (زي لمن كنا نسوي linked list في برمجة 3 كانت عباره عن نوع بيانات بس احنى بنفسنا نعرفه ونحدد خواصه وكذا, مايكون جاهز في اللغة نفسها زي integer, string	object

Primitive Data Types

1. Almost all programming languages provide a set of **primitive data types**
2. Primitive data types: Those not defined in terms of other data types
3. Some primitive data types are merely reflections of the hardware
4. Others require little non-hardware support

1- يقولك ان تقريبا كل لغات البرمجة تحتوي مجموعة من ال primitive data types

2- ويقولك ال primitive data types ماتعتمد او مايتم تعریفها او تكوینها بالأعتماد على آخر data type

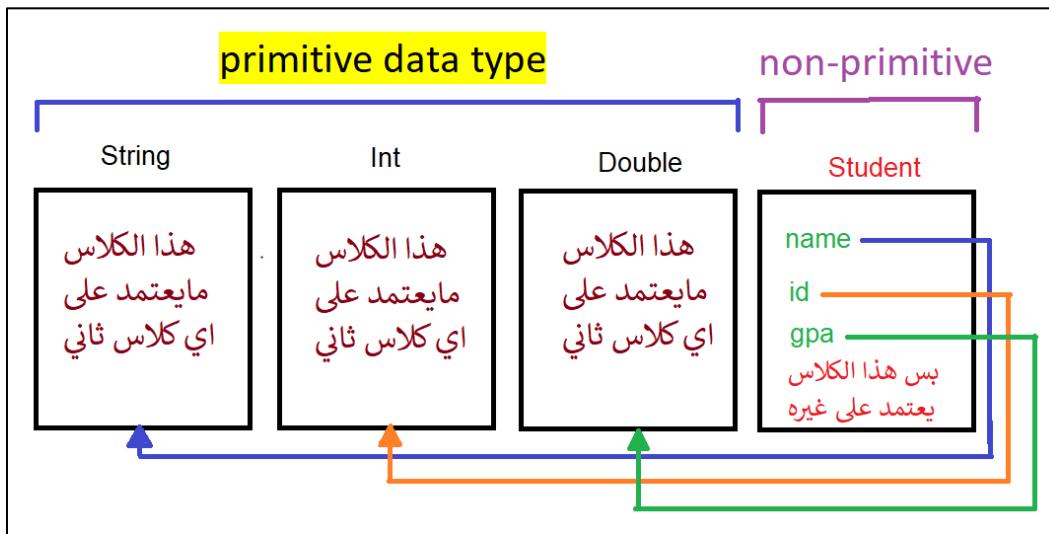
ايش يعني؟... ناخذ مثال مثلا بسوی student من نوع data type وحطیت فيه هذی ال data members

string	name
int	id
double	gpa

هذا يقولك ان هذا ال student مایعتبر primitive data type ليش؟؟ لأنه اصلا مكون من ثانية data types

زي ال string, int, double فهو اصلاً معتمد على اشياء ثانية بداخله

هذی صوره بسيطة رسمتها لكم، ان شاء الله توصلكم الفكرة



3- يقولك بعض ال primitive data types تعتمد في عملها على ال hardware

ايش يعني؟... يعني زي ال array في ال C language

لمن تحط لها index برا ال range حقها راح تجيب لك قيمة اخرى من ال memory

فهنا ال datatype ذي صارت تعتمد على ال hardware

4- يقولك ان بعض ال primitive data types تعتمد على اشياء ماهي hardware (يعني عكس النقطة 3)

هنا راح يبدأ يتعمق شوي في primitive data types ويعطيك بعض الأمثلة ويشرحها لك ...

Integer او لا

Primitive Data Types: Integer

1. Almost always an exact reflection of the hardware so the mapping is trivial
 2. There may be as many as eight different integer types in a language
 3. Java's signed integer sizes: byte, short, int, long

1- يقولك انها اكثـر نوع يمـثل الـهـارـدوـير، نـسـرـحـهاـ شـوـي

تفتکرو ال array in C language ؟ كانت تمثل او تعكس صوره عمل الميموري

برضو الريجิسترارات الي درسناها في بنيان الحاسب (اتوقع الاغلب درس الماده) الريجیسترارات كانت تتمثل بالارقام وهذا يسهل عملها وتفاعلها مع الهايدروير

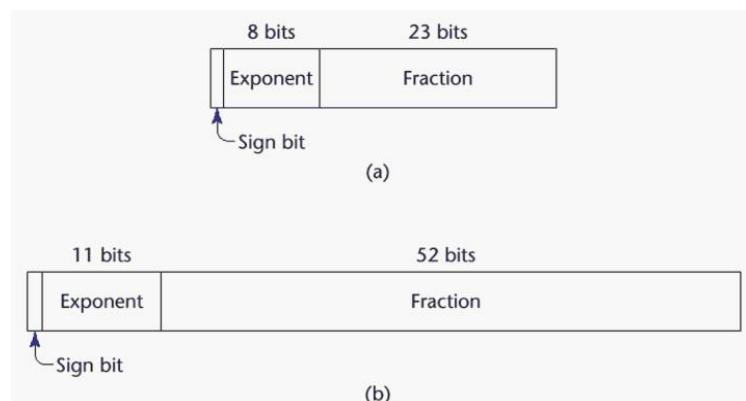
2- ممكن يكون عندك اكثر من نوع integer في نفس اللغة (اقرا النقطه رقم 3 بتفهم اكثـر)

3- يقولك انواع ال integers الي ف لغة java فالجافا عندها 4 انواع byte, short, int, long و ممكن لغات ثانية عندها اقل او اكثر من 4 integers

Floating point ثانپا

Primitive Data Types: Floating Point

1. Model real numbers, but only as approximations
 2. Languages for scientific use support at least two floating-point types (e.g., `float` and `double`; sometimes more)
 3. Usually exactly like the hardware, but not always
 4. IEEE Floating point standard 754



1- يتم تمثيل الأعداد الحقيقة بها (الأعداد الحقيقية يقصد فيها أي شيء فيه فاصلة)

2- اللغات العلمية او المستخدمة في الـ science على الاقل تستخدم نوعين وهم double, float وممكن اكث

3- يقلك إنها تشبه ال hardware، مو داماً

4- يقولك ان الى وضع هذى الرسمة الى ع اليمين والى خط المعايير هذى وقسمها هم منظمة IEEE

يعني بالعربي منظمة IEEE هم اللي حطوا كيف يتم تخزين ال floating point في الحاسوب من خلال نظامين الاول 32 بت والثاني 64 بت

ثالثاً Decimal

Primitive Data Types: Decimal

1. For business applications (money)
 - Essential to COBOL
 - C# offers a decimal data type
2. Store a fixed number of decimal digits
3. Advantage: accuracy
4. Disadvantages: limited range, wastes memory

