

بسم الله الرحمن الرحيم

CPCS 301 – Programming languages  
2<sup>nd</sup> semester 2020  
Summary for the final exam

**\*هذا التلخيص هو جهد احد الطلاب ف راح اوضح نقطتين\***

**أولاً:** هذا التلخيص لا يُغني عن السلايدات او الكتاب بالعكس انا هنا ملخص تقريبا 90 بالمية من الافكار وشارحتها بالتفصيل, الافضل بعد ماتخلص قراءة الملخص تروح تمر على السلايدات وبإذن الله تكون لميت المنهج كله

**ثانياً:** انا طالب مثلي مثلك وهذا مجرد اجتهاداً مني, وهذا مايعني انه كل الكلام الي شرحته صح ولاكن انا بحثت ودورت ونقحت المعلومات وكتبت الاصح ولاكن ان كان فيه خطأ او شي كلموني ع الواتس او اي وسيلة تواصل اخرى وراح اعدله بإذن 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
هيكلية من البيانات الوصفية التي تصف البيانات يعني مجموعة بيانات مع بعض هدفها انها تصف بيانات اخرى وهي تكون احد الاجزاء في ال <b>simple table</b>	<b>descriptor</b>
الـ <b>object</b> هو عبارة عن data-type زيه زي ال integer, string بس ال object يتم تعريفه من قبل المستخدم ( زي لمن كنا نسوي linked list في برمجة 3 كانت عبارة عن نوع بيانات بس احنى بنفسنا نعرفه ونحدد خواصه وكذا, مايكون جاهز في اللغة نفسها زيه زي ال integer, string	<b>object</b>

## 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 تعريفها او تكوينها بالاعتماد على **data type** اخر

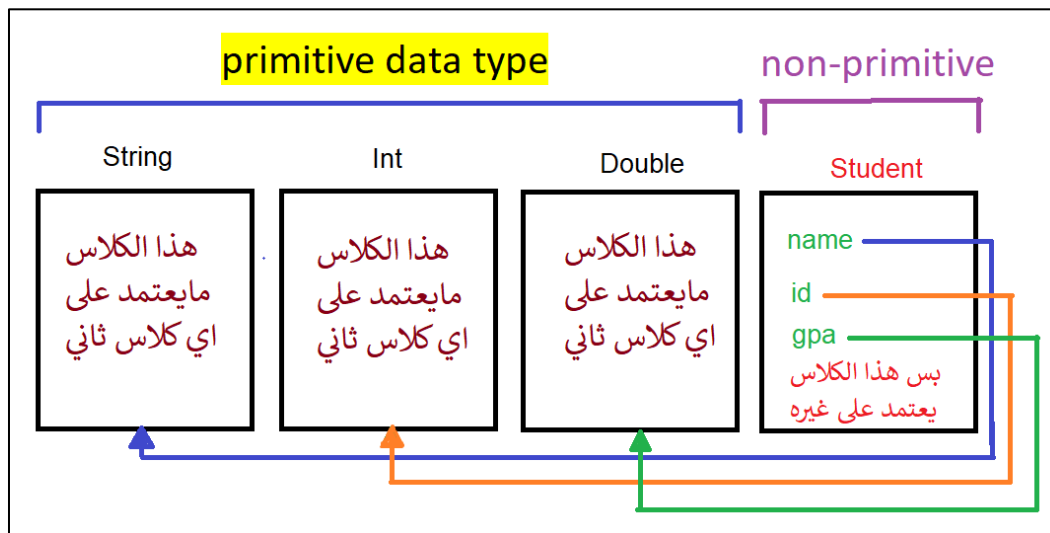
ايش يعني؟... نأخذ مثال مثلا بسوي **data type** من نوع **student** وحطيت فيه هذي ال **data members**

string	name
int	id
double	gpa

هنا يقولك ان هذا ال **student** مايعتبر **primitive data type** ليش؟؟ لانه اصلا مكون من **data types** ثانية

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

هذي صورته 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** وحيعطيك بعض الأمثلة ويشرحها لك...

