FACULTY OF COMPUTING & INFORMATION TECHNOLOGY
KING ABDULAZIZ UNIVERSITY

كلية الحاسبات وتقنية المعلومات
جامعة الملك عبدالعزيز

FCIT
KAU

بسم الله الرحمن الرحيم

CPCS 301 – Programming languages

2nd semester 2020

Summary for the final exam

هذا التلخيص هوا جهد احد الطلاب فـ راح اوضح نقطتين

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

ثانياَ: انا طالب مثلي مثلكم وهذا مجرد اجتهاداً مني, وهذا مايعني انه كل الكلام الي شرحته صح ولاكن انا بحثت ودورت ونقحت المعلومات وكتبت الاصح ولاكن ان كان فيه خطا او شي كلموني ع الواتس او اي وسيلة تواصل اخرى وراح اعدله باذن الله

عبدالله الكثيري

CH 6

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
object	الـ object هوا عباره عن data-type زيه زي ال integer, string بس ال object يتم تعريفه من قبل المستخدم (زي لمن كنا نسوي linked list في برمجة3 كانت عباره عن نوع بيانات بس احنى بنفسنا نعرفه ونحدد خواصه وكذا, مايكون جاهز في اللغة نفسها زيه زي ال integer, string

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

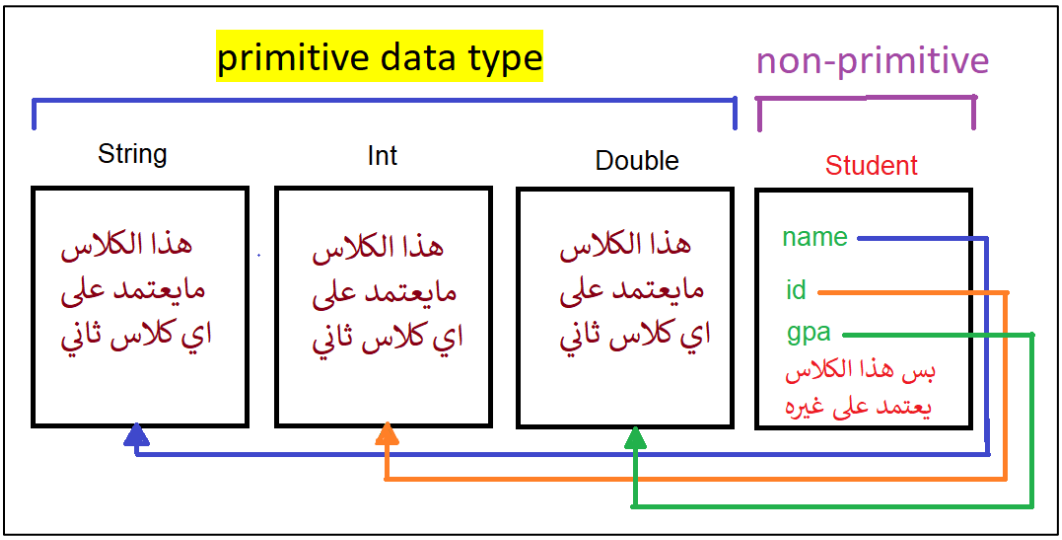
2- ويقولك ال primitive data types ماتعتمد او مايتم تعريفها او تكوينها بالأعتماد على اخر data type

ايش يعني ؟... ناخذ 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 فهوا اصلا معتمد على اشياء ثانية بداخله

هذي صوره بسيطة رسمتها لكم, ان شاء الله توصلكم الفكرة



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

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

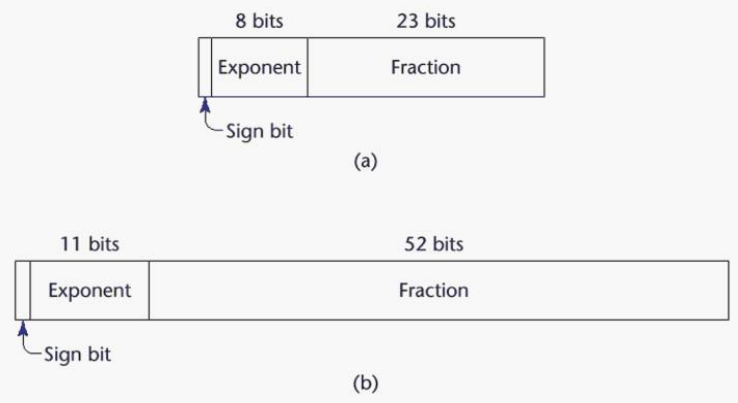
1. Almost always an exact reflection of the hardware so the mapping is trivial
2. There may be as many as eight different integer types in a language
3. Java's signed integer sizes: `byte`, `short`, `int`, `long`