1- تستخدم في ال business و المجالات الاعمال والأموال، زي لغتين هم COBOL, C#

2- تقدر تخزن فيها نوعين fixed number, decimal digits خلونا نفصلها زيادة شوي

يعني لمن يكون عندك عدد ثابت من الخانات سواء بعد الفاصله او قبلها، على سبيل المثال ناخذ الفلوس دائم الفلوس لمن تكتب ينكتب بس رقمين بعد الفاصله زي 5.25 او 17.29 دائم الهملات تجي رقمين بس بعد الفاصله ماتجي 3 او 4 هذا تعتبر fixed لأن احني عارفين عدد الخانات الي عندنا (ترا ممكن تكون fixed قبل او بعد الفاصله عادي ف اي مكان بس هذا مجرد مثال)	Fixed number
هذا الاعداد المتغيره اللي ماتدرى وش ممكن تخزن فيها زي 5246.2324458 او 21.1 اي شي يمشي	Decimal digits

فهمتواها ؟ اذا لا بفصلها لك زياده، الحين فل نفترض قلت لك كم معدلك ؟ فلنفترض انت قلت لي 4.3 او 4.33 بقولك اوه حلو ماشاء الله، طيب فلنفترض انك قلت لي 4.333 راح اقولك هاه ؟ راح استذكر الاجابة شوي

لأنني انا حاط بمخي انه دائم المعدل يكون رقم واحد يسار الفاصله وهو يكون 1 او 2 او 3 او 4 او 5 ويمين الفاصله دائم يكون فيه رقم او رقمين تبدا من 00 الى 99 **هذا مثال على Fixed number**
ل لكن لو مثلا كنت تقرأ كتاب فيزياء وجو قالولك ثابت سرعة الضوء هو 299792458.0 هنا ماراح تقدر تتوقع كم ممكن يكون طول الرقم، او كم ممكن يمين الفاصله او يسارها، **هذا مثال على Decimal digit**
3- يقولك ميزتها هي : انها صحيحة ودقيقة

4- عيبها : انها لها حد معين او لها range محدود، وكمان تضيع مساحة في ال memory

Boolean رابعاً

Primitive Data Types: Boolean

1. Simplest of all
 2. Range of values: two elements, one for "true" and one for "false"
 3. Could be implemented as bits, but often as bytes
- Advantage: readability

- ابسط شيء من بين الـ primitive data types كلهم
1- تحتوي بس على نوعين او عنصرين (true and false)
2- يمكن تمثيلها من خلال bytes او bits
3-

readability : ميزتها

Character خامساً

Primitive Data Types: Character

1. Stored as numeric codings
2. Most commonly used coding: ASCII
3. An alternative, 16-bit coding: Unicode
 - Includes characters from most natural languages
 - Originally used in Java
 - C# and JavaScript also support Unicode

1- يتم تخزينها كبيانات رقمية (زي الـ ASCII او الـ Unicode)
2- اكثر نظام رقمي يتم استخدامه هو الـ ASCII
3- وفيه نظام ثانوي اسمه Unicode ويستخدم نظام تخزين 16 بت
بالنسبة لل Unicode فيه اغلب الحروف والرموز الي موجودة في لغات البشر
يستخدم في الـ Java, C#, JavaScript

عشان تفهم اكثر شوف في النت جدول عن الـ Unicode, ASCII عشان تعرف وش نقصد بكلمة (بيانات رقمية)... راح تلاحظ انهم يخزنون في الميموري كأرقام مو كحروف

Character String Types

- Values are sequences of characters
- Design issues:
 1. Is it a primitive type or just a special kind of array?
 2. Should the length of strings be static or dynamic?

يقولك ان الـ **string** عباره عن **array of character**, طيب ايش يعني ؟
يعني فلنفترض انو ابغا اخزن اسم **Abdullah** في متغير من نوع **String**, راح يتم تخزينه بالشكل التالي

A	b	d	u	l	l	a	h
---	---	---	---	---	---	---	---

طيب نجي لـ **Design issues**

- يقولك هل هي عباره عن **array of characters** ؟ ام انها عباره عن **primitive data type** ؟
طيب هنا يقولك هل هي اصلا **primitive data type** يعني هل هي اصلا موجوده في اللجه بشكل
اساسي زي **الجافا والبايثون** ؟ ام انها حاله مخصوصه من الـ **array of characters** زي لغة **C**
الي قد استخدم لغة **C** قبل كذا , راح تلاحظ انك لمن تبغا تعرف string راح تسوّي array من نوع
و جوتها تخزن الـ **string characters**

The screenshot shows a terminal window with the following content:

```
1 #include<stdio.h>
2 int main()
3 {
4     char name[] = "abdu...ah";
5     printf("%s",name);
6     return 0;
7 }
```

Output:

```
abdullah
...
Program finished with exit code 0
Press ENTER to exit console.
```

وهذا مثال عشان توضح الصوره

طيب السلايدات الي بعدها راح يتكلم عن ال **regex** وهو يعتبر شي قديم وشرحته في الملخص الي فات
اذا ماقريت الملخص الي فات راح احط لك الرابط هناك مشرح فيه كل شي

https://drive.google.com/drive/folders/1uaCL57_qCEnv2a_cTuq8-rFO7N0Pj91S?usp=sharing

وكمان جاب كذا مثال عن ال regex في لغة بايثون, مع الجواب
طيب لو جاك سؤال في الاختبار عن ال regex كيف تجاوبه ؟
يا انك تكون فاهم وتجاوب من فهمك

او تستخد الموقعي الثاني تقدر تحط فيه اي regex ويطلع لك الجواب(برضو حطيته في الملخص الي فات)

<https://regex101.com/>

Character String Type in Certain Languages

- C and C++
 - Not primitive
 - Use `char` arrays and a library of functions that provide operations
- SNOBOL4 (a string manipulation language)
 - Primitive
 - Many operations, including elaborate pattern matching
- Java
 - Primitive via the `String` class

طيب هنا يتكلم عن ال `String` في اكثر من لغة

C,C++ - لغة 1

يقولك انها **not primitive** يعني ما هي موجودة في اللغة بشكل اساسي (شرحتها فوق)
وكمان يتم استخدام `array of characters` عشان نستخدم ال `string` (برضو شرحتها فوق)

Snobol4 - لغة 2

تعتبر **primitive** يعني مبنية موجودة باللغة بشكل اساسي, وتدعم `functions` كثيره تتعامل مع `String` السرينيق زي `length` وباقى الدوال الي ممكن تطبقها على اي متغير من نوع `String`

Java - لغة 3

تعتبر **primitive** من خلال الكلاس حق ال `String`

Character String Length Options

- **Static:** COBOL, Java's String class
- **Limited Dynamic Length:** C and C++
 - In C-based language, a special character is used to indicate the end of a string's characters, rather than maintaining the length
- **Dynamic (no maximum):** SNOBOL4, Perl, JavaScript
- **Ada supports all three string length options**

طيب هنا يتكلم عن نوع وطول المتغير String في اكثر من لغة
خلونا نرتلها فجدول عشان يطلع شكلها كيوت شوي

