שאלה 1

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

1. תמיד **אינו נכון** – לאחר בדיקה מעמיקה הן בספר הלימוד, והן באינטרנט, אני סמוך ובטוח שהתשובה היא **לא נכון לעולם**.

בעמוד 22 בספר הפסקה האחרונה (תחתית העמוד) פותחת כך:

“Array subscripts always start at zero in C, so the elements are ndigits[0]… This is reflected in the for loops that initialize and print the array…”

שפת ANSI C בזהה לשפות רבות אחרות "מעוצבת" מהבסיס שלה כשפה שצורת האינדוקס של משתנים (מטיפוסים שונים) היא אחידה ומתחילה מ-0. הדבר מיושם באופן אינהרנטי בשפת C, כאשר האינדקס הוא אזכור למקום בזיכרון התוכנית עבור משתנה שניצור. בכדי לשנות את ברירת המחדל של אינדוקס המתחיל מ-0 עלינו לשנות את הבסיס עליו השפה מושתת ומפותחת, בכדי שיהיה סטנדרט אחיד לכלל המפתחי C בעולם, על כלל מערכות ההפעלה וסוגי הקומפיילרים השונים, ולא רק לשנות בקוד שלנו את האינדוקס שיתחיל מ-1 🡨 אירוע משמעותי שכנראה לא מאפשר לנו לשנות את ברירת המחדל.

1. לפעמים נכון ולפעמים איזו נכון.

לפי עמוד 149 , פרק 6.9 בספר הלימוד Bitfield הינה הדרך שלנו להצהיר על יצירת אובייקט בעל ערכי ביטים ספציפיים וסמוכים בתוך יחידת אחסון אחת המוגדרת ביישום שמאפשרת למפתח לבצע "אלוקצייה" לזיכרון עבור טיפוסים כמו structs ו-unions בכדי ליעל את השימוש בטיפוסים הנ"ל בזיכרון המחשב. (דוגמה לכך בהמשך בעזרת קטע קוד הצהרתי שכתבתי + שגיאה רלוונטית לחריגה מגודל הביטים האפשרי לטיפוס).

לפי בדיקה שביצעתי ב-IDE ניתן לראות כי ההגבלה של 8 BITFIELDS למשתנה אכן נכונה לטיפוס מסוג CHAR אך לא עבור טיפוסים אחרים כגון INT. עם זאת, גם לטיפוסים האחרים יש מגבלה, כגון 32 ביטים עבור INT (אף עשוי להיות 16 ביטים במערכות הפעלה מסוג אחר).

הוכחה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

המשתנה c חוטף שגיאה כאשר אנחנו מנסים להצהיר בעזרת bitfields ב-struct על טיפוס char בגודל של 19 ביטים – חריגה מעל ל-8 ביטים מהם טיפוס char מיוצר/ נשמר בזיכרון. לעומת זאת ניתן לראות כי char שמוצהר ב-a בעל 8 ביטים, הינו תקין. כמו כן, ניתן לראות ש-32 ביטים שמוצהרים עבור Int d הינם תקינים לעומת החריגה של 33 ביטים עבור b.

ולכן ניתן להסיק ששדה סיביות יכול להיות בעל יותר מ-8 ביטים, תלוי בטיפוס וגם במערכת ההפעלה.