## اولاً Integer

### 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 اكثر نوع يمثل الهاردوير, نشرحها شوي

تفتكرو ال **array in C language** ؟ كانت تمثل او تعكس صورته عمل الميموري برضو ال ريجيسترات الي درسناها في بنيان الحاسب (اتوقع الاغلب درس الماده) ال ريجيسترات كانت تتمثل بالارقام وهذا يسهل عملها وتفاعلها مع الهاردوير

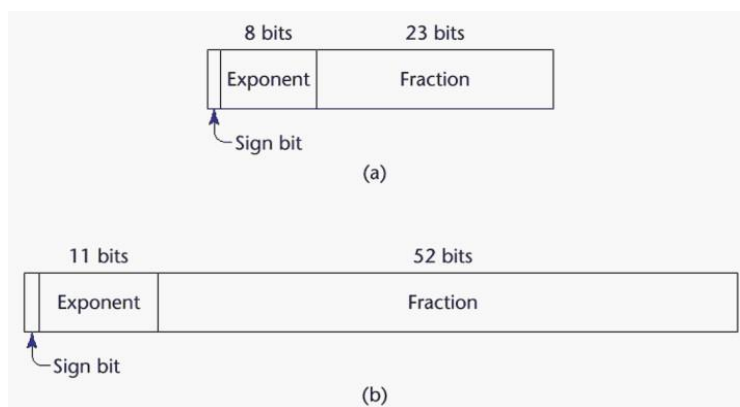
2- ممكن يكون عندك اكثر من نوع **integer** في نفس اللغة (اقرأ النقطة رقم 3 بتفهم اكثر)

3- يقولك انواع ال **integers** الي ف لغة **java** هي : **byte**, **short**, **int**, **long** فالجافا عندها 4 انواع **integers** وممكن لغات ثانيه عندها اقل او اكثر من 4

## ثانياً Floating point

### 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 الحقيقية بها (الأعداد الحقيقية يقصد فيها اي شي فيه فاصلة)

2- اللغات العلمية او المستخدمة في ال **science** على الأقل تستخدم نوعين وهم **float**, **double** وممكن اكثر

3- يقولك انها تشبه ال hardware بس مو دائماً

4- يقولك ان الي وضع هذي الرسمة الي ع اليمين والي حط المعايير هذي وقسمها هم منظمة **IEEE**

يعني بالعربي منظمة IEEE هم الي حطو كيف يتم تخزين ال floating point في الحاسب 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 ومجالات الاعمال والأموال, زي لغتين هم C#, COBOL

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 دايم الفلوس لمن تنكتب ينكتب بس رقمين بعد الفاصله زي 5.25 او 17.29 دايم الهلات تجي رقمين بس بعد الفاصله ماتجي 3 او 4 هذي تعتبر fixed لان احنى عارفين عدد الخانات الي عندنا ( ترا ممكن تكون fixed قبل او بعد الفاصله عادي ف اي مكان بس هذا مجرد مثال )	Fixed number
هذي الاعداد المتغيره الي ماتدري وش ممكن تخزن فيها زي 5246.2324458 او 21.1 اي شي يمشي	Decimal digits

فهمتوها ؟ اذا لا بفصلها لك زياده, الحين قل نفترض قلت لك كم معدلك ؟ فلنفترض انت قلت لي 4.3 او 4.33 بقولك اوه حلو ما شاء الله, طيب فلنفترض انك قلت لي 4.333 راح اقولك هاه ؟ راح استنكر الاجابة شوي

لأني انا حاط بمخي انه دايم المعدل يكون رقم واحد يسار الفاصله وهو يكون 1 او 2 او 3 او 4 او 5 ويمين الفاصله دايم يكون فيه رقم او رقمين تبدا من 00 الى 99 هذا مثال على Fixed number لآكن لو مثلاً كنت تقرا كتاب فيزياء وجو قالولك ثابت سرعه الضوء هو 299792458.0 هنا مراح تقدر تتوقع كم ممكن يكون طول الرقم, او كم ممكن يممين 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

## رابعاً Boolean

### 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

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

ميزتها : readability

## خامساً Character

### 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 ال ASCII او ال Unicode )
  - 2- اكثر نظام رقمي يتم استخدامه هو ال ASCII
  - 3- وفيه نظام ثاني اسمه Unicode ويستعمل نظام تخزين 16 بت بالنسبة لل Unicode فيه اغلب الحروف والرموز الي موجودة في لغات البشر يستخدم في ال Java, C#, JavaScript
- عشان تفهم اكثر شوف في النت جدول عن ال Unicode, ASCII عشان تعرف وش نقصد بكلمة (بيانات رقمية)... راح تلاحظ انهم يتخزنو 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 الstring عبارة 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 للـ Design issues