اللغات الي تستخدم	شرح	نوع الـString
Cobol, Java	لمن تعرف متغير وسميته Abdullah وبعدين تغير قيمته وتخلية Ahmed البوينتر راح يأشر على opject جديد وراح يسيب القديم, يعني عندي اوبجكت من نوع string وجوته قيمة وهي Abdullah وفيه بوينتر اسمه name يأشر عليه, حلوين ؟ طيب الحين جيت كتبت "name = "Ahmed" الى حيصير بالزبط انه راح نسوي opject جديد ونخزن جوته قيمة وهي ahmed والبوينتر الى كان اسمه name راح يأشر ahmed على الـ opject الجديد حقنا الى جوته قيمة ahmed وراح نطنسن الـ opject القديم	Static
C, C++	هنا ماراححتاج نتسوي opject جديد زي ماسوينا فوق في ال java, هنا راح نخزنهم في نفس ال opject وراح نغير الطول لأنه dynamic, يعني لو خزنا Abdullah راح يكون اوبجكت وطوله 8, ولو بعدين سوينا name = "ahmed" راح نخزن قيمة ahmed في نفس الـ opject القديم بس راح نغير طوله ويصير 5 بس حط فالاك ان الطول له limit معين, يعني ممكن يكون طوله اقصى شي 100 حرف او 150 حرف مانقدر تزود اكثر من كذا	Dynamic (limited)
Snobol4, Perl, JavaScript	زي الي فوق بس هنا ماله نهاية تقدر تسوی طوله قد ماتبعنا	Dynamic (not limited)

وعلى فكرة لغة Ada تدعم الثلاث انواع الي فوق كلها

بعدها فيه سلايدين كلها كلام نظري مايحتاج شرح تتكلم عن ال evaluation, implementation

هذا السترينج حقك

طوله

عنوانه في الميموري

Static string
Length
Address

Limited dynamic string

Maximum length

Current length

Address

هذا السترينج حقك

اقصى طول ممكن

طوله الحالي

عنوانه في الميموري

Compile-time
descriptor for
static strings

Run-time
descriptor for
limited dynamic
strings

في هذا السلايد بس يوريك شكل ال simple table كيف بيكون شكلها في حال لو كان static او كان limited dynamic

User-Defined Ordinal Types

- An ordinal type is one in which the range of possible values can be easily associated with the set of positive integers
- Examples of primitive ordinal types in Java
 - integer
 - char
 - boolean

طيب هنا بالمخترق المفید يقولك عندنا حاجة اسمها **Ordinal Type** وهي انواع بيانات متتابعة، طيب ايش يعني؟ وأيش استفيد انا؟

يعني بيانات لو اعطيتك هي راح تقدر تحبب القيم الي بعدها والي قبلها، طيب اديكم مثالين وراح تفهمون باذن الله

المثال الاول على **Ordinal data type**

مثلا قلت لك عندي متغير اسمه a ونوعه integer وعطيته القيمة 4، بعدين سأألك وقلت لك وش القيمة الي بعده؟ راح تقولي على طول 5، ليش طيب؟ لأنها تعتبر متسلسلة وورا بعض لمن اديك عنصر منه راح تعرف تحبب الي قبله او الي بعده، هم لهم تسلسل وزعي كذا

طيب مثلا قلت لك عندي حرف d وش الي بعده؟ بتقولي e لأنهم متتابعين ولهم ترتيب

المثال الثاني على **non-Ordinal data type**

مثلا قلت لك عندي متغير اسمه aaa ونوعه float وقيمتها هي 2.1 وقلت لك جيب الي بعده..... ماحتقدر تجاوب، لأنها تحتمل اكثر من اجابة ممكنته 2.2 او 2.11 او 2.111 او 2.1111

Enumeration Types

- All possible values, which are named constants, are provided in the definition
- C# example

```
enum days {mon, tue, wed, thu, fri, sat, sun};
```

يقولك ال Enumeration types بالختصر هي عباره عن set فيها elements والسلام عليكم زى ما انتو شايفين فوق عندنا set اسمها days وفيها elements وهي days ايش يعني طيب ؟
برضو يمدينا ناخذ منها subrange ايش يعني طيب ؟
تفتكر و الداله الي كان اسمها substring ؟ ماتفتكر و صح ؟ طيب نشرحها

هذا كود جافا

```
public class HelloWorld{  
    public static void main(String []args){  
        String name = "abdullah";  
        System.out.println(name.substring(1,4));  
    }  
}
```

وهذا الأوت بوت حقها

bdu

ما يحتاج شرح واضحة

المهم نرجع لموضوعنا, ال enumeration types زي ال string يمديك تسوى منها subrange

```
type Days is (mon, tue, wed, thu, fri, sat, sun);  
subtype Weekdays is Days range mon..fri;
```

وهذا مثال اخذ ایام الاسبوع كـ subrange من الـ Days

طيب بعدها بنخش في موضوع الـ array وكذا، طيب عندنا Design issues خلونا نشووفها

Array Design Issues

- What types are legal for subscripts? (index)
- Are subscripting expressions in element references range checked? هل يشيك على الرينج؟
- When are subscript ranges bound? تحديد طولها
في السي قبل الرن تايم، في الجافا خلال الرن تايم عادي
- When does allocation take place? متى يحدد مكانها في الميموري؟
الاجابة الرن تايم ويكون يا هيب او ستاك
- What is the maximum number of subscripts
maxint اخر اندركس
- Can array objects be initialized? Arr of obj
- Are any kind of slices allowed? Sub arrays
java -> no, matlab -> yes

اتمنى تكون واضحة...

Index Syntax

- FORTRAN, PL/I, Ada use **parentheses**
 - Ada explicitly uses parentheses to show uniformity between array references and function calls because both are *mappings*
- Most other languages use **brackets**

طيب هنا يتكلم عن الـ index

مثال: array(10) يقولك في Fortran, PL/I, Ada يستخدموا ال ()

مثال: array[10] وفي غالبية اللغات الاخرة يستخدموا ال []

في كل الحالتين بينادي العنصر العاشر في المصفوفه بس يختلف شكل القوس

Arrays Index (Subscript) Types

- FORTRAN, C: integer only
- Pascal: any ordinal type (integer, Boolean, char, enumeration)
- Ada: integer or enumeration (includes Boolean and char)
- Java: integer types only
- C, C++, Perl, and Fortran do not specify range checking
- Java, ML, C# specify range checking

طيب هنا يقولك كل لغة وأي الـ index حق الـ types الي ممكن نستخدمها ...
يعني مثلا سوينا [array[i]] هنا يقولك الـ i ايش ممكن يكون نوعها هل
طيب الي محدوده بالأخضر مشروحة واضحة

