Anna Jasielec

Podstawy programowania, grupa nr 4

Sprawozdanie z laboratorium nr 9

Cel laboratorium: Opanowanie podstaw wykorzystania wskaźników w C.

Przebieg zajęć:
Utworzyłam katalog roboczy lab_9 i skopiowałam ze strony przedmiotu plik wskaźniki_simple.c do podkatalogu simple. Zanalizowałam kod i uruchomiłam go. Staramy się unikać arytmetyki wskaźników, ponieważ zmiana adresu jest ściśle związana z typem obiektu, na który wskazuje wskaźnik. (Dodanie do wskaźnika liczby 2 nie spowoduje przesunięcia się w pamięci komputera o dwa bajty. Tak naprawdę przesuniemy się o dwukrotność rozmiaru typu zmiennej.) Przeprowadziłam modyfikacje na pliku.
Zmieniłam deklarację (prototyp) prosta_funkcja, tak aby mogła przyjmować dwa wskazane argumenty z funkcji main. Zmieniłam definicję funkcji, dostosowałam nagłówek do prototypu (dwa argumenty):

```
void prosta_funkcja( int *int_ptr, double *doublePtr )
```

Wydrukowanie wartości argumentów i zmiennych, na które wskazują:

```
Wewnątrz prostej funkcji:
 wartosc argumentow: int_ptr: 1, doublePtr: 3.140000
int_ptr wskazuje na: 140736443334788, doublePtr wskazuje na: 140736443334776
```

Zmieniłam wywołanie prosta_funkcja w funkcji main.

```
prosta_funkcja(wskaznik_do_int, double_p);
```

Zmodyfikowałam wartości wskazywanych zmiennych. Wydrukowanie wartości argumentów i zmiennych, na które wskazują:

```
Po zmianie wartosci zmiennych:
wartosc argumentow: int_ptr: 36, doublePtr: 9.140000
int_ptr wskazuje na: 140726820447252, doublePtr wskazuje na: 140726820447240
```

W funkcji prosta funkcja są drukowane inne wartości niż w funkcji main. Zmodyfikowałam wartości przesłanych argumentów. Wydrukowanie wartości argumentów i zmiennych, na które aktualnie wskazują:

```
Po zmianie wartosci argumentow
wartosc argumentow: int_ptr: -1605852200, doublePtr: 0.000000
int_ptr wskazuje na: 140727292562616, doublePtr wskazuje na: 140727292562264
```

Następnie skopiowałam ze strony przedmiotu pliku rownanie_kwadratowe. Zanalizowałam program i uruchomiłam go. Przeniosłam kod rozwiązania równania kwadratowego do osobnej funkcji. Stworzyłam nagłówek (prototyp) jako deklarację na początku plku i zdefiniowałam funkcję na końcu.

```
11  //$deklaracja funkcji
12  int rownanieKwadratowe(SCALAR a, SCALAR b, SCALAR c, SCALAR *x1_ptr, SCALAR *x2_ptr);
```

```
55  //$ definicja funkcji
56 ▼ int rownanieKwadratowe(SCALAR a, SCALAR b, SCALAR c, SCALAR *x1_ptr, SCALAR *x2_ptr) {
```

Funkcja jest typu int, ponieważ chcemy, aby zwracała jakąś wartość. Wywołanie funkcji w funkcji main bezpośrednio po wczytaniu współczynników równania.

```
rownanieKwadratowe(a, b, c, &x1, &x2);
```

Wydruk obliczonych wartości (także w funkcji main).

```
Program rozwiązywania równania kwadratowego ax^2 + bx + c = 0

Podaj parametr a: 1
Podaj parametr b: 4
Podaj parametr c: 2
Dwa pierwiastki rzeczywiste: x1 = -3.414213657379e+00, x2 = -5.857864618301e-01
```

Następnie na podstawie slajdów z wykładu napisałam program zawierający funkcję sortowania bąbelkowego. Stworzyłam tablicę o zadanym rozmiarze i wypełniłam ją losowymi wartościami.

```c
//$stworzenie tablicy o danym rozmiarze
printf("Jaki ma być rozmiar tablicy? ");
int rozmiar;
scanf("%d", &rozmiar);
int tab[rozmiar];

//$ wypełnienie tablicy losowymi wartościami
srand(time(NULL));
for(int i = 0; i < rozmiar; i++)
  {
    tab[i] = rand() % 1000 + 1;
  }
```

Zdefiniowałam funkcję zamien_wyrazy i zastosowałam ją wewnątrz funkcji sortowania.

```c
void zamien_wyrazy(int *int_1_p, int *int_2_p)
{
  int temp = *int_1_p;
  *int_1_p = *int_2_p;
  *int_2_p = temp;
}

void sortowanie_babelkowe(int *A, int n)
{
  int i, j;
  for (i = 0; i < n - 1; i++){
    for (j=0; j < n - 1 - i; j++){
      if (A[j] > A[j+1])
      {
        zamien_wyrazy(&A[j], &A[j + 1]);
      }
      }
    }
}
```

Przetestowałam działanie programu dla tablic o różnych rozmiarach.

Ostatnim krokiem była dalsza modyfikacja i analiza funkcji rownanie_kwadratowe i funkcji sortowanie_babelkowe.

Wnioski:

- *int *p* to deklaracja wskaźnika do wartości. Sam *p* to adres w pamięci, **p* to element na który wskazuje.
- Jeśli chcemy przydzielić pamięć do wskaźnika użyjemy *p = &a*.
- Staramy się unikać arytmetyki wskaźników. Dodanie do wskaźnika liczby 2 nie spowoduje przesunięcia się w pamięci komputera o dwa bajty tylko o dwukrotność rozmiaru typu zmiennej.
- Użycie wskaźników jako argumentów w funkcji pozwala na zmianę rzeczywistych wartości wskazywanych zmiennych.
- Przy wywołaniu funkcji, której argumenty to wskaźniki, należy przekazać do niej adresy zmiennych (poprzez operator &).
- Sortowanie bąbelkowe jest jednym z najprostszych rodzajów porządkowania danych. Jednak przy większej ilości danych program kompiluje się bardzo długo.