1- يقولك انها اكثر نوع يمثل الهاردوير, نشرحها شوي
تفتكرو ال array in C language ؟ كانت تمثل او تعكس صوره عمل الميموري
برضو الريجيسترات الي درسناها في بنيان الحاسب (اتوقع الاغلب درس الماده) الريجيسترات كانت تتمثل بالارقام وهذا يسهل عملها وتفاعلها مع الهاردوير
2- ممكن يكون عندك اكثر من نوع integer في نفس اللغة (اقرا النقطه 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

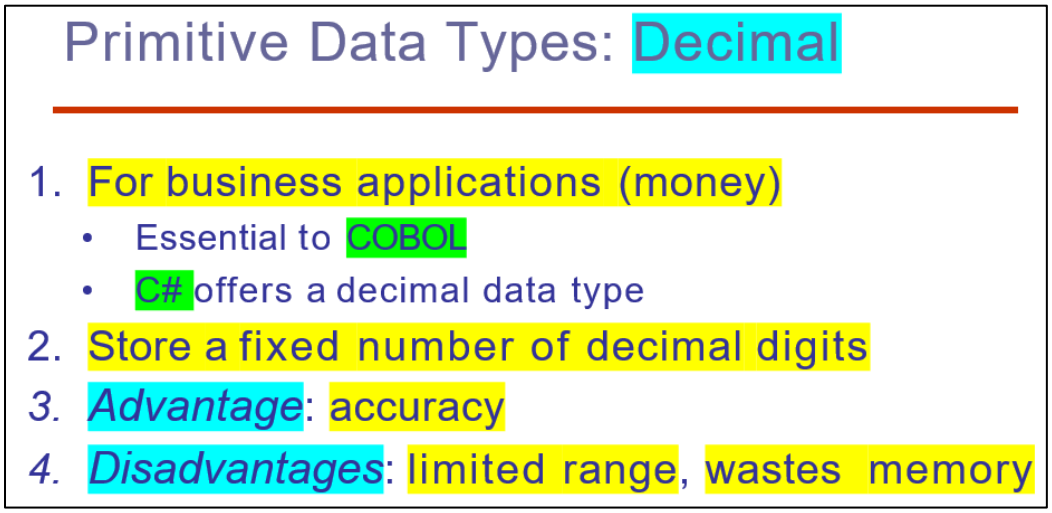
1. Model real numbers, but only as approximations
2. Languages for scientific use support at least two floating-point types (e.g., `float` and `double`; sometimes more
3. Usually exactly like the hardware, but not always
4. IEEE Floating point standard 754



1- يتم تمثيل الأعداد الحقيقية بها (الأعداد الحقيقيه يقصد فيها اي شي فيه فاصلة)
2- اللغات العلمية او المستخدمة في ال science على الأقل تستخدم نوعين وهم double, float وممكن اكثر
3- يقولك انها تشبه ال hardware بس مو دايماً
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



1- تستخدم في ال business ومجالات الاعمال والأموال, زي لغتين هم COBOL, C#

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

Fixed number	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 الهللات 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)
Decimal digits	هذي الاعداد المتغيره الي ماتدري وش ممكن تخزن فيها زي 5246.2324458 او 21.1 اي شي يمشي

فهمتوها ؟ اذا لا بفصلها لك زياده, الحين فل 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