راح نتكلم شوي عن الي محدد بالسماوي

يقولك الـ range checking ماتسوبي C,C++,Perl,Fortran طيب وش يعني ؟
مثلا عندي اراي في لغة C وطولها 10 وجيست انت وطبعت التالى [array[100]
هنا ماراح يديك ايرور وراح يرجع لك قيمه عشوائية من الميموري, لأنه ما يتأكد من الـ range
والـ Java, ML, C# لو سويت نفس الي فوق وطبعت الاندكس رقم 100 راح يديك ايرور, لأنها تتحقق
هل الاندكس الي انت تبغاه جوا الرينج الي بين 0 و 10 او لا

طيب بعدها بيكلم عن الـ Categories للأندكس

بنحطها ف جدول عشان تربط امورنا

النوع	وصف	مثال
Stack-dynamic	حجم الاراي ومكانها في الميموري كلهم dynamically وتصير وقت ال bounded run-time	In Ada, you can use stack-dynamic arrays as Get(List_Len); declare List: array (1..List_Len) of Integer; begin ... end;
Fixed heap-dynamic	حجم الاراي ثابت مايتغير بس مكانها في الميموري متغير	In C/C++, using malloc/free to allocate/deallocate memory from the heap Java has fixed heap dynamic arrays C# includes a second array class ArrayList that provides fixed heap-dynamic
Heap-dynamic	هنا يقولك حجم الاراي ومكانها في الميموري متغيرين ميزتها: مرنة من ناحيه انك تكبر الاراي وقت ماتبغا وتصغرها وقت ماتبغا فما تستهلك الاراي اكبر من الحجم المطلوب	Perl, JavaScript, Python, and Ruby support heap-dynamic arrays Perl: @states = ("Idaho", "Washington", "Oregon"); Python: a = [1.25, 233, 3.141519, 0, -1]

ادري انها تلحس المخ شوي بس فالغالب الدكتور مايركرز على الاشياء النظرية ذي, حتى لو مافهمت سوي

سكيب مو مشكلة

وهنا بحط لكم مرجعين لو تبعوا تقرأ زياده و تستفيد

http://comp.eng.ankara.edu.tr/files/2015/09/subscript_binding_multid_arrays.pdf

<http://mpstudy.com/describe-subscript-bindings-and-array-categories/>

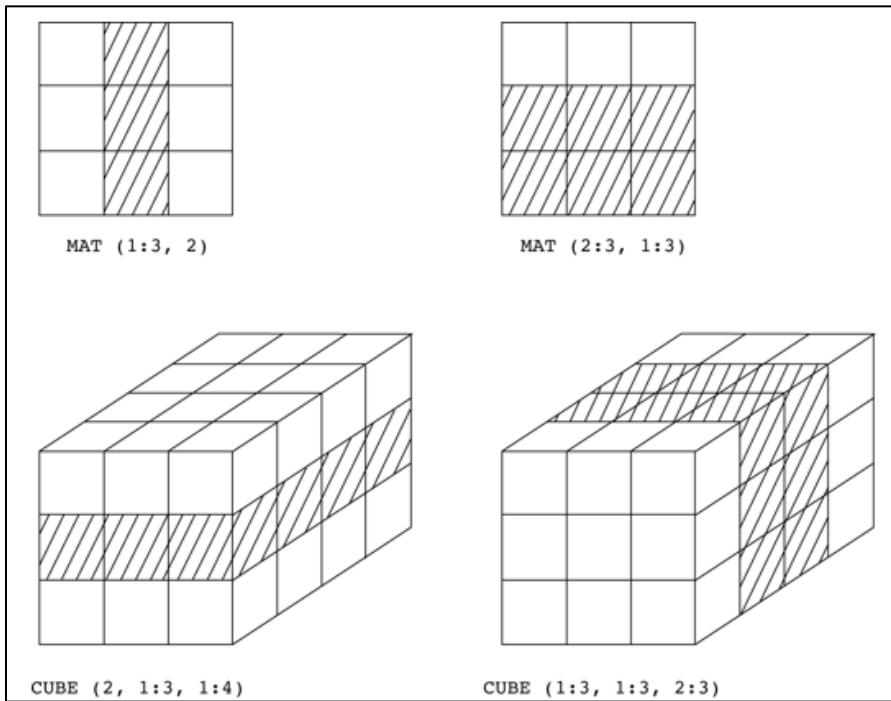
بعدها فيه كم سلايد يتكلم عن كيفية انشاء الاراي في بعض اللغات, اشياء سهله بس مر عليها قراءة ماتحتاج شرح

بعدها بيكلم عن شيئين

وهي الاراي المربعه الي عدد صفوفها واعمدتها زي بعض Rectangle array

وهي الي تكون مختلفة يعني ممكن الصف الاول فيه 3 اعمدة بس الصف الثاني فيه 5 Jagged array

درسنها في برمجة 2 لو فاكرينها



هنا يشرح مفهوم ال **Slices** وهي عباره عن قطعه او جزئية من ارای ثانيه زي ما هو واضح ف الصوره المربع المحدد بخطوط كثيره هي **slice** ومخوذة من الاراي الاساسية الى هي بالابيض

Implementation of Arrays

Access function maps subscript expressions to an address in the array

Access function for single-dimensioned arrays:

$$\text{address}(\text{list}[k]) = \text{address}(\text{list}[\text{lower_bound}]) + ((k-\text{lower_bound}) * \text{element_size})$$

يجب عدد الالمنتس بين اللور والالمنت المطلوب ويضربهم في مساحة الالمنت الواحد (بالجافا ال انتجر ب4)

يجب الادرس حق اول المنت في الاراي

$$\text{Ex: } \text{address}(\text{list}[5]) = \text{address}(\text{list}[0]) + ((5-0)*4) \\ \text{address}(\text{list}[5]) = 500 + 20 = 520$$

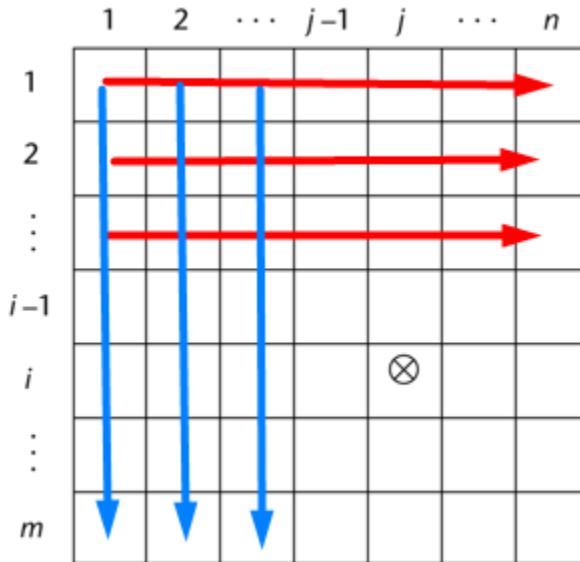
في الصوره ذي يشرح لك كيف تجيب ال address حق اي عنصر بس باستخدام الاندكس تبعه يعني لو قلت لك جيب ال address حق العنصر رقم 5 في الاراي الي اسمها list راح تسوى زي المثال الي فوق

