Fundusze Europejskie Wiedza Edukacja Rozwój
Rzeczpospolita Polska
AGH Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Unia Europejska Europejski Fundusz Społeczny

Zintegrowany Program Rozwoju
Akademii Górniczo-Hutniczej w Krakowie
Nr umowy: **POWR.03.05.00-00-Z307/17**

Instrukcja do ćwiczeń laboratoryjnych

Nazwa przedmiotu	Programowanie Imperatywne
Numer ćwiczenia	2
Temat ćwiczenia	Wskaźniki, tablice i zarządzanie pamięcią

Poziom studiów	I stopień
Kierunek	Informatyka
Forma i tryb studiów	Stacjonarne
Semestr	2

dr inż. Monika Dekster

Wydział Informatyki, Elektroniki i Telekomunikacji
Kraków, 2019

1 Cel ćwiczenia

Celem laboratorium jest praktyczne zastosowanie operacji na tablicach (statycznych i dynamicznych) i wskaźnikach. Studenci są zachęcani do projektowania proponowanych programów w stylu "klasycznym" (indeksowanie) oraz wskaźnikowym (wykorzystanie arytmetyki wskaźnikowej i operatora dereferencji).

Tego typu podejście powinno pomóc w zrozumieniu sposobu zarządzania pamięcią w języku C (stos i sterta), interpretacji indeksu jako przesunięcia od adresu bazowego (początku tablicy) i arytmetyki wskaźnikowej.

2 Wprowadzenie do ćwiczenia

Do wykonania ćwiczenia niezbędna jest podstawowa wiedza dotycząca organizacji pamięci w języku C oraz wykorzystania wskaźników, opisana np. w [1, 2].

3 Plan ćwiczenia

3.1 Proste operacje na tablicach statycznych

W tej części należy podkreślić, że tablice statyczne powinny być deklarowane ze stałym rozmiarem (stała literalna lub zdefiniowana dyrektywą `#define`). Wprawdzie standard C99 pozwala na definiowanie tablic ze zmiennym rozmiarem (Variable Size Arrays), jednak ten pomysł okazał sie przyczyną problemów i nie został wdrożony do C++.

Przykładowe programy do zaimplementowania na zajęciach:

1. Zadeklaruj dwie tablice całkowite, `t1` i `t2` o długości $N1$ i $N2$ i wypełnij je liczbami losowymi z przedziału [1, 10]

   (a) Napisz funkcję `void reverse()`, która odwraca jednowymiarową tablicę (pierwszy element staje się ostatnim, itd.),

   (b) Napisz funkcję sortująca tablicę jednowymiarową i zastosuj ją do posortowania tablic `t1` i `t2`,

   (c) Napisz i przetestuj funkcję `int one_two(int[], int, int[], int, int[])`, która przyjmuje jako argumenty dwie posortowane tablice liczb całkowitych i tworzy nową tablicę, której wartości będą pochodziły z przekazanych tablic, nie będą się powtarzały i będą posortowane rosnąco. Funkcja zwraca długość tablicy wynikowej.

2. Utwórz tablicę `T[N]`, wypełnij ją wartościami losowymi z przedziału $[-10, 10]$. Wykonaj statystykę procentowego pojawienia się każdej z wartości.

3. Zadeklaruj i zaincjalizuj tablicę dziesięciu liczb całkowitych i tablicę dziesięciu liczb typu `double`, zadeklaruj dwa wskaźniki jeden typu `int`, drugi `double`. Wypisz wartości i adresy wszystkich elementów obu tablic. Przypisz adresy pierwszych elementów tablic odpowiednim wskaźnikom i wypisz wartości i adresy wszystkich elementów tablic używając wskaźników.

4. Napisz funkcję, która wypisuje wartości tablicy t począwszy od elementu o indeksie k, będącym parametrem funkcji. Po wypisaniu elementu $t[n-1]$ należy wypisać kolejno elementy $t[0]$, $t[1]$, ..., $t[k-1]$. Kolejny parametr funkcji określa, czy wypisania należy dokonać "w przód" czy "w tył".