1- يقولك هل هي عبارة عن primitive data type ؟ ام انها عبارة عن array of characters ؟  
طيب هنا يقولك هل هي اصلا primitive data type يعني هل هي اصلا موجوده في اللغه بشكل اساسي زي الجافا والبايثون ؟ ام انها حاله مخصصه من ال array of characters زي لغة C  
الي قد استخدم لغة C قبل كذا, راح تلاحظ انك لمن تبغا تعرف string راح تسوي array من نوع characters وجوتها تخزن الstring حقا

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

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 ماقرئت الملخص الي فات راح احط لك الرابط هناك مشروح فيه كل شي

[https://drive.google.com/drive/folders/1uaCL57\\_qCEnv2a\\_cTuq8-rFO7N0Pj91S?usp=sharing](https://drive.google.com/drive/folders/1uaCL57_qCEnv2a_cTuq8-rFO7N0Pj91S?usp=sharing)

وكم ان 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 تستخدم الموقع الثاني تقدر تحط فيه اي regex ويطلع لك الجواب (برضو حطيتاه 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

### 1- لغة C, C++

يقولك انها not primitive يعني ماهي موجودة في اللغة بشكل اساسي (شرحتها فوق)

وكم ان يتم استخدام array of characters عشان نستخدم ال string (برضو شرحتها فوق)

### 2- لغة Snobol4

تعتبر primitive يعني مبنية وموجوده باللغة بشكل اساسي, وتدعم functions كثيره تتعامل مع

السترينق زي length وباقي الدوال الي ممكن تطبقها على اي متغير من نوع String

### 3- لغة Java

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 نرتبها فجدول عشان يطلع شكلها كيوت شوي

اللغات الي تستخدم	شرح	نوع الString
Cobol, Java	لمن تعرف متغير وسميته Abdullah وبعدين تغير قيمته وتخليه Ahmed البوينتر راح يأشر على object 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 "Ahmed" = name الي حيصير بالزبط انه راح نسوي object جديد ونخزن جوته قيمه وهي ahmed والبوينتر الي كان اسمه name راح يأشر على الobject الجديد حقنا الي جوته قيمة ahmed وراح نطنش الobject القديم	Static
C, C++	هنا ماراح نحتاج نسوي object جديد زي ماسوينا فوق في ال java, هنا راح نخزنهم في نفس ال object وراح نغير الطول لأنه dynamic, يعني لو خزنا Abdullah راح يكون اوبجكت وطوله 8, ولو بعدين سوينا "ahmed" = name راح نخزن قيمه ahmed في نفس الobject القديم بس راح نغير طوله ويصير 5 بس حط فبالك ان الطول له limit معين, يعني ممكن يكون طوله اقصى شي 100 حرف او 150 حرف ماتقدر تزود اكثر من كذا	Dynamic (limited)
Snobol4, Perl, JavaScript	زي الي فوق بس هنا ماله نهاية تقدر تسوي طوله قد ماتبغا	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

هنا السترينق حقا

طوله

عنوانه في الميموري

Static string
Length
Address

Compile-time  
descriptor for  
static strings

Limited dynamic string

Maximum length

Current length

Address

Run-time  
descriptor for  
limited dynamic  
strings

هنا السترينق حقا

اقصى طول ممكن

طوله الحالي

عنوانه في الميموري

في هذا السلايد بس يوريك شكل ال simple table كيف بيكون شكلها في حال لو كان static او كان limited dynamic , بس افهمها وخذ نظرة عليها

## 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 بالمختصر المفيد يقولك عندنا حاجة اسمها **Ordinal Type** وهي انواع بيانات متتابعه, طيب ايش يعني ؟ وأيش استفيد انا ؟

يعني بيانات لو اعطيتك هي راح تقدر تجيب 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 سألتك وقلت لك وش القيمة الي بعده ؟ راح تقولي على طول 5, ايش طيب ؟ لانها تعتبر متسلسله وورا بعض لمن اديك عنصر منه راح تعرف تجيب الي قبله او الي بعده, هم لهم تسلسل وزى كذا

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

### المثال الثاني على non-Ordinal data type

مثلا قلت لك عندي متغير اسمه aaa ونوعه float وقيمته هي 2.1 وقلت لك جيب الي بعده.....  
ماحتقر تجاوب, لأنها تحتل اكثر من اجابة ممكنه 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 بالمختصر هي عبارة عن set فيها elements والسلام عليكم  
زي ما انتو شايقين فوق عندنا set اسمها days وفيها elements وهي mon,tue.....sun  
وبرضو يمدينا ناخذ منها subrange ايش يعني طيب ؟  
تفتكرو الداله الي كان اسمها substring ؟ ماتفتكرو صح ؟ طيب نشرحها