هنا يقولك الاراي لمن تكون ... $2d, 3d, 4d$ لها نظامين في انك تتبع العناصر

Two common ways:

- Row major order (by rows) – used in most languages
- column major order (by columns) – used in Fortran

طريقة بالصفوف وطريقة بالأعمدة



زي هنا ممكن ابغاك تجيب العنصر الي فالاندكس رقم (j,i) وهذا على حسب ممكن ابدا امشي بالصفوف
اول حبه حبه زي الي محدد باللون الاحمر او امشي بالأعمدة زي الي محدد بالازرق

بعدها بنتكلم عن الـ Associative arrays

hi_temps = ("Mon" => 77, "Tue" => 79, "Wed" => 65, ...);

وهذا مثال عليها, يصير انت بس تنادي مثلا temps[Mon] وهو راح يرجع لك قيمة 77

يعني هي زي الاراي بس نوعاً ما اسهل حسب استخدامك او حسب نوع برنامجك, انت افهم الفكرة ومامعليك

هذا مثال اخر بلغة PHP

```
<?php
$age=array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
$age1=array("Peter", "Ben", 2, 9.3);
$age["Ali"] = "343";
echo "Ali is " . $age['Ali'] . " years old. " . $age1[1] . " " .
$age1[94];
?>
```

طيب بعدها كان يتكلم عن ال Record وكذا

هي بسيطة اقرها بنفسك وراح تفهمها بأذن الله بس راح اوضح شي

ايش هو الريكورد؟ طيب خلينا نحط مثال... بيانات المواطنين في سيرفرات وزارة الداخلية

تمام؟ كل شخص موجود اسمه هناك هو عباره عن ريكورد، الريكورد هو عباره عن صف فيه مجموعة من البيانات وتكون انواعها مختلفه، مثلا الاسم يكون string رقم الجوال int الجنس char رصيد البنك مثلا ممكن يكون الى اخره، حاليا بعد ما فهمت وش يعني ريكورد، ارجع كمل السلايدات بتحصل كل شي مفهوم لك باذن الله float وهذا مثال بسيط لتتضاح الصورة

```
type Emp_Rec_Type is record
    First: String (1..20);
    Mid: String (1..10); ] string
    Last: String (1..20); ]
    Hourly_Rate: Float;   float
end record;
```

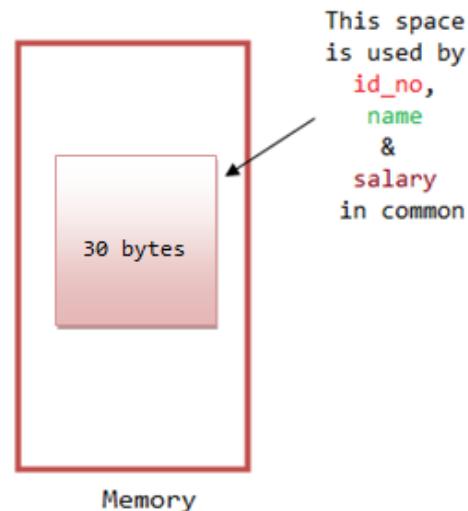
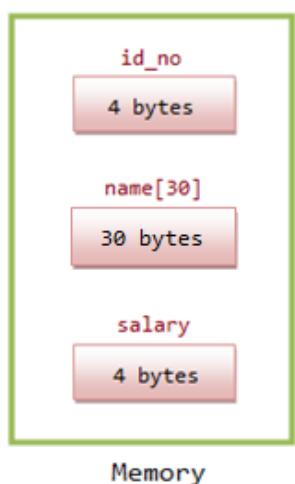
بعدها بدا يتكلم عن ال Unions Type

تكلمت عن ال union في الملخص الي فات بس برجع اذنكه بأختصار

لمن يكون عندك او بجيكت ويكون داخله مجموعة متغيرات

```
struct Employee
{
    int id_no;
    char name[30];
    float salary;
};
```

```
union Manager
{
    int id_no;
    char name[30];
    float salary;
};
```



اضافة بسيطه كمان:

فيه لغات تدعم ال Union مع range checking هذي تسمى Discriminant

و فيه لغات تدعم ال Union بدون range checking هذي تسمى free union

افهم الرسمه الي فوق والمفهوم بشكل كامل, برضو كالعاده كم سلайд حط فيها اكوا德 للغات زي ada و C بس مر عليها وخذ نظرة والامور بخير بادن الله

Pointer and Reference Types

- A **pointer** type variable has a range of values that consists of **memory addresses** and a special value, **nil** (**nil = null**)
- Provide the power of indirect addressing
- Provide a way to manage dynamic memory
- A pointer can be used to access a location in the area where storage is dynamically created (usually called a **heap**)

هذا السلايد يقولك عندنا شي اسمه **pointer**

ال**pointer** هو الشي الي يأشر على ال **memory** وال**range** حقه هي **قيم الميموري**

حسب علمي انها تبدا ب **00000000** وتنتهي **ffffffffff** ومعاهم قيمة ال **null** لو كان فاضي

وعنده خصائص زي الوصول المباشر للميموري **direct access**

وبرضو يقدر يوصل للأماكن الي فيها التخزين **dinamically** ويسمو المكان هذا **heap**

Pointer Operations

- Two fundamental operations: assignment and dereferencing
- Assignment is used to set a pointer variable's value to some useful address
- Dereferencing yields the value stored at the location represented by the pointer's value
 - Dereferencing can be explicit or implicit
 - C++ uses an explicit operation via *
 - $j = *ptr$
 sets j to the value located at ptr

يقولك عندنا عمليتين تحصل لك Pointers وهم: dereferencing و assignments

عملية الـ assignments انك تحدد او تخزن قيمة address معين في متغير من نوع pointer

```
int *pointer = &variable;
```

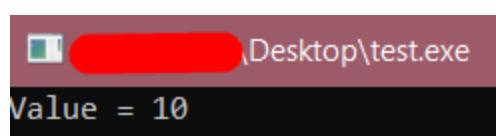
مثل هذا السطر ينكتب في لغة C (مو موجود بالسلайдات بس حطيته لكم عشان تفهمون الفكرة)

راح يرجع قيمة الـ address حق المتغير الي اسمه pointer object ويخرنه في variable

```
#include <iostream>
using namespace std;

int main() {
    int kau = 10;
    int *pointer = &kau;
    cout << "Value = " << *pointer;
}
```

ال kod الي فوق هذا برنامج C++ بسيط يرجع لك الادرس حق المتغير kau ويطبع القيمة وهذا الـ output



هذا العملية يسموها assignment لأنك جبت قيمة الادرس حق kau وسويت لها assign في متغير من نوع pointer وبعدها من خلال الـ pointer تطبع القيمة (تعامل بالـ address مو بالمتغير نفسه)

لو تلاحظ السطر حق الطباعه طبعت القيمة الي يأشر عليها الـ pointer ماطبعت باستخدام المتغير

