التاريخ: / / الموضوع: _____

Lec 3

Error Types:-

1- Syntax error
2- Run-time error
3- Linker error
4- Logical error
5- Semantic error

Programming errors are also known as The bugs or faults, and the process of removing these bugs is known as debugging.

1- Syntax error

- syntax errors are also known as The compilation errors as they occurred at the compilation time, or we can say That → Thesyntax errors are thrown by the compilers.

* الأخطاء دى بتحدث أثناء الكتابة عشان انت متبعتش قواعد اللغة أو بتكتب حاجه مخالفه لقواعد اللغة وكل لغة ليها قواعد

2- Run-Time error

- sometimes the errors exist during execution-time even after The successful compilation known as run time error, The compilar doesn't point to these error
- The divion by zero is the common example of The run Time error

لو ظهر للدكتور ده مثلا بيقا Run time error

```
Process returned -1073741676 (0xC0000094)
```

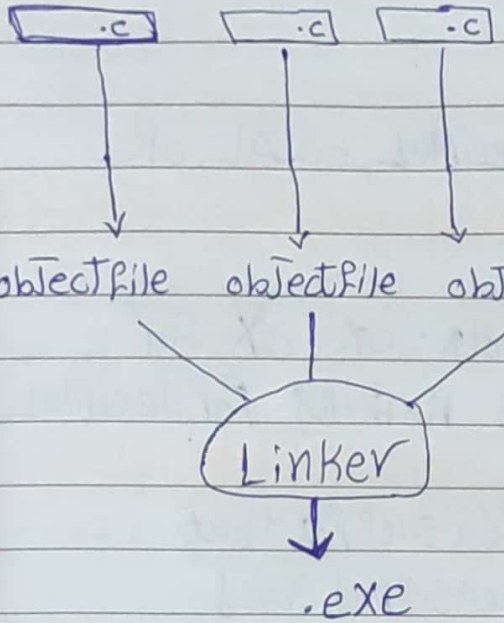
3- Linker error

- Linker errors are mainly generated when the executable file of the Program is not created

- This can be happened either due to the wrong function Prototyping or usage of the wrong header file.

- For example, using a variable declared without any definition.

ال Linker error بيحصل ازاى ؟
لو انا عندى مجموعة من الفايلات زى الشكل دا



ال compiler بيعمل كل فايل لوحده ← machine code ← object file
وبعدين بيجى على عملية Linking
ال Linker بيحول الـ object files لـ .exe

4- logical error :-

- is an error that leads to undesired output
- These errors produce the incorrect output
- The occurence of these errors mainly depends upon the logical thinking of the developer

التاريخ: / / الموضوع: _____

5- Semantic error

semantic errors are the errors that occurred when the statements are not understandable by the compiler

Example:-

1- use of an uninitialized variable
2- errors in expressions
3- Array index out of bound
4- Type compatibility

Literals :-

integer Literals:-

- An integer literal can be a decimal, octal, or hexadecimal constant.

- A Prefix specifies the base or radix: ox oX For hexadecimal, o For octal and nothing For decimal ob For binary
- An integer litral can also have a suffix that is a combination of U, u for unsigned and L l For Long

Example :-

Suffix	Prefix
215U	85 → decimal
215u	ox43 → hexa
210L, 210l	o213 → octal
210ul, 210UL	← مينفعش اكتب البريفكس x و الـ octal
	← // اكتب UU أو LL

oxfeeL

Prefix ← ↳ Suffix

Floating Point Literals :-

has an integer Part, a decimal Point, a fractional Part and an exponent Part.

لازم وانت بتكتب Floating Point literals تخلي بالك من:

١- لو مكتب رقم في e أو E لازم يجي بعد e أو E رقم

٢- لو مكتب رقم في f أو F لازم يكون في decimal Point

3145 e-5 ✓ 3145 E-5 ✓

3.14F ✓ 314 f ✗ → invalid suffix

3.14 f ✓ 314E ✗ → exponent has no digits

.E-5 ✗ → expected expression before (.) token

Character Literals :-

are enclosed in single quotes → 'x' can be stored in a simple variable of char type

String _ Literals :-

- string literals or constants are enclosed in double quotes" "

- A string containg characters that are similar to character literals

Example :-

→ char arr1[20] = "Embedded Diploma";

→ char arr2[20] = "Embedded \

\

\

Diploma"; → char arr3[20] = "Embedded" " "Diploma";

التاريخ: / /

الموضوع: _____

operators :-

An operator is a symbol that tells the compiler to perform specific mathematical or logical function

Types of operators:

1- Arithmetic operators

+ - * / % ++ --

2- Relational operators

== != > < >= <=

3- Logical operators

&& || !