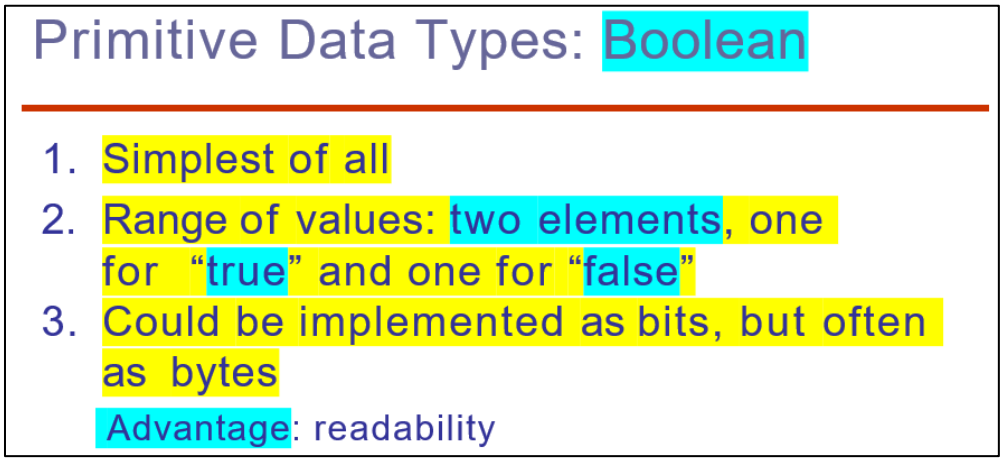
لأني انا حاط بمخي انه دايم المعدل يكون رقم واحد يسار الفاصله وهو يكون 1 او 2 او 3 او 4 او 5 ويمين الفاصله دايم يكون فيه رقم او رقمين تبدا من 00 الى 99 هذا مثال على Fixed number

لاكن لو مثلا كنت تقرا كتاب فيزياء وجو قالولك ثابت سرعه 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

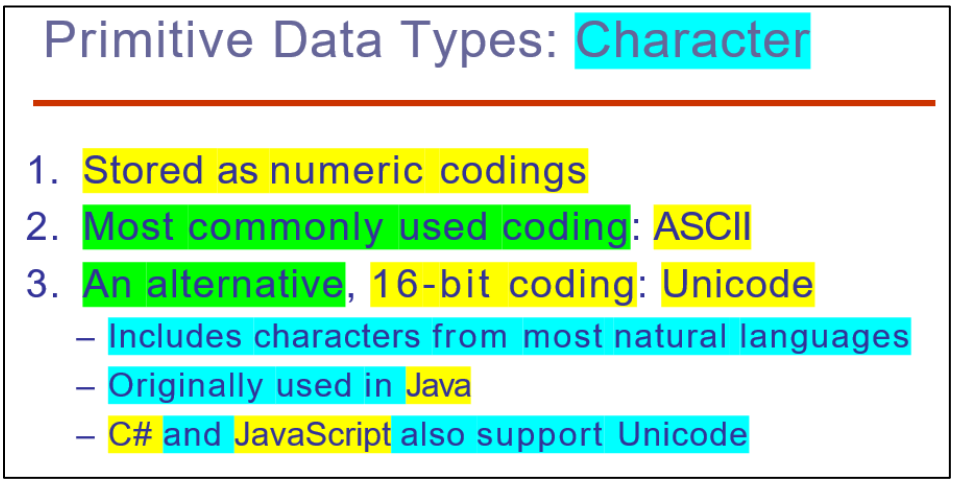
رابعاً Boolean



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

ميزتها : readability

خامساً Character



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 الـ 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 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 للـ Design issues
1- يقولك هل هي عباره عن primitive data type؟ ام انها عباره 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 حقك

```c
#include<stdio.h>
int main()
{
    char name[] = "abdullah";
    printf("%s",name);
    return 0;
}
```