هذا كود جافا

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

وهذا الأوت بوت حقها

bdu

مايحتاج شرح واضحة

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

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

وهذا مثال اخذ ايام الاسبوع كـ subrange من ال Days

## 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 الرنتايم ويكون يا هيب او ستاك
- What is the maximum number of subscripts  
maxint اخر اندكس
- 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

يقولك في Fortran, PLI, Ada يستخدمو الـ () مثال: array(10)

وفي غالبية اللغات الاخرة يستخدمو الـ [] مثال: array[10]

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 ال types حق ال index الي ممكن نستخدمها...  
يعني مثلا سويننا array[i] هنا يقولك ال i ايش ممكن يكون نوعها هل int, string,....  
طيب الي محدده بالأخضر مشروحه وواضحة

## راح نتكلم شوي عن الي محدد بالسماوي

يقولك ال C, C++, Perl, Fortran ماتسوي range checking طيب وش يعني ؟  
مثلا عندي اراي في لغة C وطولها 10 وجيت انت وطبعت التالي array[100]  
هنا ماراح يدريك ايروور وراح يرجع لك قيمه عشوائية من الميموري, لأنه ما يتأكد من ال range  
وال Java, ML, C# لو سويت نفس الي فوق وطبعت الأندكس رقم 100 راح يدريك ايروور, لانها تتحقق  
هل الأندكس الي انت تبغاه جوا الرينج الي بين 0 و 10 او لا

## طبيب بعدها يتكلم عن الـ Categories للأندكس

بنحطها ف جدول عشان تزيبط امورنا

النوع	وصف	مثال
<i>Stack-dynamic</i>	حجم الاري ومكانها في الميموري كلهم dynamically bounded وتصير وقت ال run-time	In Ada, you can use stack-dynamic arrays as  Get(List_Len); declare List: array (1..List_Len) of Integer begin ... end;
<i>Fixed heap-dynamic</i>	حجم الاري 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
<i>Heap-dynamic</i>	هنا يقولك حجم الاري ومكانها في الميموري متغيرين ميزتها: مرنة من ناحيه انك تكبر الاري وقت ماتبغا وتصغرها وقت ماتبغا فما تستهلك الاري 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 مايركز على الاشياء النظرية ذي, حتى لو مافهمت شوي

سكيب مو مشكلة

وهنا بحط لكم مرجعين لو تبغا تقرا زياده وتستفيد

[http://comp.eng.ankara.edu.tr/files/2015/09/subscript\\_binding\\_multid\\_arrays.pdf](http://comp.eng.ankara.edu.tr/files/2015/09/subscript_binding_multid_arrays.pdf)

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

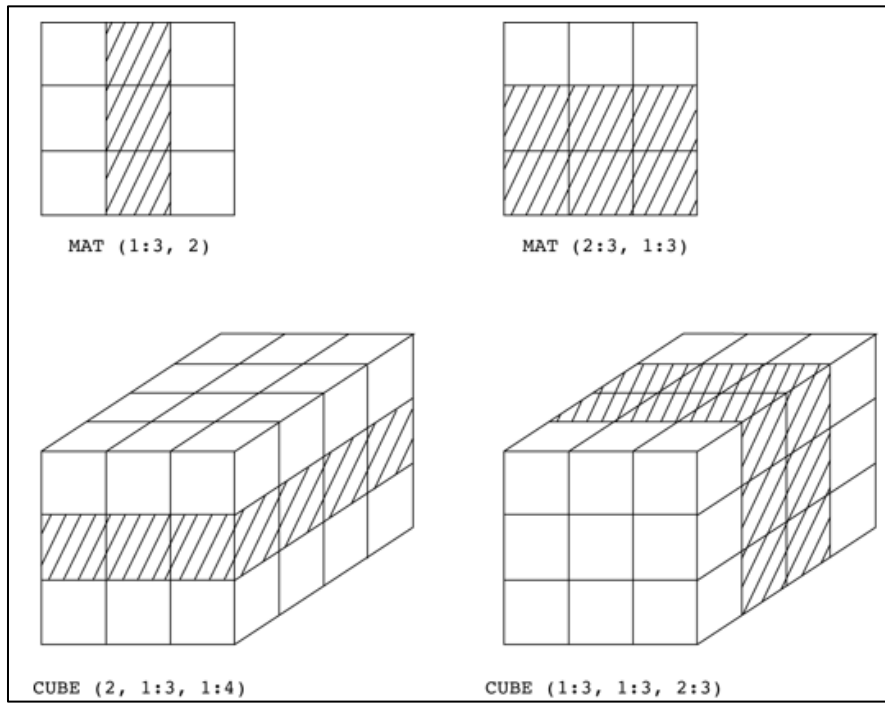
بعدها فيه كم سلايد يتكلم عن كيفيه انشاء الاري في بعض اللغات, اشياء سهله بس مر عليها قراءة ماتحتاج شرح

