**ארדואינו - טיפוסי משתנים**

1. **void** - ללא ערך . משמש בדרך כלל להודיע שפונקציה איננה מחזירה ערך או לא מקבלת ערכים . לדוגמא : void func1( ) אומר שהפונקציה func1 איננה מחזירה ערך .
2. **boolean** - בוליאני. מכיל אחד משני ערכים, או true או false . משתנה בוליאני הוא בגודל ביית - byte . boolean run=false; . ניתן לרשום מאוחר יותר run= !run; ואז הופכים את המצב של המשתנה ל true .
3. **char** - טיפוס תווי. "תופס" ביית אחד בזיכרון ושומר ערך של תו. תו נרשם בגרשיים בודדים, לדוגמא : 'A' . למספר תווים יחד (מחרוזת ) , שמים גרשיים. לדוגמא : "ABC" . התווים נשמרים כמספרים עם קוד האסקי ASCII )) שלהם. לדוגמא : char tav='A'; . ניתן לבצע פעולות מתמטיות על התווים. לדוגמא : 'A' + 1 = 'B' (ערך האסקי של 'A' הוא 41H או 65 עשרוני. כשמוסיפים לו 1 נקבל 66 עשרוני שהוא ערך האסקי של 'B' .

תחום הערכים של char הוא -128 (מינוס 128) עד127 . אם רוצים ביית אחד של מספרים לא מסומנים עדיף להשתמש במשתנה מטיפוס byte .

1. **unsigned char** - תווי לא מסומן. "תופס" ביית אחד ותחום הערכים שלו הוא בין 0 ל 255. עדיף להשתמש בטיפוס משתנה byte .
2. **byte** - ביית. מכיל 8 ביטים של מספר לא מסומן, מ 0 עד 255 . לדוגמא : byte num=250;
3. **int**  - קיצור של integer - שלם . טיפוס משתנה ראשי לשמירת מספרים. בארדואינו אונו ( ובשאר המעגלים מבוססים על ATMega) , הוא מכיל 2 בתים – 16 ביטים. תחום הערכים הוא -32768 ÷ 32767 . בארדואינו Due יש בו 4 בתים – 32 ביטים. ואז תחום הערכים הוא : -2147483648 ÷ 2147483647 ( 2 בחזקת 31 ( . המספרים השליליים נשמרים בשיטת המשלים ל 2 . הביט הגבוה – MSB – משמש כביט הסימן . תוכנת הארדואינו דואגת שבפעולות האריתמטיות נקבל את התוצאות המצופות. בפעולות הזזה ימינה ( האופרטור << ) יכולים לקבל תוצאות לא צפויות . דוגמא : int number=1000; .

כדאי לשים לב שכאשר עוברים את התחום של המספרים יש "גלגול" . לדוגמא : אם הגדרנו int x=-32768; ואחר כך נרשום : x=x-1; אז נקבל x=32767 (עבר למקסימום החיוביי !!). אם נרשום עכשיו x=x+1; נקבל x=-32768 ( עברנו למקסימום השלילי).

1. **word**  - מילה . כמו המשתנה unsigned int אז בארדואינו אונו ( ובשאר המעגלים מבוססים על ATMega ), הוא מכיל 2 בתים ושומר מספרים חיוביים בלבד. טווח הערכים : 0 עד 65535 . בארדואינו Due יש בו 4 בתים – 32 ביטים. ואז טווח הערכים הוא : מ 0 ועד 4,294,967,295 .

גם כאן אם נגדיר unsigned x=65535; ואחר כך נרשום x=x+1; נקבל x=0 ולהפך. אם x=0 ונוריד ממנו אחד אז הוא יהיה שווה ל 65535 .

1. **Long** - ארוך - משתנה של 32 ביטים ( 4 בתים )למספרים שלמים גדולים יותר מטיפוס המשתנה int. טווח ערכים  **-2,147,483,648 עד 2,147,483,647** . דוגמא : long salary = 123456L;
2. **unsigned long** - ארוך לא מסומן . כמו טיפוס long אבל מכיל מספרים שלמים חיוביים בלבד . טווח ערכים 0 עד 4,294,967,295 . (2^32 - 1) שתיים בחזקת 32 פחות 1 דוגמא : unsigned long = 654327L; .
3. **float** - ממשי - טיפוס נתון למספרים בעלי נקודה צפה שיש להם נקודה עשרונית . כלומר מספרים המכילים גם מספר שלם וגם חלק של שבר. (לדוגמא – 213.1478 ) . מכיל 4 בתים וטווח הערכים  **3.4028235E+38 עד -3.4028235E+38** ( E אומר אקספוננט , כלומר 10 בחזקת 38 ). יש לו 6-7 ספרות של דיוק (כולל ספרות השלם). פעולות עם טיפוס משתנה float נמשכות זמן ארוך מפעולות עם שלמים int . דוגמא : float my\_float = 3.127 ; .
4. **double** - ממשי כפול . מספר עם נקודה צפה מדויקת כפולה. בארדואינו אונו ובשאר הלוחות מבוססי ATMEGA הוא תופס 4 בתים. והוא נותן תוצאות כמו float . בארדואינו DUE הוא תופס 8 בתים. במערכות רבות הוא נותן דיוק של 15 ספרות (לא בארדואינו ! ). במידה ומשתמשים בקוד ממערכות אחרות יש לבדוק את התוצאות כי הן לא מספיק מדויקות !! .
5. **string** - מחרוזת (האות s נרשמת באותיות קטנות ולא גדולות). מחרוזות טקסט מיוצגות ב 2 דרכים . אפשר להשתמש בטיפוס הנתון string או להשתמש במערך מטיפוס תווי char שמסתיים ב null . להלן דוגמאות חוקיות להגדרת מחרוזות:

