Anna Jasielec

Podstawy programowania, grupa nr 4

Sprawozdanie z laboratorium nr 6

Cel laboratorium: Opanowanie podstaw wykorzystania tablic i pętli w C.

Przebieg zajęć: Praca na pliku simple_loop.c:

1. Inicjowanie tablicy w momencie definiowania:

```c
// a. wszystkie wartości
    int tablica_int[ROZMIAR_TABLICY] = {12, 3, 5, 6, 4, 8, 23, 2, 122, 33};

// b. początkowe wartości
    int tablica_int[ROZMIAR_TABLICY] = {9, 88, 43, 6};
```

2. Nadanie wartości w pętli for oraz while:

```c
// a. zależnych od wartości zmiennej sterujacej
    for (int i=0; i<ROZMIAR_TABLICY; i++){
        tablica_int[i]=i*2;
    }

// b. wczytywanych z klawiatury za pomocą scanf
    printf("Podaj wartosci tablicy: ");

    for (int i=0; i<ROZMIAR_TABLICY; i++){
        scanf("%d", &tablica_int[i]);
    }
```

```c
//a.
 int k = 0;
while (k<ROZMIAR_TABLICY){
    tablica_int[k]=k*2;
    k++;
}

//b.
 int i = 0;
while (i<ROZMIAR_TABLICY){
    scanf("%d", &tablica_int[i]);
    i++;}
```

3. Stworzenie pętli do wypisującej wartości tablicy:

```c
 int j=0;

do {
    printf("Iteracja %d: tablica_int[%d] = %d\n", j, j, tablica_int[j]);
    j++;
}
while (j<ROZMIAR_TABLICY);
```

```
Iteracja 0: tablica_int[0] = 0
Iteracja 1: tablica_int[1] = 2
Iteracja 2: tablica_int[2] = 4
Iteracja 3: tablica_int[3] = 6
Iteracja 4: tablica_int[4] = 8
Iteracja 5: tablica_int[5] = 10
Iteracja 6: tablica_int[6] = 12
Iteracja 7: tablica_int[7] = 14
Iteracja 8: tablica_int[8] = 16
Iteracja 9: tablica_int[9] = 18
```

4. Użycie pętli for do wyszukiwania wartości maksymalnej w tablicy:

```c
int max=0;
for (int i = 0; i < ROZMIAR_TABLICY; i++){
    if (tablica_int[i]>max) {
        max = tablica_int[i];
    }
}
  printf("Wartosc maksymalna wynosi: %d\n", max);
```

```
Wartosc maksymalna wynosi: 18
```

5. Za pomocą pętli while obliczanie sumy elementów tablicy:

```c
int i = 0, suma = 0;
while (i < ROZMIAR_TABLICY){
    suma += tablica_int[i];
    i++;
}
printf("Suma elementow tablicy wynosi: %d\n", suma);
```

```
Suma elementow tablicy wynosi: 90
```

6. Zdefiniowanie tablicy znaków i nadanie jej losowych wartości z zakresu 33-126 za pomocą funkcji srand:

```
char tab_char [ROZMIAR_TABLICY];
```

```
 srand(time(NULL));

for (int i=0; i<ROZMIAR_TABLICY; i++){
  tab_char[i] = 32 + rand()%94+1;
}

 for(int i=0; i<ROZMIAR_TABLICY; i++){
  printf("Iteracja %d: tab_char[%d] = %c\n", i, i, tab_char[i]);
   }
```

```
Iteracja 0: tab_char[0] = N
Iteracja 1: tab_char[1] = a
Iteracja 2: tab_char[2] = Y
Iteracja 3: tab_char[3] = _
Iteracja 4: tab_char[4] = r
Iteracja 5: tab_char[5] = ]
Iteracja 6: tab_char[6] = `
Iteracja 7: tab_char[7] = 0
Iteracja 8: tab_char[8] = ,
Iteracja 9: tab_char[9] = %
```

