Pointers to function and callback functions

Pointers to function הם דבר נהדר בשפת C

הם מאפשרים לנו תכנות מודולרי, כזה שמזכיר object oriented design in cpp

עם דימיון לפונקציה וירטואלית ב CPP אנחנו מקבלים אפשרות לקרוא למספר פונקציות באמצעות pointer אחד.

נוכל לאתחל מספר רב של פונקציות על ידי מערך של מצביעים ולקרוא לכל הפונקציות באמצעות אותו מצביע

למשל ב for loop

התחביר של pointers to function הוא:

void (\*pfunc)(void);

int (\*pfunc1)(int a, int b);

מגדירים פונקציה עם \* , לפונקציה יש return value ו פרמטרים

לאתחל מצביע כזה:

pfunc1 = &add;

int a = pfunc1(1, 2);

printf("a = %d\n", a);

pfunc1 = &sub;

a = pfunc1(1, 2);

printf("a = %d\n", a);

כאשר הפונקציות הן:

int add(int a, int b)

{

return a + b;

}

int sub(int a, int b)

{

return a - b;

}

החוזק של מצביע לפונקציה הוא במערך:

int(\*functionParameters[8])(char \*str, int size) = { 0 };

מגדירים מערך של מצביעים , ממש כמו מערך של pointers to integer

אבל הפעם נוכל להשתמש במצביע לקרוא לפונקציה:

functionParameters[0] = &read\_mat\_input\_parameters;

functionParameters[1] = &add\_mat\_input\_parameters;

functionParameters[2] = &print\_mat\_input\_parameters;

functionParameters[3] = &sub\_mat\_input\_parameters;

functionParameters[4] = &mult\_mat\_input\_parameters;

functionParameters[5] = &transpose\_mat\_input\_parameters;

functionParameters[6] = &mult\_scalar\_input\_parameters;

functionParameters[7] = &stop\_input\_parameters;

נניח שנרצה לקרוא בשלב מסוים לכל הפונקציות הללו

במקום לקרוא להן כך:

read\_mat\_input\_parameters)”A” , 0);

add\_mat\_input\_parameters)”A” , 1);

print\_mat\_input\_parameters)”A” , 2);

sub\_mat\_input\_parameters)”A” , 3);

mult\_mat\_input\_parameters)”A” , 4);

transpose\_mat\_input\_parameters)”A” , 5);

mult\_scalar\_input\_parameters)”A” , 6);

stop\_input\_parameters)”A” , 7);

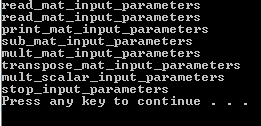
נוכל לבצע for loop

for (int i = 0; i < 8; i++)

{

functionParameters[i]("A", 3);

}



או שנניח שאנחנו קוראים מהמשתמש איזו מספר שהיא פונקציה לביצוע:

int x;

scanf("%d", &x);

functionParameters[x]("Hello", 10);

נוכל בצורה אלגנטית לקרוא ישר לפונקציה לביצוע כי כבר אתחלנו את כל המצביעים

Callback Functions

נושא נוסף ש pointers to function עוזרים לנו הם ב callback functions

נניח שיש לנו פונקציה ארוכה ואנחנו רוצים מידיי פעם להדפיס משהו למסך.

הפונקציה היא פונקציה סגורה והיא לא מדפיסה בתוכה , מה שכן היא גם לא יודעת מה להדפיס , זה לא תפקידה

או שהפונקציה היא אחת שמעתיקה קובץ או שולחת אותו ברשת והיא רוצה לדווח החוצה לפני שהיא סיימה מה המצב שלה.

באמצעות callback function אנחנו מעבירים מצביע לפונקציה שאנחנו רוצים שהיא תקרא לפונקציה שלנו

typedef void(\*call\_back) (int, int);

void informProgress(int a, int b)

{

printf("In call back function, a:%d \t b:%d \n", a, b);

}

void copyFile(call\_back inform)

{

int a = 5;

int b = 7;

for (int i = 0; i < 10; i++)

{

inform(i\*a , b + i);

printf(" the function continue doing more work here...");

}

}

int main(int argc, char\* argv[])

{

copyFile(informProgress);

return 0;

}

הגדרנו typedef לפונקציה

והעברנו את שם הפונקציה כפרמטר

זו דרך מצוינת לבנות API .

נניח שאנחנו בונים תוכנית גדולה ואנחנו רוצים לספר על מצבה, אם יש שגיאות תוך כדי, לשלוח מידע כלשהוא

מאד נוח לבקש מה שנקרא registration callback בהתחלה

ואחר כך להשתמש בקריאה אליו.

שימו לב שאם אנחנו לא רוצים שיקראו ל callback אבל עדיין להגדיר אותו נוכל להעביר null ולבדוק

if (inform != NULL)

inform(i\*a, b + i);