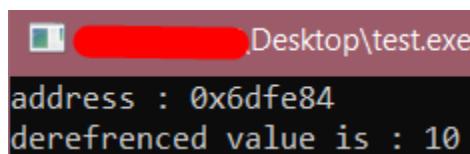
```

#include <iostream>
using namespace std;

int main() {
int id = 10;
int * id_pointer = &id;
int dereferencing = *id_pointer;
cout << "address : " << id_pointer << endl;
cout << "dereferenced value is : " << dereferencing << endl;
return 0;
}

```

برضو هذا برنامج ثاني، خلونا نشغله ونشوف الـ output بعدين نشرح وش صار عشان نفهم بأذن الله



طيب هنا ايش صار، بالبدايه عرفت متغير من نوع int وعطيته قيمة 10

بعدها خزن الـ address حق المتغير id pointer داخل object اسمه id_pointer

بعده جبت القيمة الي يأشر لها الـ pointer الي اسمه id_pointer وخرزتها في متغير اسمه dereferencing

طيب ايش صار هنا ؟

لاحظ انه السطرين الاخيره لمن طبعت ما استخدمت المتغير الي اسمه id بس في الـ output راح تلاحظ الـ 10

مطبوعه ... كف كذا ؟

الي صار انی خزنت قيمة الميموري الي هي 0x6dfe84 في متغير اسمه id_pointer

بعدها قلت جيب القيمة الي موجودة في الـ address رقم 0x6dfe84 وحطها في متغير اسمه dereferenced

وبعدين اطبع القيمة ذي باستخدام المتغير dereferenced

هذي العملية كاملة اسمها dereferencing

الشرح الي فوق موجود بالسلайдات بس مكتوب كلام سطرين وماتم توضيحه، ما أدرني هل مطلوب منكم بس تعاريف او ممكن الدكتور يجيب لنا مثال كود، لو جاب كود محد راح يعرف لانه ماتم شرحه بس هنا الحمد لله تم شرح المفهوم بالأمثله والنتائج.

بالنهائيه يمكن انی زودت شوية كلام بس حبيت افيديكم سواء استفدت في المادة ذي او بعدين لاننا بندرسها في المواد المتقدمة او حتى تفيديك بعدين لحياتك العملية بأذن الله

Problems with Pointers

- Dangling pointers (dangerous)
 - A pointer points to a heap-dynamic variable that has been de-allocated
- Lost heap-dynamic variable
 - An allocated heap-dynamic variable that is no longer accessible to the user program (often called *garbage*)

مشاكل ممکن تواجهك وانت تستخدمو الـ Pointers

Dangling Pointers

لمن يكون عندك pointer يأشر على variable هذا ينحذف, فيصير الـ variable يساوي null

Lost heap-dynamic variable

طيب لازم اوريكم كود مكتوب بعدها نشرح لأن فكرتها شوي تلخبط

```
#include <iostream>
using namespace std;

int main(){
int a = 10;
int b = 20;

int *pointer1 = &a;
int *pointer2 = &b;
cout << "Pointer 1 point to value : " << *pointer1 << endl;
cout << "Pointer 2 point to value : " << *pointer2 << endl;

*pointer2 = *pointer1; ←

cout << "Pointer 1 point to value : " << *pointer1 << endl;
cout << "Pointer 2 point to value : " << *pointer2 << endl;
}
```

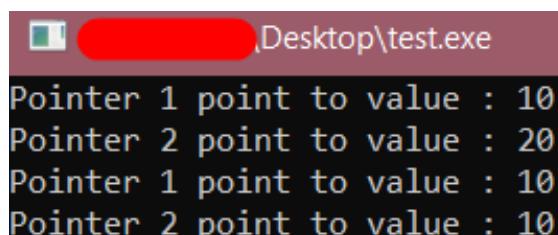
طيب هذا الكود بشرحه شوي, عندي متغيرين: a=10, b=20

وعندى اثنين pointers كل واحد فيهما يأشر على متغير pointer2=b و pointer1=a

عند السهم الأخضر, خليت pointer2 يساوي قيمة pointer1 يعني كل الـ pointers صاروا يأشروا على a

حاليا المتغير b مانقدر نوصل له لأنه مانقدر نوصله, لأن كل الـ pointers يأشرون على مكان ثانى

وهذا الـ output حق الكود الي فوق عشان تعزز فهمك



```
Desktop\test.exe
Pointer 1 point to value : 10
Pointer 2 point to value : 20
Pointer 1 point to value : 10
Pointer 2 point to value : 10
```

طيب بداية الشابتر ذا عندنا الـ Arithmetic Expressions ومنها 3 انواع

unary = one operand, binary = tow operands, ternary = three operands

الـ operands يقصد فيها الحروف والمتغيرات

بعدها عندنا الـ operators وهي الاشارات الرياضية والحسابية، ولهم ترتيب وأولوية

العملية	الأولوية
الاقواس ()	1
الـ unary operators الى هي ~,!,-,++,+,--	2
** الأسس	3
/,* الضرب والقسمة	4
-,+ الجمع والطرح	5

Exceptions in Precedence Rules

- Most of programming languages give module operator “%” high precedence over unary operator “_”
- That is to say $-a\%b$ is equivalent to $-(a\%b)$
 - $-8\%3 \rightarrow -(8\%3) \rightarrow -2$
- Perl gives unary operator “_” a higher precedence
- $X \bmod Y = X - \lfloor x/y \rfloor * Y$ 

```
#!/usr/bin/perl
$a= -5 % 3;
print "Value of a is $a \n";
```

هنا يقولك ان لغة Perl تدي الـ أولوية اعلى من الـ %، فراح تفرق النتيجة عندك لمن تحسبها زي الكومبایلر حق الـ Perl او كمبایلر للغة ثانية، المهم واعطاك معادلة الي محددة بسهم أخضر، بس ما علينا

بالأختبار نستخدم online compiler احسن عشان نضمن الدرجة، في حال انك تبغاء تطبق يدوي طبق على المعادلة الي فوق عشان تقدر تحسب باقي القسمه في لغة Perl

Infix, Prefix, Postfix بعدها شرح الـ

In unary operator:

- Prefix: $op\ a$

في حال انه unary وكان Prefix تكون الاشارة قبل, مثل: $=a$ او $+a$ اي اشاره تكون قبل الحرف

ملاحظة: $op = \text{operator}$ يعني اي اشاره رياضية

In binary operator:

- Infix: $a\ op\ b$
- Prefix: $op\ a\ b$ or $op(a, b)$ or $(op\ a\ b)$
- Postfix: $a\ b\ op$

الـ infix: يكون الـ op في النص بين الحرفين

الـ prefix: يكون بالبداية قبل الحرفين

الـ postfix: يكون بالاخير بعد الحرفين

طبعا احنى متعددين لمن نكتب حنكتب كذا $4 * (3 + 2)$ هذي تكون infix

بس لو نبغا نحولها لـ prefix حصیر كذا $(+3\ 2)\ 4$

In ternary operator

- C-based languages (e.g., C, C++ and Java)