- Obliczenie wartości średniej w tablicy i porównanie z teoretyczną średnią:

```
    float suma = 0;

 for (int i = 0; i < ROZMIAR_TABLICY; i++){
     suma += tab_char[i];
 }

 float srednia = suma/ROZMIAR_TABLICY;
 printf("Srednia elementow tablicy wynosi: %f\n", srednia); */
```

```
Srednia elementow tablicy wynosi: 82.699997
Srednia teoretyczna elementow tablicy wynosi: 79.500000
```

Im większy rozmiar tablicy, tym średnia jest bardziej przybliżona do średniej teoretycznej. Dla tablicy o rozmiarze 100000:

```
Srednia elementow tablicy wynosi: 79.532959
Srednia teoretyczna elementow tablicy wynosi: 79.500000
```

- wyszukiwanie zadanej wartości:

```
char szukana;
printf("Podaj szukany znak:");
scanf(" %c", &szukana);

_Bool znaleziono = 0;
for (int i = 0; i<ROZMIAR_TABLICY; i++) {
    if (tab_char[i] == szukana) {
        znaleziono = 1;
        printf("Znaleziono znak %c w %d elemencie tablicy.\n", szukana, i);
        break;
  }
}
if (!znaleziono) {
       printf("Nie znaleziono znaku %c.\n", szukana);
}
```

```
Podaj szukany znak:Z
Znaleziono znak Z w 2 elemencie tablicy.
```

Analiza i modyfikowanie pliku obliczPI.c.

- Napisanie własnego miniskryptu kompilującego uruchamiającego program oblicz_PI.c.

```
gcc -std=c99 oblicz_PI.c -lm
./a.out
```

- Rozważenie różnych typów pętli (for, while, do).
  Po zastosowaniu alternatywy – pętli for, program wykonuje się o wiele dłużej, ale wynik jest dużo dokładniejszy.
  Wyniki dla pętli for oraz do…while dla tej samej dokładności:

```
PI obliczone:    3.141571998596191, PI z biblioteki matematycznej:    3.141592653589793
Zalozona dokladnosc:    0.009999999776483, rzeczywisty blad:    0.000020654993602
```

```
PI obliczone:    3.079154253005981, PI z biblioteki matematycznej:    3.141592653589793
Zalozona dokladnosc:    0.009999999776483, rzeczywisty blad:    0.062438400583812
```

- Przerwanie obliczeń gdy kolejne wyrazy dodawane w pojedynczej iteracji osiągną wartość mniejszą od zadanej granicy bliskiej 0 oraz kiedy przybliżane wartości PI w kolejnych iteracjach przestaną się zmieniać. Kompilator wykonując pętlę `while (suma_minus > SMALL_NUMBER && suma_plus > SMALL_NUMBER)` w pewnym momencie zaczyna wyświetlać takie same wartości PI. Wtedy warunek 2 przerywa program.

Wnioski:

- Tablicy można nadać wartości od razu przy definiowaniu (np. int tab[3]={6,7,5}), w nawiasie kwadratowym jest rozmiar tablicy.
- Inicjacji można też wykonać poza definiowaniem (np. *tab[0]=5*), w nawiasie kwadratowym jest indeks.
- Do inicjacji można użyć pętli.
- Indeksy w tablicy zaczynają się od 0.
- Funkcja *srand* tworzy liczbę pseudolosową. Jak damy jej argument *time(NULL)* za każdym uruchomieniem programu zarodek liczb pseudolosowych będzie inny.
- Średnia liczb losowych z pewnego przedziału jest zawsze bliska średniej teoretycznej. Im większa tablica tym bliżej.
- Aby móc poprawnie kompilować pliki użyjemy komendy *gcc -std=c99 nazwa_pliku.c -lm.*
- Funkcja *ceil* zaokrągla liczby w górę, gdy w liczbie występuje część ułamkowa.
- Pisząc skomplikowane programy można używać różnych rodzajów pętli.
- Dogłębna analiza różnych wariantów umożliwia nam stworzenie optymalnego programu.