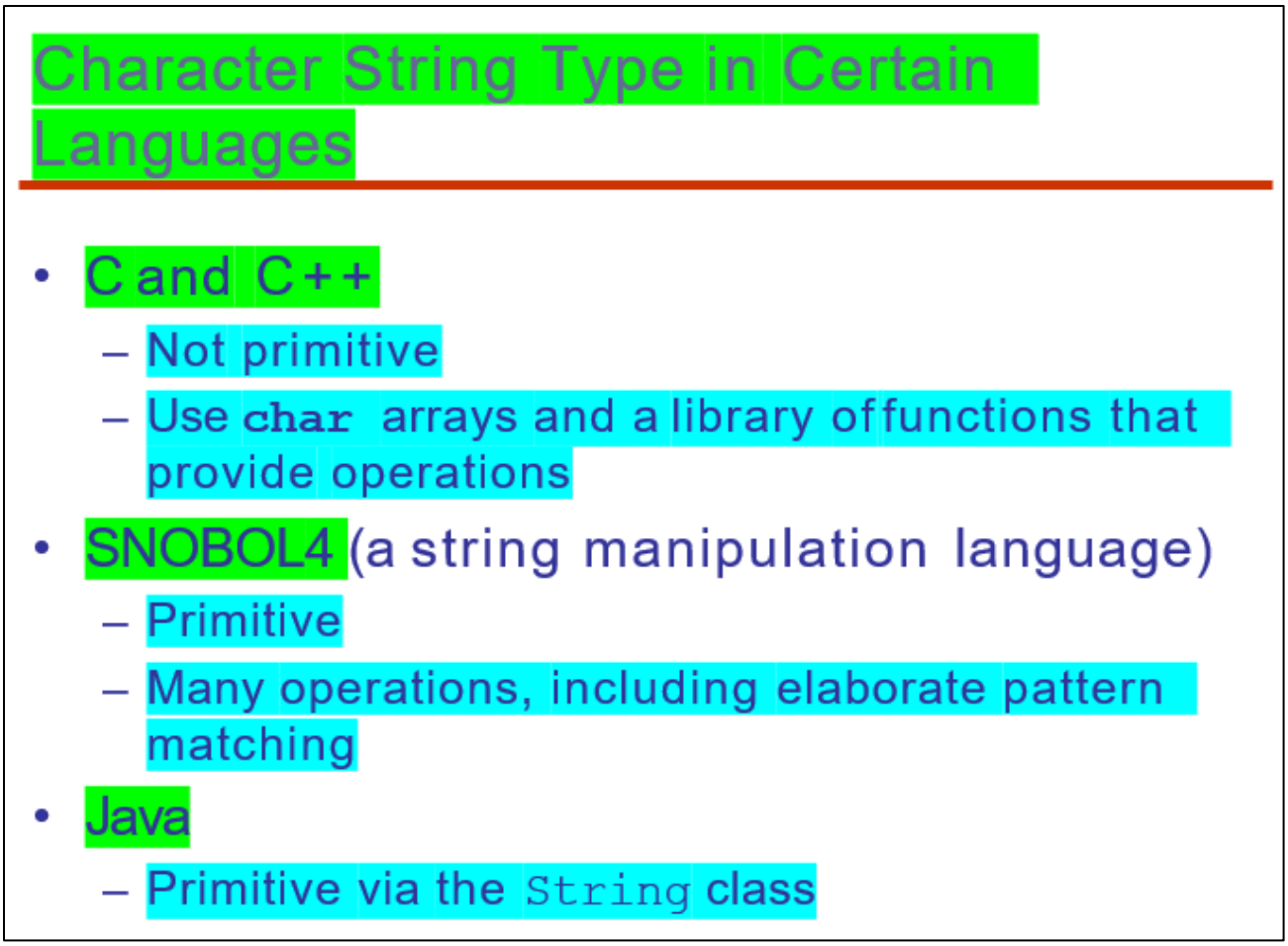
```
abdullah

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

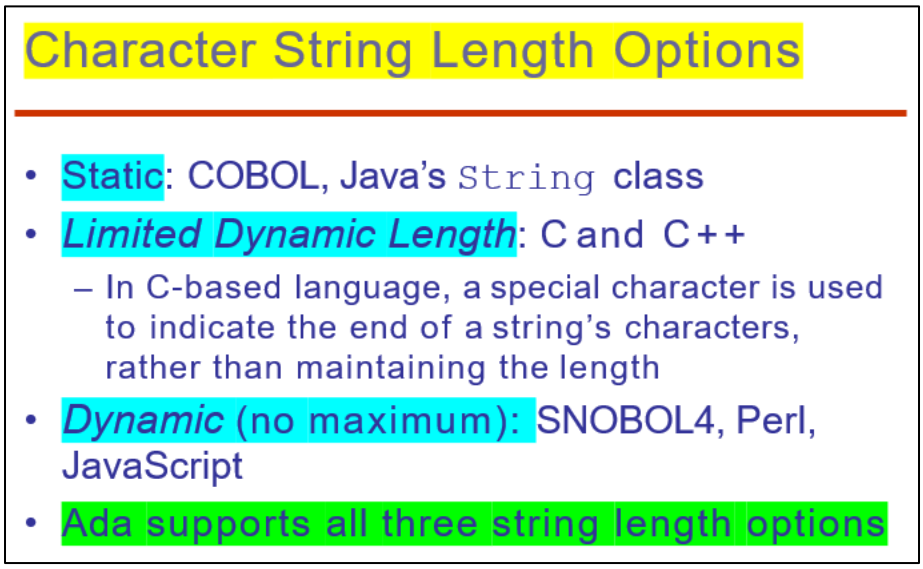
وهذا مثال عشان توضح الصوره

طيب السلايدات الي بعدها راح يتكلم عن ال regex وهوا يعتبر شي قديم وشرحته في الملخص الي فات اذا ماقريت الملخص الي فات راح احط لك الرابط هناك مشروح 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 الregex كيف تجاوبه ؟
يا انك تكون فاهم وتجاوب من فهمك
او تستخدم الموقع الثاني تقدر تحط فيه اي regex ويطلع لك الجواب(برضو حطيته في الملخص الي فات)
https://regex101.com/



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)
وكمان يتم استخدام array of characters عشان نستخدم ال string (برضو شرحتها فوق)
2- لغة Snobol4
تعتبر primitive يعني مبنية وموجوده باللغه بشكل اساسي, وتدعم functions كثيره تتعامل مع String زي length وباقي الدوال الي ممكن تطبقها على اي متغير من نوع السترينق
3- لغة Java
تعتبر primitive من خلال الكلاس حق الstring



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

نوع الString	شرح	اللغات الي تستخدم
Static	لمن تعرف متغير وسميته Abdullah وبعدين تغير قيمته وتخليه 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 الي حيصير بالزبط انه راح نسوي opject جديد ونخزن جوته قيمه وهي ahmed والبوينتر الي كان اسمه name راح يأشر على الopject الجديد حقنا الي جوته قيمة ahmed وراح نطنش الopject القديم	Cobol, Java
Dynamic (limited)	هنا ماراح نحتاج نسوي 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 قيمه ahmed في نفس الopject 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	C, C++
Dynamic (not limited)	زي الي فوق بس هنا ماله نهاية تقدر تسوي طوله قد ماتبغا	Snobol4, Perl, JavaScript

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 حقك
طوله
عنوانه في الميموري

Static string
Length
Address

Compile- time descriptor for static strings

هنا السترينق حقك
اقصى طول ممكن
طوله الحالي
عنوانه في الميموري

Limited dynamic string
Maximum length
Current length
Address

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 , 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 القيمه 4, بعدين سألتك وقلت لك وش القيمه الي بعده ؟ راح تقولي على طول 5, ليش طيب ؟ لانها تعتبر متسلسله وورا بعض لمن اديك عنصر منه راح تعرف تجيب الي قبله او الي بعده, هم لهم تسلسل وزي 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.....
ماحتقدر تجاوب, لأنها تحتمل 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 بالمختصرهي عباره عن set فيها elements والسلام عليكم
زي ما انتو شايفين 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