بعدها يتكلم عن شيئين

**Rectangle array** وهي الاري المربعة الي عدد صفوفها واعمدتها زي بعض

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

درسناها في برمجة 2 لو فاكرينها



هنا يشرح مفهوم ال **Slices** وهي عبارة عن قطعة او جزئية من اراي ثانيه زي ماهو واضح ف الصورة المربع المحدد بخطوط كثيره هي **slice** ومأخوذة من الاراي الاساسية الي هي بالابيض

## Implementation of Arrays

Access function maps subscript expressions to an address in the array

Access function for single-dimensioned arrays:

$$\text{address}(\text{list}[k]) = \text{address}(\text{list}[\text{lower\_bound}]) + ((k - \text{lower\_bound}) * \text{element\_size})$$

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

يجيب الادرس حق اول المنت في الاراي

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

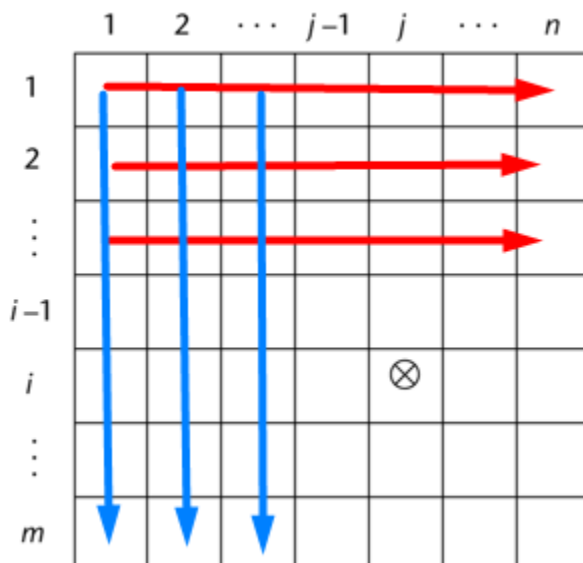
في الصورة ذي يشرح لك كيف تجيب ال address حق اي عنصر بس بأستخدام الاندكس تبعه يعني لو قلت لك جيب ال address حق العنصر رقم 5 في الاراي الي اسمها list راح تسوي زي المثال الي فوق ٨٨٨

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 فالانديكس رقم (i,j) وهذا على حسب ممكن ابدأ **امشي بالصفوف** اول حبه حبه زي الي محدد باللون الاحمر او **امشي بالأعمدة** زي الي محدد بالازرق

### 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 الفكره وما عليك

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 اقراها بنفسك وراح تفهمها بأذن الله بس راح اوضح شي

**ايش هوا الريكورد ؟** طيب خلينا نخط مثال... بيانات المواطنين في سيرفرات وزارة الداخلية

تمام ؟ كل شخص موجود اسمه هناك هوا عبارته عن ريكورد, الريكورد هوا عبارته عن صف فيه مجموعه من البيانات وتكون انواعها مختلفه, مثلا الاسم يكون string رقم الجوال int الجنس char رصيد البنك مثلا ممكن يكون float والى اخره, حاليا بعد مافهمت وش يعني ريكورد, ارجع كمل السلايدات بتحصل كل شي مفهوم لك باذن الله

وهذا مثال بسيط لتتضح الصورة

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

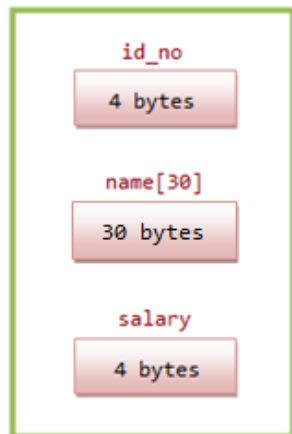
string  
float

بعدها بدا يتكلم عن ال **Unions Type**

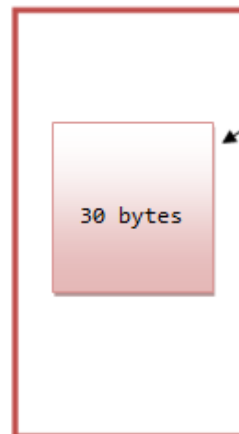
تكلمت عن ال **union** في الملخص الي فات بس برجع اذكره بأختصار