- An example:

```
average = (count == 0) ? 0 : sum / count
```

- Evaluates as if written like

```
if (count == 0) average = 0  
else average = sum / count
```

- Python

```
x = true_value if condition else false_value  
min= a if a<b else b
```

هذی مجرد امثلة على الـ Ternary operators, خذ نظرة وبس

طيب بعدها عندنا حاجة اسمها Operator associativity rule

طيب نشرحها سررريع, لمن يكون عندك المعادله التالية على سبيل المثال: $2 - 3 * 4 + 5$

بالبداية بتضرب ال 3 في 4 وبيصير 12+5-2, وبعدين بتطرح يصير 10+5- وبعدين يصير الجواب 5

طيب هذی الرياضيات الطبيعية وهو اتنا نعطي الأولوية لل Ashtonat حسب معرفتكم السابقة

في لغة APL لا ماعندها ذا الكلام, الـ APL اولويتها من اليمين لليسار يعني تجمع 4+5 وبعدين المعادله تصير 9-3*2

وبعدها تصير 27-2 ويطلع الجواب 25

عندنا حاجة اسمها Potentials for side effects

طيب نشرحها بشكل مفصل وحلو، هي المشكلة هي تشير لمن تستخدم متغير ما يكون local موجود في main طيب خلونا نشوف الكود عشان نفهم صح

```
#include <stdio.h>
int a = 10;
int fun(){
    a= 15;
    return 30;
}
int main(void) {
    // your code goes here
    int x = a + 10 + fun();
    a= 10;
    int y = fun() + 10 + a;
    printf(" x = %d \n y = %d
\n",x,y);
    return 0;
}
```

Output:
x = 50
y = 55

طيب خلونا نسوي trace للكود

اول شي حيخش المين وبيعمل متغير اسمه x وبيكون قيمته عباره عن جمع قيمة a + 10 + fun() مع 10 صارو 40 طيب اول شي خش الدالة fun() وكانت قيمتها 30 لأنها تسوی 30 return بعدها جمع الـ 30 مع 10 صارو 50 بعدين جا بيجمع قيمة الـ a، بس ما عنده في الـ main قيمة الـ a، فراح يضطر يستخدم الـ global الي قيمتها 10 وراح يجمع الـ 40 مع الـ 10 ويدينا الناتج 50

طيب هنا راح اشرح حاجة مو مطلوبه في المنهج بس راح تفهمك ليش استخدم الـ a الي هي global بدلالي موجودة في الدالة fun (للفائدة اللي بيغا يفهم لو ماتبعا عادي سوي سكيب للمربع الأزرق ذا وروح كمل بعده.

الي صار هنا أول شي بيغا يجمع قيمة الـ a مع 10، طيب أول حاجة الـ a مالها قيمة واصلا ماتمتعريفها قبل كذا في main

طيب هنا زي مانعرف الـ scoping حق الـ functions نقدر نستخدم متغيراتها أو متغيرات ابوها الي هي الي فوقها، في هذه الحاله ما عندنا a فراح نطلع للسكوب حق الأب، الي هو الـ global وراح يحصل الـ a الي قيمتها 10 وراح يستخدمها بعدها راح يجمع الـ 10 مع الـ 10 الثانية ويصير 20، راح يجمع الـ 20 الحين مع قيمة fun، راح نلاحظ أن فيها fun بس على حسب مادرتنا انه لمن نخش للـ fun راح نسوى قيمة جديدة للـ a والي هي 15 ولمن نخلص من الدالة راح نسوى deallocation للمتغير فراح ينحذف ونرجع للمين نجمع الـ return الي هو 30 مع الـ 20 ونطبع الناتج الي هو 50 (حرفيًا الـ a الي جوا الـ fun مرا مامنها فايدة في الحالة ذي)

نرجع نكمل الكود, بعدها موجود عندنا $a=10$, هنا تم تعریف متغیر في الـ stack اسمه a وقيمة 10

طیب نكمل راح يسوی متغیر اسمه a وراح يحط فيه قيمة $fun + 10 + a$

أول شي راح يخشن الـ fun وأول شي حيصادفه هو انه راح يسوی لقیمة الـ a وراح تصیر 15, وبعدها حيسوی return بقیمة 30 بس لاحظ هنا خط ملیون خط, بحكم أن الـ a الي في الـ stack هي حق الـ main فماراح يصیر لها deallocation لمن ينتهي من الـ fun, بس اللهم انه حيصیر لها overwrite بالقیمة الجديدة الي هي 15 بعدها راح يجمع قیمة الـ 30 حق الـ 10 مع الـ 40 ويدینا 40, وراح يجمع الـ 40 مع الـ a يصیر 55=15+40

لازم نفرق بين الحالتين وننتبه, هنا بيان مین المبرمج الحقيقی ومین الفاهم في التفاصیل الصغیره التافھة ذی

بعدها فيه سلايدین يتکلم عن الـ Overloaded Operators

بالعربی هي لمن يكون عندك operator (اشارات ریاضیة زي + - وكذا), ويكون لها اکثر من استخدام يعني مثلا الـ + تستخدم لجمع الارقام, ولدمج النصوص وكمان تسوی دمج للـ arrays, هذا یسمونه operator overloading, عشان حل المشكلة ذی لازم نسوی لكل عملية متغیر خاص فيها, زي في البايثون فيه فرق بين علامات القسمة: / تقسم وترجع لك الباقي, بس // تقسم بدون باقي

```
>>> 7/2  
3.5  
>>> 7//2  
3
```

بعدها تکلم عن الـ type conversion ويشمل حاجتين

الأول: casting او برضو یسموه narrowing

هو التحويل من نوع متغیر كبير الى صغير, زي من int الى float, ولازم المبرمج یكتب هذا الشی لأنها ماتصیر

تلقائیاً من اللغة او الكومبایلر (Explicit type conversion)

```
double d = 10;  
int i;  
i = (int) d
```



الثانی: widening او برضو یسموه coercion

هو التحويل من نوع صغير الى كبير, زي لمن تخزن int داخل متغیر float وهو لازم یکتبه المبرمج

وهذا الشی یصیر تلقائی من الكومبایلر (Implicit type conversion)

بعدها فيه كم سلايد يتكلم عن بعض الاشارات وأشياء زي الـ or, and وأشياء سهلة واحتاج شرح, فقط قراءة

طيب نجي لحاجة بسيطه يقولك الـ JavaScript والـ PHP عندهم اشارة وهي ==, وش الفرق بينها وبين == يساوي الـ value بدون ماطالع على الـ type يعني 9 نوعها int تساوي 9 نوعها string بس الـ == تساوي الـ value مع مساواة الـ type برضو, يعني 9 نوعها int ماتساوي الـ 9 نوعها string

عندنا حاجة اسمها Short circuit evaluation

وهي على سبيل المثال عندي الـ if statement هذي if (a > b) || (b++ / 3)

