גירסה 1.00 – 3.1.2005

מספר נושאים בשפת C++

ניר אדר

Nir Adar

Email: underwar@hotmail.com

Home Page: http://underwar.livedns.co.il

אנא שלחו תיקונים והערות אל המחבר.

מסמך זה מכיל שני נושאים עיקריים:

1. מספר הבדלים בין שפת C לשפת C++.
2. מספר נושאים ומאפיינים חדשים שהוספו לסטנדרט החדש של C++ ב-1997.

מסמך זה אינו מהווה מדריך שלם בנושא. בחרתי להציג נושאים אלו כי לדעתי הם אינם ידועים מספיק למתכנתים.
אחת הסיבות העיקריות, לדעתי, היא העובדה שהמהדר Borland C++, שהינו לרוב פופולרי עבור מתכנתים מתחילים, אינו תומך בכל הנקודות המוצגות כאן, ולכן לדעתי קיימים אנשים שפספסו נושאים אלו במהלך לימוד השפה.

הבדלים בין שפת C לשפת C++

הקשחת המשמעות של const

שימוש ב-const מגדיר אובייקטים כבלתי ניתנים לשינוי. לפיכך, קטעי קוד העלולים לאפשר שינוי שלהם, לא יעברו קומפילציה.

קטע הקוד הבא יעבור קומפילציה בשפת C, אך לא יעבור קומפילציה ב-C++ (syntax error):

```
const int i = 5;
int *z = &i;
```

בשפת C++ המספר נחשב ממש כקבוע ולא כמשתנה, ולפיכך – לא קיימת עבורו כתובת בזמן ריצה.

הקשחת הדרישות מ-const נועדה לבטל בשפת C++ את הצורך בשימוש ב-#define.
בנוסף, על מנת לבטל לחלוטין את הצורך ב-#define לצורכי מאקרו נוספו לשפה פונקציות inline וכן template functions.

הגדרת משתנים

בשפת C++, אנו יכולים להגדיר משתנים בכל קטע של הפונקציה, ולא רק בראש בלוק. אי לכך, נוכל להגדיר על המשתנה, מיד לפני שנרצה להשתמש בו, ולאו דווקא בראש הפונקציה. למשל, התוכנית הבאה היא תוכנית חוקית לגמרי ב-C++, אולם לא ב-C:

```
int main()
{
	printf("Please enter 2 numbers: ");
	int x, y;
	scanf("%d%d", &x, &y);
	int sum = x+y;
	printf("The sum of x+y is %d\n", sum);
	return 0;
}
```

הסיבה לכך ש-C++ מאפשרת להגדיר משתנים בכל נקודה שנרצה ולא רק בראש בלוק, כמו C, היא העובדה שבאגים רבים בשפת C הופיעו עקב שימוש במשתנים לא מאותחלים.

על ידי הגדרת משתנים בכל מקום בתוכנית, נוכל להגדיר את המשתנים שאנו משתמשים בהם קרוב ככל הניתן לרגע האתחול שלהם ולצמצם משמעותית תופעה זו.

ניתן להגדיר משתנה בתוך הגדרת לולאה או משפט if. משתנה כזה קיים רק בתחום של בלוק זה. למשל:

```
int main()
{
	for (int i = 0; i < 10; ++i) cout << i << endl;
	for (int i = 0; i < 10; ++i) cout << i << endl;

	return 0;
}
```

בכל אחת מהלולאות נוצר משתנה i המשמש את הלולאה, ומיד אחריה מפסיק להתקיים.

בצורה כזו נמנע מאיתנו שימוש בטעות במשתנה הלולאה לאחר שזה סיים את תפקידו. אנחנו בעצם מדגישים בעזרת השפה שהמשמעות היחידה לקיום המשתנה היא המעבר על הלולאה.