3.2 Tablice statyczne wielowymiarowe

W tej części studenci implementują proste programy wykorzystujące tablice wielowymiarowe. W szczególności należy podkreślic układ elementów takiej tablicy w pamięci, sposób wyliczania przesunięcia elementu o zadanych indeksach od adresu poczatku tablicy, przekazywanie tablic wielowymiarowych do funkcji (konieczność podania wszystkich, oprócz pierwszego, wymiarów tablicy).

1. W funkcji `main()` utwórz dwie tablice dwuwymiarowe, każda o rozmiarze [8][8], zainicjalizuj je wartościami losowymi. Napisz i przetestuj funkcje, która:

   (a) oblicza i wypisuje na ekran sumę elementów na przekątnej, sumę elementów w poszczególnych kolumnach i sumę wszystkich elementów,

   (b) mnoży dwie tablice kwadratowe przez siebie,

   (c) wypełnia tablicę kwadratową kolejnymi liczbami całkowitymi po spirali, zaczynajac np. od lewego górnego rogu,

3.3 Dynamiczna alokacja pamięci

Implementacja programów wykorzystujacych pamieć zaalokowaną dynamicznie. Studenci poznają funkcje służace do alokacji (`malloc, calloc, realloc`) i dealokacji (`free`). Należy podkreślić konieczność ręcznego zwalniania pamięci. Studenci powinni przetestować swoje programy używając narzędzia `valgrind`, które z powodzeniem wykrywa problemy z zarządzaniem pamiecią.

1. Napisz program, który wczytuje z klawiatury liczbę wierszy tablicy dwuwymiarowej oraz długości poszczególnych wierszy a następnie alokuje, wypełnia liczbami losowymi i wypisuje na ekran tablicę o podanej strukturze.

2. Napisz program, który na wejściu otrzymuje rozmiar macierzy kwadratowej, n, po którym następuje n wierszy, każdy z nich zawiera n liczb całkowitych. Program alokuje macierz, wczytuje wartości jej elementów i sprawdza, czy jest ona trójkątna dolna [3].

3. Napisz program, który alokuje obszar o wielkości $n \times m$ (n, m - dane wejściowe). Następnie stwórz i zainicjalizuj odpowiednią tablicę wskaźników tak, aby każdy wskaźnik wskazywał na początek kolejnego obszaru o długości m (będzie n takich obszarów, można je traktować jako kolejne wiersze tablicy 2D). Posługując się tą tablicą wypełnij kolejne wiersze losowymi wartościami i wydrukuj je.

3.4 Przykładowe programy

Przed wykonaniem ćwiczenia można zaznajomić się z przykładowymi programami [4]. Demonstrują one nie tylko sposób wykorzystania wskaźników ale także odpowiedni styl programowania.

4 Sposób oceny

W trakcie zajęć student powinien zaimplementować trzy zadane programy (po jednym z działów opisanych w punktach 3.1–3.3). Szczegółowe tematy zostaną podane i omówione podczas zajęć. W każdym przypadku oceniana będzie:

1. Poprawność programu.
2. Każdy program zostanie przetestowany z użyciem narzędzia `valgrind`. Programy wykazujące błędy nie zostaną przyjęte.
3. Poprawność stylu (można wzorować się na przykładowych programach [4]).

Za poprawne wykonanie:

1. Jednego programu – ocena 3.0
2. Dwóch programów – ocena 4.0
3. Trzech programów – ocena 5.0

Za częściowe lub wykazujące pewne braki rozwiązanie można uzyskać pół stopnia.

Literatura

[1] B.W. Kernighan, D.M. Ritchie: *Język ANSI C*, WNT Warszawa (2000), pp. 130–161.

[2] Prezentacja z wykładu: `https://www.icsr.agh.edu.pl/~mbargiel/teaching/jtp/sources/c-slides.php#point`

[3] Macierz trójkątna dolna: `https://pl.wikipedia.org/wiki/Macierz_tr%C3%B3jk%C4%85tna`.

[4] Przykładowe programy: `https://www.icsr.agh.edu.pl/~mbargiel/teaching/jtp/labs/lab02.php`