طيب وش يصير؟ يقولك عشان اختصار الوقت نشوف القوس الأول الي هو (a>b), هل هو true ؟ اذا كان ماراح يشيك على القوس الثاني عشان توفير الوقت, لأن بالأساس الـ or لو وحده منهم true خلاص كلهم يصيرو فماراح يفرق الثاني true ولا false, لكن لو الاول كان false راح يشيك على الثاني

السلайдات الجاية كلها كلام فاضي وأشياء نصرية درسناها في برمجة 1, اقرأها بس وتفهمها باذن الله

طيب اول شي في هذا الشابتر هو **Control Structure** وتنقسم لثلاث اجزاء

الجزء الأول:

Tow-way selection statements

الكتاب واضح من عنوانه يعني عندك **selection** بطريقين مثل if....else او للـ if يايروح للـ else ماعنده غيرهم
حط ببالك ممكن يكون عندك **nested selection** يعني if جوتها if و else ثانية, مثل:

```
if(...){  
    if(...){  
  
    } else{  
  
    }  
} else {  
  
}
```

Multiple-way selection statement

زي الـ **switch** او الـ **if** الي تكون كثيره مثل:

```
if(...){  
  
} else if(...){  
  
} else if(...){  
  
} else if(...){  
  
}
```

الجزء الثاني:

Counter-controlled loops

بساطة هي الـ **loop** الي تحتوي على **counter** وشرط للنهاية وقيمة الزيادة, مثل:

```
for (int i = 0 ; i < array.length ; i++)
```

زي الكود الي فوق فيه **counter** يبدا من 0 وفيه شرط النهاية الي هو طول الـ **array** والزيادة الي هي بـ 1 (i++)

Logically-controlled loops

طيب هذى اقرب مثال عليها هي الـ **loop** الي ماتقدر تحدد كم مرا ممكن تمشي الـ **loop**, يعني ممكن تسوى **while loop** شرطها انو المستخدم مثلا مايدخل عدد سالب, انت ماتدرى متى المستخدم يدخل سالب فهذا يعتبر شي مربوط باللوجيك وشي ماتعرف عدد تكراره ومربوط دائم بشرط يا **true** او **false**

وهذى منها نوعين:

الي يكون فيها الشرط حق الـ loop بعد الـ block مثل: do loop body while (ctrl_expr T)	Post-test
الي يكون فيها الشرط حق الـ loop قبل الـ block مثل: while (ctrl_expr T) loop body	Pre-test

الباقي كله سلайдات كان يتكلم عن الـ **Break** والـ **loops** بشكل عام واشياء اساسيات برمجة

الباقي كله كلام ينفهم بمجرد النظر

ذا الشابتر قصير, ويتكلم عن الـ **subprogram**

طيب وش يقصد؟ يعني الـ **methods**

طيب جالس يقولك شوية اساسيات زي:

Basic Definitions

1. A **subprogram definition** describes the **interface** to and the **actions** of the subprogram abstraction
2. A **subprogram call** is an **explicit request** that the subprogram be executed
3. A **subprogram header** is the first part of the definition, including the name, the kind of subprogram, and the formal parameters
4. The **parameter profile (signature)** of a subprogram is the **number**, **order**, and **types** of its parameters

1- يقولك شكل الكود او طريقة الكتابه من خلال النظر لها تقدر تحدد عمل البرنامج وأيش يسوي, يعني مثلا لو شفت **loop** فيها **method swap** انت كذا بتفهم ان هذي الـ **method** الفايدة منها **sorting**

2- لمن تنادي الـ **method main** من الـ **method** لازم تكتب اسمها عشان تناديها, تسوي **invoking**

3- يقولك الـ **header** حق الـ **method** هو عباره عن اسمها ونوعها (يقصد وش ترجع **(void, int, string)** وكمان المتغيرات الي تاخذها

4- يقولك الـ **parameter profile** او يسموه **signature** يعني التوقيع, هوا عدد متغيراتها وترتيبهم وانواعهم
ليش سموه توقيع ؟ لأن من خلله تقدر تفرق بين الـ **methods** لو كان عندك تشابه فالأسماء, نفترض عندنا **2 methods** فيه

Student(string name, int id, float gpa)

Student (int id, string name, float gpa)

طيب حاليا الأسم واحد, بس على حسب ترتيبك للمتغيرات وقت الاستدعاء راح تختار وحدة منهم

لو حطيت اول شي الاسم بعدين الرقم الجامعي بعدين المعدل كذا تنادي الأولى

لو حطيت اول شي الرقم الجامعي بعدين الأسم بعدين المعدل كذا تنادي الثانية

عشان كذا اسمه توقيع لأن من خلله تقدر تفرق بين الـ **methods** ويخلطي كل وحدة غير عن الثانية

Actual/Formal Parameter Correspondence – Positional

1. The binding of actual parameters to formal parameters is by position: the first actual parameter is bound to the first formal parameter and so forth
2. Safe and effective
3. When lists are long, it is easy for a programmer to make mistakes in the order of actual parameters in the list
4. Most of programming languages

تعرفون وش مكتوب هنا؟ بشرحها على السريع...

1- يقولك لمن تمرر المتغيرات حقك للـ method راح تتمرر بالترتيب, مثل:

```
main(){  
    sum(int1,int2,int3)  
}      ↗   ↗   ↗  
sum(num1,num2,num3){  
}
```

2- آمنه وفعاله, يقصد افضل من انك تحط الكود حقك كله في الـ main, يصير تقسيمه اجزاء داخل methods

3- لمن يكون عندك متغيرات الـ method كثيره, ممكن يغلط فيها المبرمج, مثل:

student(name, id, gpa, courses, projects,,)

4- اغلب لغات البرمجة تدعم الـ methods ويمديك تسويها وتستخدمها

شافين الكلام التافه الي فوق؟ اغلب الشابتر ذا كلام زي كذا, اشياء ماتحتاج شرح مجرد قراءة تكفيك بأذن الله, لو كان عندك سؤال في جزئية معينة تقدر تسأل زملائك بأذن الله ما يقسووا او توافقوا او يقروا او تواصل معاي والله يعين ويسهل للجميع

آخر شي والله الحمد, يتكلم عن الـ Procedures والـ Function

الـ **Function**: عباره عن مجموعة اوامر, تأخذ **input** واحد او أكثر, وترجع قيمة

مثل: لو سوينا **method** عشان تحسب العمر او تحسب مساحة مربع او تحسب قيمة لأي شي

أسم **function** اخذته من الرياضيات بحكم انها ت Sovi عمليات حسابية داخلها

الـ **Procedure**: مجموعة من الأوامر تنفيذية, مثل: لو سوينا **method** لطباعه معلومات فقط

يتشبهون بشكل كبير بـ **functions** بينهم في جزئية الـ **return** هل هي ترجع قيم ولا لا؟

وشكرا لكم ☺

الله يوفقكم ويسعدكم, ولا تنسونا من دعائكم