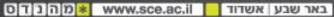


מעבדה 2



תתחבר לרעיונות גדולים







(Functions Overloading) העמסת פונקציות

- שתי פונקציות או יותר בעלות שם זהה, אבל ארגומנט(ים) שונים ידועות כפונקציות עמוסות.
- בשפת ++¢, שתי פונקציות יכולות להיות עם שם זהה אם מספר הארגומנטים/ או סוג הארגומנטים שמועברים שונים.
 - בשפת ++D החתימה של הפונקציה היא השם של הפונקציה ורשימת הארגומנטים
 שלה(לעומת שפת C בה החתימה של הפונקציה היא השם שלה בלבד).
- לא ניתן לבצע העמסת פונקציות אם הן נבדלות רק בערך המוחזר מהפונקציה הערך המוחזר מהפונקציה אינו חלק מחתימת הפונקציה.

- דוגמא - (Functions Overloading) העמסת פונקציות

```
#include <iostream>
using namespace std;
void print(int i);
void print(double f);
void print(char* c);
void print(int i) {
    cout << "Printing int: " << i << endl;</pre>
void print(double f) {
    cout << "Printing float: " << f << endl;</pre>
void print(char* c) {
    cout << "Printing character: " << c << endl;</pre>
int main() {
    // Call print to print integer
    print(5);
    // Call print to print float
    print(500.263);
    // Call print to print character
    print("Hello C++");
    return 0;
```

(Functions Overloading) העמסת פונקציות

כאשר ישנה קריאה לפונקציה מועמסת ++C מחפש את הפונקציה המתאימה -

- 1. אם קיימת התאמה אחת או יותר תיקרא הפונקציה המתאימה ביותר אם קיימת התאמה מדוייקת הקומפיילר יקרא לה, אחרת הקומפיילר יבצע implicit conversion.
 - 2. כאשר אין התאמה או שיש יותר מפונקציה אחת המתאימה באותה קרבה נקבל שגיאת קומפילציה.

(דו משמעות)ambiguity

- כל ה-implicit conversion נחשבות שוות, וכל ההמרות המוגדרות על ידי משתמשים נחשבות שוות.
 - לכן, אם קריאה לפונקציה מתאימה למספר פונקציות באמצעות implicit conversion או המרה המוגדרת על ידי משתמש, התוצאה תהיה ambiguity.

(דו משמעות)ambiguity

```
#include <iostream>
using namespace std;
void print(float f);
void print(double d);
void print(char* c);
void print(double d) {
    cout << "Printing dpuble: " << d << endl;</pre>
void print(float f) {
    cout << "Printing float: " << f << endl;</pre>
void print(char* c) {
    cout << "Printing character: " << c << endl;</pre>
int main(void) {
                                more than one instance of overloaded function "print" matches the argument list:
    print(5);
    return 0;
                      © C2668
                               'print': ambiguous call to overloaded function
```

(ערכי ברירת מחדל) Default Values

- ערכי ברירת מחדל לפרמטרים = האפשרות לתת לפרמטרים ערך דיפולטיבי במידה והמשתמש לא מכניס ערך משלו.
 - . ניתן להגדיר ערכי ברירת מחדל לפרמטרים של פונקציה C++ בשפת
 - התוצאה של מתן ערכי ברירת מחדל היא העמסת פונקציות.
 - הפרמטרים המקבלים ערך ברירת מחדל חייבים להופיע בסוף רשימת הפרמטרים(יכול להיות שכל הפרמטרים יקבלו ערכי ברירת מחדל).
- במילים אחרות: אם נתנו ערך ברירת מחדל לפרמטר כלשהו, אנחנו חייבים לתת ערך ברירת מחדל לכל הפרמטרים המופיעים אחריו (מימינו)
 - הדבר מאפשר להשמיט את הפרמטר בזמן הקריאה לפונקציה.
 - אם משמיטים פרמטר אחד יש להשמיט את כל הפרמטרים שאחריו (מימינו).