```java
public class HelloWorld{

    public static void main(String []args){
       String name = "abdullah";
       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 الstring

```
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 الـ Days

طيب بعدها بنخش في موضوع الـ array وكذا, طيب عندنا Design issues خلونا نشوفها

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 الـindex

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

وفي غالبية اللغات الاخرة يستخدمو ال [] 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 الـ types حق الـ index الي ممكن نستخدمها...
يعني مثلا سوينا 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 يديك ايرور وراح يرجع لك قيمه عشوائية من الميموري, لأنه ما يتأكد من الrange
وال Java, ML, C# لو سويت نفس الي فوق وطبعت الأندكس رقم 100 راح يديك ايرور, لانها تتحقق
هل الأندكس الي انت تبغاه جوا الرينج الي بين 0 و 10 او لا

طيب بعدها بيتكلم عن الـ Categories للأندكس
بنحطها ف جدول عشان تزبط 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 مايركز على الاشياء النظرية ذي, حتى لو مافهمت سوي 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://mpstudy.com/describe-subscript-bindings-and-array-categories/

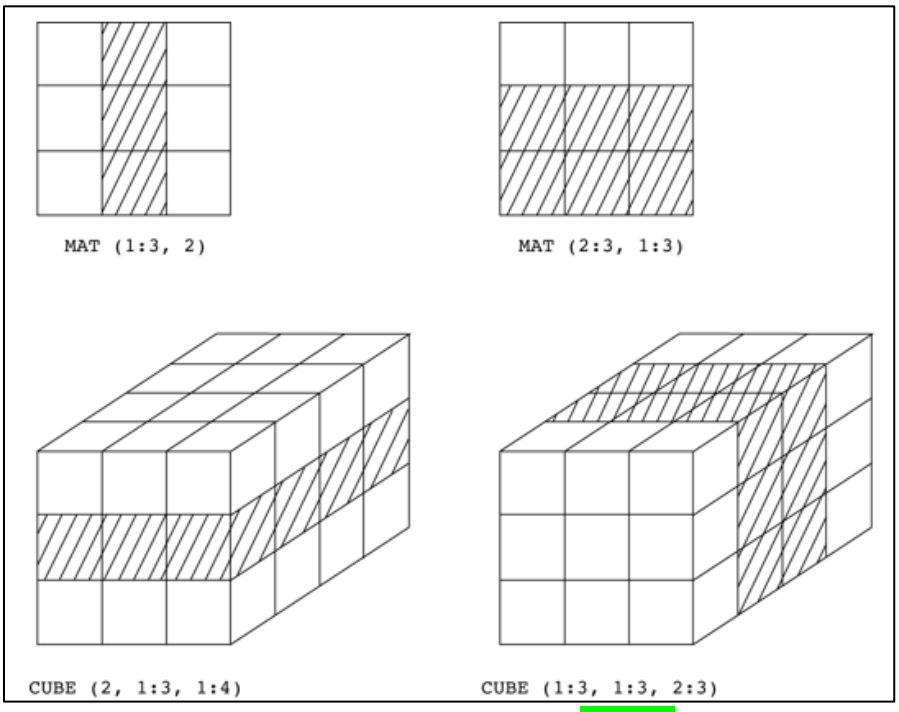
بعدها فيه كم سلايد يتكلم عن كيفيه انشاء الاراي في بعض اللغات, اشياء سهله بس مر عليها قراءة ماتحتاج شرح

بعدها يتكلم عن شيئين

Rectangle array وهي الاراي المربعه 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 عباره عن قطعه او جزئية من اراي ثانيه
زي ماهو واضح ف الصوره 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:

address(list[k]) = address (list[lower_bound])
+ ((k-lower_bound) * 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

Ex: address(list[5]) = address(list[0])+((5-0)*4)
address(list[5]) = 500 + 20 = 520

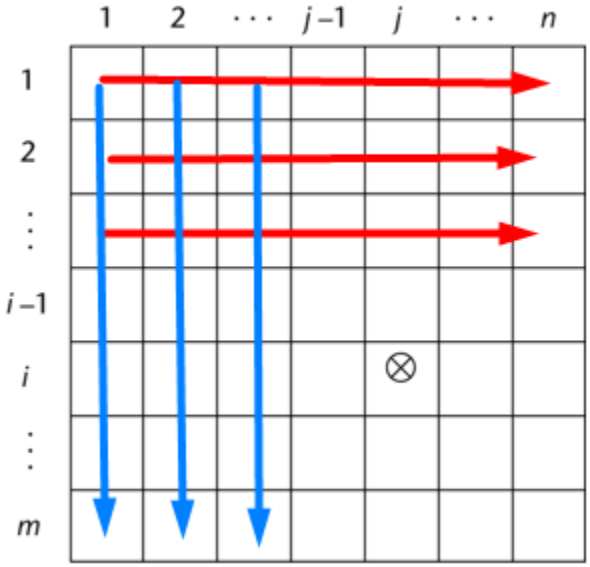
في الصوره ذي يشرح لك كيف تجيب الaddress حق اي عنصر بس بأستخدام الاندكس تبعه
يعني لو قلت لك جيب ال address حق العنصر رقم 5 في الاري الي اسمها list راح تسوي زي المثال الي فوق ^^^

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 (i,j) 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 الفكره وماعليك

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 نحط مثال... بيانات المواطنين في سيرفرات وزارة الداخليه

تمام ؟ كل شخص موجود اسمه هناك هوا عباره عن ريكورد, الريكيورد هوا عباره 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;
```