لمن يكون عندك اوبجيكت ويكون داخله مجموعة متغيرات

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



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



This space  
is used by  
id\_no,  
name  
&  
salary  
in common

اضافة بسيطه كمان:

فيه لغات تدعم ال Union مع range checking هذي تُسمى Discriminant

وفيه لغات تدعم ال Union بدون range checking هذي تُسمى free union

افهم الرسمه الي فوق والمفهوم بشكل كامل, برضو كالعاده كم سلايد حظ فيها اكواد للغات زي ada و C بس مر عليها وخذ نظرة والامور بخير باذن 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 يآشر على ال memory وال range حقه هي قيم الميموري حسب علمي انها تبدأ ب 00000000 وتنتهي ب ffffffff ومعاهم قيمة ال null لو كان فاضي

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

وبرضو يقدر يوصل للأماكن الي فيها التخزين **dinamiclly** 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 لـ Pointers وهم: **assignments** و **dereferencing**

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

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

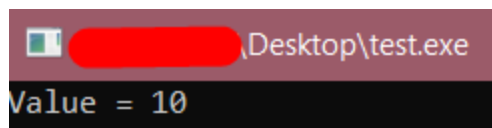
مثل هذا السطر ينكتب في لغة C (مو موجود بالسلايدات بس حظيته لكم عشان تفهمون الفكرة)

راح يرجع قيمه الـ address حق المتغير الي اسمه variable ويخزنه في pointer object الي اسمه pointer