 char Str1[15];

 char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};

 char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};

 char Str4[ ] = "arduino";

 char Str5[8] = "arduino";

 char Str6[15] = "arduino";

האפשרויות שבדוגמאות הן :

* הגדרת מערך מטיפוס תווי מבלי לאתחל אותו כמו ב Str1 .
* הגדרת מערך תווים (עם תו נוסף ) והקומפיילר יוסיף את ה null כמו ב Str2 . ה null נרשם גם '\0' .
* ניתן להוסיף במפורש את ה null כמו ב Str3 .
* אתחול המחרוזת בין גרשיים ללא כמות התווים. הקומפיילר יתאים את מספר התווים ועוד ה null למחרוזת כמו עם Str4 .
* הגדרת המחרוזת עם הכמות המתאימה של התווים לתווים הרשומים בין הגרשיים כמו ב Str5 .
* הגדרת המערך ואתחל שלו עם השארת כמות נוספת של תווים למחרוזת גדולה יותר כמו Str6 .

בדרך כלל יש בסוף מחרוזת את התו null . זה מאפשר לפונקציות כמו Serial.print( ) לדעת איפה המחרוזת מסתיימת.

**מערך של מחרוזות**

 ניתן להשתמש במערך של מחרוזות, לדוגמא - כאשר עובדים עם LCD. לדוגמא קוד המשתמש במערך של פוינטרים מטיפוס char ( נזכור ששם של מערך הוא הכתובת של האיבר הראשון במערך !):

char \*myStrings[]={"This is string 1", "This is string 2", "This is string 3",

"This is string 4", "This is string 5","This is string 6"};

void setup( )

{

Serial.begin(9600);

}

void loop( )

{

for (int i = 0; i < 6; i++){

 Serial.println(myStrings[i]);

 delay(500);

 }

}

1. **String** - מחרוזת עם S גדולה - זוהי מחלקה - class המאפשרת פעולות על מחרוזות טקסט מורכבות יותר מאשר טיפוס המשתנה string . ניתן לשרשר מחרוזות , לחפש תת מחרוזת בתוך מחרוזת, או להחליף תת מחרוזת ועוד. טיפוס זה לוקח יותר מקום בזיכרון ממחרוזת רגילה אבל הוא יותר שימושי.
2. Arrays - מערכים. מערך הוא קבוצה של איברים – משתנים - המסודרים אחד אחרי השני בזיכרון, לכולם אותו השם, כולם מאותו הטיפוס, אבל מה שמבדיל ביניהם הוא המיקום שלהם בתוך הקבוצה. לאיברי המערך ניגשים עם אינדקס. הדוגמאות הבאות מתאימות להגדרת מערך:

 int myInts[6];

 int myPins[] = {2, 4, 8, 3, 6};

 int mySensVals[6] = {2, 4, -8, 3, 2};

 char message[6] = "hello";

ניתן להגדיר מערך ללא אתחול כמו במערך myInts[ ] . במערך myPins[ ] לא קבענו את כמות האיברים. הקומפיילר יספור את כמות הנתונים שכתבנו וזה יהיה גודל המערך ( 5 איברים). במערך mySensVals[ ] קבענו את הגודלואתחלנו אותו. המערך message [ ] הוא למעשה מחרוזת בת 6 איברים המאותחלת ב hello ומקום נוסף ל null ( '\0' ). כדי להכניס נתון לאחד האיברים במערך (לדוגמא לאיבר האחרון במערך myInts [ ] , ניתן לרשום myInts[5] = 100; . יש לזכור את הכלל שחריגה ממערך או מחרוזת היא באחריות המשתמש. כלומר אם מוגדר מערך בן 5 איברים ובלולאת for מכניסים נתונים ל 6 איברים , הקומפיילר לא ייתן שגיאה וכשהתכנית תרוץ ייכנס לכתובת שנמצאת אחרי האיבר החמישי (האחרון ) במערך נתון שייתכן שיפגע במשתנה אחר והתכנית לא תעבוד כרצוי.