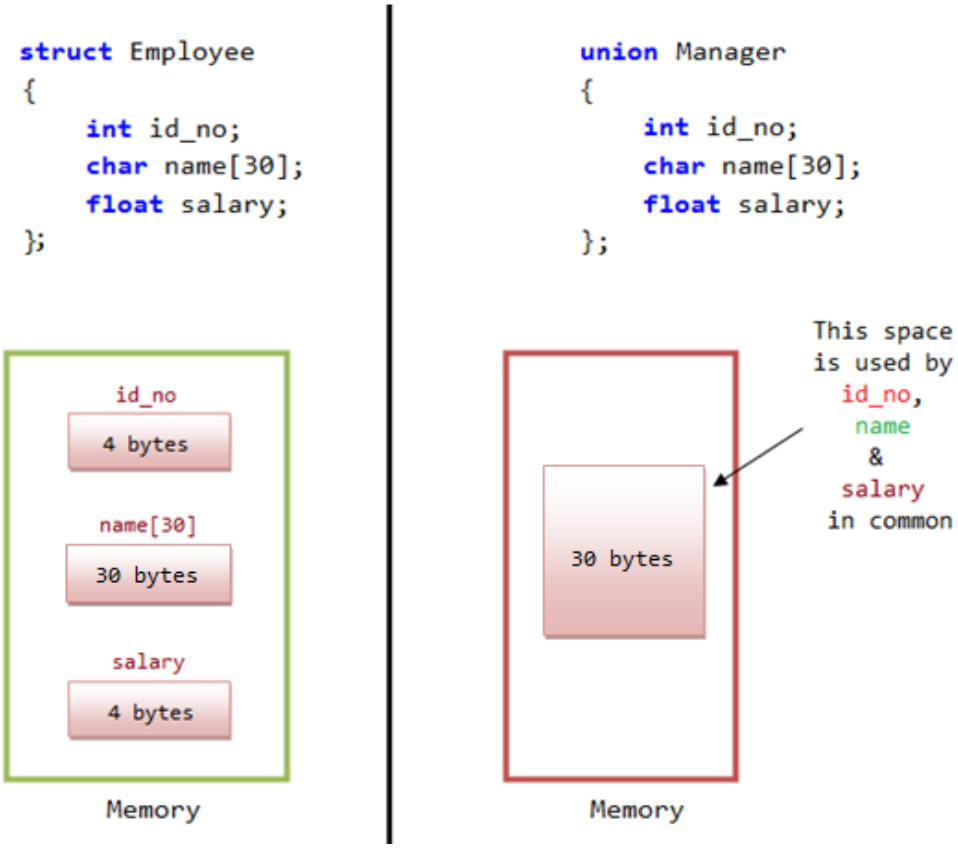
string

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



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 حط 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

الpointer هو الشي الي يأشر على الmemory والrange حقه هي قيم الميموري

حسب علمي انها تبدا بـ 00000000 وتنتهي 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 للـ Pointers وهم: assignments و dereferencing

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

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

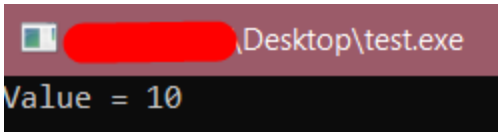
مثل هذا السطر ينكتب في لغة C (مو موجود بالسلايدات بس حطيته لكم عشان تفهمون الفكره)

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



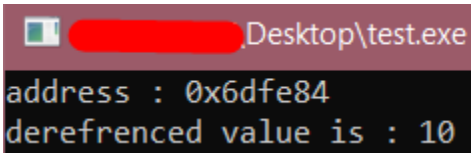
هذي العملية يسموها assignment لأنك جبت قيمة الادرس حق kau وسويت لها assign في متغير من نوع pointer وبعدها من خلال ال pointer تطبع القيمة (تتعامل بالـ address مو بالمتغير نفسه)

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

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

int main(){
int id = 10;
int * id_pointer = &id;
int derefrencing = *id_pointer;
cout << "address : " << id_pointer << endl;
cout << "derefrenced value is : " << derefrencing << 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 opject اسمه id_pointer (assignment)

بعده جبت القيمة الي يأشر لها الـ pointer الي اسمه id_pointer وخزنتها في متغيراسمه 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... كيف كذا ؟

الي صار اني خزنت قيمة الميموري الي هي 0x6dfe84 في متغيراسمه 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 بالسلايدات 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 الحمدلله تم شرح المفهوم بالأمثله والنتايج.