4- Bitwise operators

& | ^ ~ << >>

5- Assignment operators

= += -= *= /= %= <<= >>=

&= |= ~=

6- conditional operator

7- sizeof operator

sizeof()

التاريخ: / /

Arith metic operator:-

الموضوع:

++ → increment operator, increase the integer value by one

-- → Decrement operator, decrease the integer value by one

Pre-increment and Post increment

→ is used to increment The Value of a variable before using it in a expression

→ in the Pre-increment, Value is first incremented and then used inside the expression

→ is used to increment The Value of variable after executing expression completely in which Post increment is used

→ in the Post-increment, Value is first used in a expression and then incremented

Note:-

if we assign the Post-incremented Value to the same Variable Then The Value of That variable will not get incremented, it will remain the same Like it was before.

- This special case is only with Post [increment / decrement]
- The Pre increment, decrment works normal in this case.

Example

```
Num1 = 2
Num1 = Num1++;
```

Num1 لو حبت طبعت ← النتائج هيكون 2

```
Num1 = 3
Num1 = ++Num1;
```

النتائج هيكون 4 ← لو طبعت Num1

Relational operator :-

← نتائج العلية دى بيبقا True ، False
← بتقوم زى علية مقارنة بين حاجتين
← بستعملها أكتر مع if condition ، For

True = 1
False = 0

Logical operator :-

← نتائج الـ operators دى برضه بتبقا 0 أو 1
← نفس فكرة البوبات المنطقية

&& → Logical and
|| → Logical OR
! → Logical NOT

Bitwise operator :-

&	|	^	~
Bitwise and	Bitwise or	Bitwise Xor	Bitwise not

<< Bitwise shift Left >> Bitwise shift right

A & B

هنا بيعمل ايه؟
← بيحول قيمة الـ A لـ Binary و الـ B لـ Binary
ويعمل and لكل bit مع الـ bit اللى قصادها
ونفس الكلام مع باقى الـ operators

Example :-

A = 60 B = 13
A = 0011 1100
B = 0000 1101
A&B → 0000 1100
A|B → 0011 1101
A^B → 0011 0001
~A → 1100 0011

* الـ Bitwise operator بستخدمها مع الـ bit manipulation
العمليات على bit

لو أنا عندي متغير من نوع char char → 1 byte → 8 bit

unsigned char Btn_State = 0x55;

كدا المتغير دا متخزن فيه 55 بالـ hex يعني بالـ binary ← 0101 0101
وأنا عاوز اعدل على بت معينة اخليها بصفر أو واحد

← عاوز احط فيها 0 هعمل إيه ؟!
هعمل حاجة زي Mask ← وليكن متغير تاني من نوع unsigned char
Btn_State_1 = 0b _____
هشوف أنا عاوز احط في انهي بت 0 واحطها بصفر في الـ Mask
والمش عاوز اغيره هحط بواحد وبعدين اعمل & بين المتغيرين

Btn_state = Btn_state & Btn_state_1

* وليكن عايز الـ Btn_State يبقا بالشكل دا 0000 0101
Btn_State_1 = 0b 1111 1010 هيبقا الـ Mask مخزن فيه القيمة دي

← عاوز احط فيها 1 هعمل إيه ؟!
هعمل ماسك و اشوف انا عاوز احط 1 في انهي بت واحط 1 مكان الماسك
والجزء المش عاوز اعدل فيه هحط مكانه (0) وبعدين اعمل OR بين الاتنين

← نفس الفكرة أيضًا لو عايز اعكس عدد معين من البتات يعملها XOR
مع 1 والمش عايز اعكسه تخليه صفر (0)

0000 1010 ← A
0000 1100 ← B
0000 0110 ← A⊕B
A^B

الموضوع: _____ التاريخ: / /

* الـ BitwiseNot ~
بتعكس بتخلي صفر ← واحد ، واحد ← صفر
0 → 1
1 → 0

* الـ Bitwise xoR ^
نفس فكرة بوابة xoR ← متشابهين بصفر ، مختلفين بواحد
بستعملها عشان عليه الـ toggle

* الـ Bitwise and &
بستعملها لو عايز اصفر بت معينة يعني اخليها بصفر

* الـ Bit wise oR |
بستعملها لو عايز اعمل set لبت معينة يعني اخليها بـ 1