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

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

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

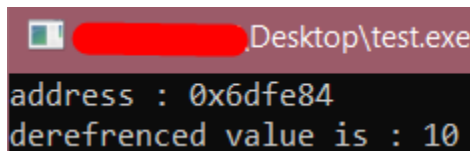


هذي العملية يسموها **assignment** لأنك جبت قيمة الادرس حق kau وسويت لها assign في متغير من نوع pointer وبعدها من خلال الـ pointer تطبع القيمة (تتعامل بالـ address مو بالمتغير نفسه)  
لو تلاحظ السطر حق الطباعة طبعت القيمة الي يأشر عليها الـ pointer ما طبعت بأستخدام المتغير

```
#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 داخل pointer object اسمه id\_pointer (assignment)

بعده جبت القيمة الي يآشر لها الـ pointer الي اسمه id\_pointer وخزنتها في متغير اسمه dereferencing

طيب ايش صار هنا ؟

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

مطبوعه... كيف كذا ؟

الي صار اني 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 موموجود بالسلاميات بس مكتوب كلام سطرين وماتم توضيحه, ما أدري 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.

بالنهايه يمكن اني زودت شوية كلام بس حببت افيدكم سواء استفدت في المادة ذي او بعدين لاننا بندرسها

في المواد المتقدمة او حتى تفيدك بعدين لحياتك العملية باذن 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 ممكن تواجهك وانت تستخدم الـ **Pointers**

## Dangling Pointers

لمن يكون عندك pointer يـأشـر على variable والـ variable هذا يـنـحـذف, فيصير الـ pointer 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

وعندي اثنين **pointers** كل واحد فيهم يـأشـر على متغير **pointer1=a** و **pointer2=b**

عند **السهم الأخضر**, خلّيت **pointer2** يساوي قيمة **pointer1** يعني كل الـ **pointers** صارو يـأشـروا على a

حاليا المتغير b مانقدر نوصل له لأنه مانقدر نوصله, لأن كل الـ **pointers** يـأشـرون على مكان ثاني

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 الشابتز 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

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

## 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
  - $X \text{ mod } Y = X - \lfloor x/y \rfloor * Y$  ←

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

هنا يقولك ان لغة **Perl** تدي ال - أولوية اعلى من ال %, فراح تفرق النتيجة عندك لمن تحسبها زي الكومبايلر حق ال Perl او كمبايلر للغة ثانية, المهم واعطاك معادلة الي محددة **بسهم أخضر**, بس ماعلينا

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

## بعدها شرح الـ Infix, Prefix, Postfix

In unary operator:

- Prefix:  $op\ a$

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

ملاحظة:  $op = \text{operator}$  يعني اي اشارة 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: الـ  $op$  يكون بالبداية قبل الحرفين

الـ postfix: الـ  $op$  يكون بالآخر بعد الحرفين

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

بس لو نبغا نحولها لـ prefix حثصير كذا  $(4 + 3) * 2$

In ternary operator

- C-based languages (e.g., C, C++ and Java)
- An example:  
 $\text{average} = (\text{count} == 0) ? 0 : \text{sum} / \text{count}$
- Evaluates as if written like  
 $\text{if } (\text{count} == 0) \text{ average} = 0$   
 $\text{else average} = \text{sum} / \text{count}$
- Python  
 $x = \text{true\_value if condition else false\_value}$   
 $\text{min} = a \text{ if } a < b \text{ else } b$

هذي مجرد امثلة على الـ Ternary operator, خذ نظرة وبس

طيب بعدها عندنا حاجة اسمها Operator associativity rule

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

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

طيب هذي الرياضيات الطبيعية وهو اننا نعطي الأولوية للاشارات حسب معرفتكم السابقة

في لغة APL لا ماعندها ذا الكلام, الـ APL اولويتها من اليمين لليساير يعني تجمع  $4 + 5$  وبعدين المعادله تصير  $2 - 3 * 9$  وبعدها تصير  $2 - 27$  ويطلع الجواب -25

## عندنا حاجة اسمها Potentials for side effects

طبيب نشرحها بشكل مفصل وحلو, هيا المشكلة هذي تصير لمن تستخدم متغير مايكون local موجود في ال-main طبيب 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

## طبيب خلونا نسوي trace للكود

اول شي حيخش المين وبيعمل متغير اسمه x وبيكون قيمته عبارته عن جمع قيمة  $fun() + 10 + a$  طبيب اول شي خش الدالة  $fun()$  وكانت قيمتها 30 لأنها تسوي  $return 30$  بعدها جمع الـ 30 مع 10 صارو 40 بعدين جا بيجمع قيمة الـ a, بس ماعنده في الـ main قيمة الـ a, فراح يضطر يستخدم الـ global الي قيمتها 10 وراح يجمع الـ 40 مع الـ 10 ويدينا الناتج 50

طبيب هنا راح اشرح حاجة مو مطلوبة في المنهج بس راح تفهمك ليش استخدم الـ a الي هي global بدال الي موجودة في الدالة fun (للفائدة للي بيغا يفهم لو ماتبغا عادي سوي سكيب للمربع الأزرق ذا وروح كمل بعده.

الي صار هنا أول شي بيغا يجمع قيمة الـ a مع 10, طبيب أول حاجة الـ a مالها قيمة واصلا ماتم تعريفها قبل كذا في ال-main

طبيب هنا زي مانعرف الـ scoping حق الـ functions نقدر نستخدم متغيراتها أو متغيرات ابوها الي هي الي فوقها, في هذي الحالة ماعندنا 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 الثانيه ويصير 20, راح يجمع الـ 20 الحين مع قيمة fun, راح نلاحظ أن fun فيها a بس على حسب مدارسنا انه لمن نخش للـ fun راح نسوي قيمة جديدة للـ a والتي هي 15 ولمن نخلص من الدالة راح نسوي deallocation للمتغير فراح ينحذف ونرجع للمين نجمع الـ return الي هو 30 مع الـ 20 ونطبع الناتج الي هو 50 (حرفياً الـ a الي جوا الـ fun مرا مامننا فايده في الحالة ذي)