بالنهايه يمكن اني زودت شوية كلام بس حبيت 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 يأشر على variable والـ variable هذا ينحذف, 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 يأشر على متغير pointer1=a و pointer2=b

عند السهم الأخضر, خليت pointer2 يساوي قيمة pointer1 يعني كل الـ pointers صارو يأشروا على a حاليا المتغير b مانقدر نوصل له لأنه مانقدر نوصله, لأن كل الـ pointers يأشرون على مكان ثاني

وهذا الـ output حق الكود الي فوق عشان تعزز فهمك

```
[redacted]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
```

CH 7

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

الأولوية	العملية
1	الاقواس ()
2	الـ unary operators الي هيـ +,-,++,--,!,~
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 ➔ -(8%3)➔ -2
  - **Perl** gives unary operator "–" a higher precedence
  - X mod Y = X - ⌊x/y⌋ * Y

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

هنا يقولك ان لغة Perl تدي ال – أولوية اعلى من الـ %, فراح تفرق النتيجة عندك لمن تحسبها زي الكومبايلر حق الـ 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 حتكون الاشارة قبل, مثل: a= او a+ المهم اي اشارة تكون قبل الحرف

ملاحظة: op = 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 بالاخير بعد الحرفين

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

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

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
```

هذي مجرد امثلة على الـ Ternary operatos, خذ نظرة وبس

طيب بعدها عندنا حاجة اسمها Operator associativity rule

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

بالبداية بتضرب ال 3 في 4 وبيصير 5+12-2, 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 لليسار يعني تجمع 5+4 وبعدين المعادله تصير9*3-2 وبعدها تصير 27-2 ويطلع الجواب 25-

عندنا حاجة اسمها Potentials for side effects

طيب نشرحها بشكل مفصل وحلو, هيا المشكلة هذي تصيرلمن تستخدم متغير مايكون local موجود في الـmain

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()

طيب 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

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

الي صار هنا أول شي يبغا يجمع قيمة الـ a مع 10, طيب أول حاجة الـ a مالها قيمة واصلا ماتم تعريفها قبل كذا في الـmain

طيب هنا زي مانعرف الـ scoping حق الـ functions نقدر نستخدم متغيراتها أو متغيرات ابوها الي هي الي فوقها, في هذي الحاله 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 ويصيرو 20, راح يجمع الـ 20 الحين مع قيمة fun, راح نلاحظ أن fun فيها a بس على حسب مادرسنا انه لمن نخش للـ fun راح نسوي قيمة جديدة للـ a والي 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 مامنها فايدة 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 الـmain 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 40+15=55

لازم نفرق بين الحالتين وننتبه, هنا يبان مين المبرمج الحقيقي ومين الفاهم في التفاصيل الصغيره التافهة 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 مثلا الـ + تستخدم لجمع الارقام, ولدمج النصوص وكمان 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 تلقائي 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 لحاجة 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, لاكن لو الاول كان false راح يشيك على الثاني

السلايدات الجاية كلها كلام فاضي واشياء نضرية درسناها في برمجة1, اقراها بس وبتفهمها باذن الله

CH 8

طيب اول شي في هذا الشابتر هو Control Structure وتتقسم لثلاث اجزاء

الجزء الأول:

Tow-way selection statements

الكتاب واضح من عنوانه يعني عندك selection بطريقين مثل if....else يايروح للـ if او 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

ببساطة 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 بـ1 (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 **بعد** الـ block مثل: **do** **loop body** **while (ctrl_expr_T)**
Pre-test	الي يكون فيها الشرط حق الـ loop **قبل** الـ block مثل: **while (ctrl_expr_T)** **loop body**

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

الباقي كله كلام ينفهم بمجرد النظر

CH 9

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 method فيها loop وجوتها swap انت كذا بتفهم ان هذي الـ method الفايدة 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 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 ؟ 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 الأسم واحد, بس على حسب ترتيبك للمتغيرات وقت الأستدعاء 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

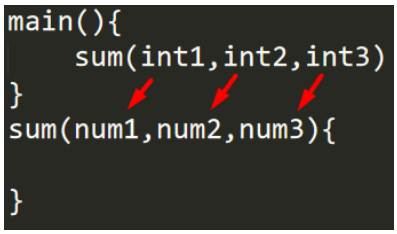
عشان كذا اسمه توقيع signature لأن من خلاله تقدر تفرق بين الـ methods ويخلي كل وحدة غير عن الثانية

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:



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, cources, projects, ......,)
```

4- اغلب لغات البرمجة تدعم الـ methods ويمديك تسويها وتستخدمها

شايفين الكلام التافه الي فوق؟ اغلب الشابتر 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 وييسر للجميع

اخر شي ولله الحمد, يتكلم عن الـ Function والـ Procedures

الـ Function: عباره 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 ولالا ؟

وشكرا لكم😊

الله يوفقكم ويسعدكم, ولا تنسونا من دعائكم