הסטנדרט החדש של C++

namespace

namespace הוא מושג חדש, המאפשר לנו לעטוף מחלקות ופונקציות בשם נוסף.
בתוכנית גדולה יתכנו התנגשויות בין שמות מחלקות שונות ופונקציות. בעזרת namespace ניתן לפתור את בעיה זו - נגדיר "מרחב שמות" שיעטוף את המחלקות ויהיה חלק מהשם המלא של המחלקה.
לדוגמא:

```
namespace space1
{
	void func()
	{
		cout << "hello!" << endl;
	}
}

namespace space2
{
	void func()
	{
		cout << "world?" << endl;
	}
}

int main(int argc, char *argv[])
{
	space1::func();
	space2::func();

	return 0;
}
```

נקודות לגבי namespaces:

- Namespaces יכולים להיות מקוננים. במקרה זה שם המחלקה/הפונקציה מורכב משרשור באמצעות :: של השמות השונים.
- Namespaces ניתנים להרחבה. ניתן בכל מקום לכתוב namespace X ולהוסיף עוד פונקציות ומחלקות לאותו namespace.

using

לא נוח להקליד כל הזמן את השם המלא של פונקציה/מחלקה הנמצאים בתוך namespace, ולכן באה המילה using לעזור לנו.

אם נכתוב בקובץ מסויים את ההצהרה:

```
using space2::func;
```

נאמר ל-C++ שבקובץ זה אם נכתוב func() ללא ציון מרחב שמות, אז יש להשתמש אוטומטית בזה של space2.

ניתן לומר ל-C++ להשתמש גם במרחב שמות שלם, למשל:

```
using space2;
```

הצהרה כזו תאמר לשפת C++ להוסיף את כל השמות המוגדרים ב-space2 כך שנוכל להשתמש בהם מבלי לציין את ה-namespace. בסעיף הבא נראה כיצד הספריה הסטנדרטית של C++ מאורגנת ב-namespaces.

שימוש ב-namespace

נהוג לעטוף כל תוכנית ב-namespace כללי, וכל מודול, יחידה לוגית בתוכנית (לא מחלקה, אלא אוסף מחלקות המייצגות רכיב בתוכנית) ב-namespace משלהם. לדוגמא: חלוקת התוכנית ל-namespace: ממשק המשתמש יהיה אחד, הלוגיקה הפנימית תהיה namespace שני.

הספריה הסטנדרטית של C++, צורת השימוש החדשה בקבצי ה-headers

בשפת C++ לפני 1997, צורת הוספת קבצי ה-header היתה דומה לזו של שפת C, לדוגמא:

```
#include <iostream.h>
```

הסטנדרט החדש (1997) קבע שני שינויים המשפיעים על צורת הכתיבה:

1. עבור הספריות הסטנדרטיות של שפת C++, נוותר על הסיומת h.
   למשל, נכתוב iostream במקום iostream.h.
2. כל הספריות הסטנדרטיות נמצאות תחת namespace בשם std. על מנת להשתמש במחלקות ובפונקציות השונות, יש לצרף את ה-namespace או את החלקים ממנו בהם אנו מעוניינים להשתמש בקוד.

דוגמא להוספת ה-header לפי הסטנדרט החדש:

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

איך בעצם ממומש העניין של השמטת הסיומת h. מהמחלקות הסטנדרטיות? התשובה הינה פשוטה – בעבר היו בספריה include של המהדר קבצי ה-h, עם סיומת זו. למשל היה בה את הקובץ iostream.h. כיום הקבצים נמצאים באותה ספריה, אך פשוט ללא כל סיומת.

יש לציין כי ניתן עדיין להשתמש בסיומת h. אם נרצה בכך. הספריות הישנות עדיין קיימות לצורך תאימות עם קוד קיים. יש להזהר עם זאת לא להשתמש בפרויקט אחד בתערובת שחלקה ספריות ישנות, וחלקה ספריות חדשות. במקרה כזה הפרויקט לא יעבור הידור ונקבל שגיאה כי המחלקות הסטנדרטיות יהיו מוגדרות פעמיים בקוד (פעם אחת בכל include).

כיצד ניגשים לספריות של C, כדוגמת string.h לפי הסיומת החדשה?
הגישה אליהן נעשית על ידי השמטת ה-h, בדומה לספריות של C++, והוספת התחילית c.
לדוגמא, במקום לכתוב:

```
#include <string.h>
```

נכתוב

```
#include <cstring>
```