نرجع نكمل الكود, بعدها موجود عندنا  $a=10$ , هنا تم تعريف متغير في الـ stack اسمه  $a$  وقيمته 10

طيب نكمل راح يسوي متغير اسمه  $y$  وراح يحط فيه قيمة  $fun + 10 + a$

أول شي راح يخش الـ  $fun$  وأول شي حيصادفه هو انه راح يسوي  $overwrite$  لقيمة الـ  $a$  وراح تصير 15, وبعدها حيسوي  $return$  بقيمة 30 بس **لاحظ هنا حط مليون خط**, بحكم أن الـ  $a$  الي في الـ stack هي حق الـ  $main$  فماراح يصير لها  $deallocation$  لمن ينتهي من الـ  $fun$ , بس اللهم انه حيصير لها  $overwrite$  بالقيمة الجديدة الي هي 15 بعدها راح يجمع قيمة الـ 30 حق الـ  $fun$  مع الـ 10 ويدينا 40, وراح يجمع الـ 40 مع الـ  $a$  يصيرو  $55=15+40$

**لازم نفرق بين الحالتين وننتبه, هنا بيان مين المبرمج الحقيقي ومين الفاهم في التفاصيل الصغيره التفاهة ذي**

بعدها فيه سلايدين يتكلم عن الـ Overloaded Operators

بالعربي هي لمن يكون عندك operator (اشارات رياضية زي + - وكذا), ويكون لها اكثر من استخدام

يعني مثلاً الـ + تستخدم لجمع الارقام, ولدمج النصوص وكمات تسوي دمج للـ arrays, هذا يسمونه operator overloading, عشان نحل المشكلة ذي لازم نسوي لكل عملية متغير 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 تكلم عن الـ **type conversion** ويشمل حاجتين

**الأول: casting** او برضو يسموه **narrowing**

هو التحويل من نوع متغير كبير الى صغير, زي من float الى int, ولازم المبرمج يكتب هذا الشي لانها ماتصير

تلقائياً من اللغة او الكومبايلر (**Explicit type conversion**)

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

Type Cast  
Operator

**الثاني: coercion** او برضو يسموه **widening**

هو التحويل من نوع صغير الى كبير, زي لمن تخزن int داخل متغير float ومو لازم يكتبه المبرمج

وهذا الشي يصير تلقائياً من الكومبايلر (**Implicit type conversion**)

بعدها فيه كم سلايد يتكلم عن بعض الاشارات واشياء زي الـ or, and واشياء سهله واتحتاج شرح, فقط قراءة

طيب نجي حاجة بسيطه يقولك الـ JavaScript والـ PHP عندهم اشارة وهي ===, وش الفرق بينها وبين ==

الـ == يساوي الـ value بدون مايطالع 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 true ماراح يشيك على القوس الثاني عشان توفير الوقت, لأن بالأساس الـ or لو وحده منهم true خلاص كلهم يصيرو true فماراح يفرق الثاني true ولا false, لآكن لو الاول كان false راح يشيك على الثاني

السلايدات الجاية كلها كلام فاضي واشياء نظرية درسناها في برمجة 1, اقراها بس وبتفهمها باذن الله

## CH 8

طيب اول شي في هذا الشايتر هو **Control Structure** وتتقسم لثلاث اجزاء

### الجزء الأول:

#### Tow-way selection statements

الكتاب واضح من عنوانه يعني عندك selection بطريقتين مثل if....else if يايروح لل if او لل else ما عنده غيرهم  
خط ببالك ممكن يكون عندك nested selection يعني if جوتها if و else ثانية, مثل:

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

#### Multiple-way selection statement

زي ال switch او ال if الي تكون كثيره مثل:

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

### الجزء الثاني:

#### Counter-controlled loops

ببساطة هي ال loop الي تحتوي على counter وشرط للنهاية وقيمة الزيادة, مثل:

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

زي الكود الي فوق فيه counter يبدأ من 0 وفيه شرط للنهاية الي هو طول ال array والزيادة الي هي 1 (i++)

## Logically-controlled loops

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

وهذي منها نوعين:

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

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

الباقي كله كلام ينفهم بمجرد النظر

هذا 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 method فيها loop وجوتها swap انت كذا بتفهم ان هذي الـ method الفايده منها **sorting**

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

3- يقولك الـ header حق الـ method هو عبارته 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 ؟** لان من خلاله تقدر تفرق بين الـ methods لو كان عندك تشابه فالأسماء, نفترض عندنا برنامج فيه 2 methods

**Student(string name, int id, float gpa)**

**Student (int id, string name, float gpa)**

طيب حاليا الاسم 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 الاسم بعدين المعدل كذا تنادي الثانية

عشان كذا اسمه توقيع signature لأن من خلاله تقدر تفرق بين الـ methods ويخلي 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 تقسمه 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 ويمدك تسويها وتستخدمها

شايفين الكلام التافه الي فوق؟ اغلب الشايتر ذا كلام زي كذا, اشياء ماتحتاج شرح مجرد قراءة تكفيك بأذن الله, لو كان عندك سؤال في جزئية معينه تقدر تسأل زملائك بأذن الله ما يقصروا او تواصل معاي والله يعين ويبسر للجميع

آخر شي والله الحمد, يتكلم عن الـ Function والـ Procedures

الـ Function: عبارته عن مجموعة اوامر, تاخذ input واحد او أكثر, وترجع قيمة return a value

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

أسم function اخذته من الرياضيات بحكم انها تسوي عمليات حسابية داخلها

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

يتشابهون بشكل كبير بس الفرق بينهم في جزئية الـ return هل هي ترجع قيم ولا لا ؟

وشكرا لكم 😊

الله يوفقكم ويسعدكم, ولا تنسوننا من دعائكم