Sprawozdanie – tablice znaków

Tablice znaków w języku C pozwalają na przechowywanie zestawów danych w pamięci komputera w uporządkowany sposób. Tzn kolejne elementy tablicy są zapisywane w pamięci jeden po drugim. Umożliwia to łatwe operowanie na dużych zestawach danych. Pozwala również na zapisanie ciągów znaków czyli wyrazów lub zdań. Deklaracja: typ_przechowywanych_danych nazwa_tablicy [liczba_elementów]. Liczba elementów musi być wartością stałą tzn stałą liczbową lub zmienną const.

Zapis zmiennych do tablicy do momentu napotkania EOF lub \n:

```
while (((magicarray1[i1] = getchar()) != EOF) && ck != '\n') {i1++;}
wartość (i1 + 1) to liczba elementów w tablicy
```

Rozpoznawanie wczytywanych zmiennych (z modyfikacją na if … else if … else). Zmienna c to char, wg tablicy ASCII wartość zera to 48 więc jeżeli chcemy traktować c jako liczbę należy każdorazowo odejmować 48 lub ‘0’. Wartości odpowiednich komórek w 10 elementowej tablicy ndigit wskazuje ile razy dana cyfra została wprowadzona.

```
while (c!='\n') {
        scanf("%c", &c);
        magicarray2[i2] = c;
        if (0 <= (c-48) && (c-48) <= 9)
            ndigit[c-48]++;
        else if (c == ' ' || c == '\n' || c == '\t')
            nwhite++;
        else
            nother++;
        i2++;}
```

Sprawdzenie czy dwa napisy są identyczne: dla optymalizacji wpierw sprawdzamy długość. Następnie sprawdzamy każdy kolejny element, jeżeli będą się różnić to wychodzimy z pętli.

```
   bool czy_rowne = false;
    if (i1 ==i2)
    {
        czy_rowne = true;
        for (int i = 0; i <= i1; i++)
        {
            if (magicarray1[i] != magicarray2[i])
             {
                czy_rowne = false;
               break; }}}
```

Zamiana kolejności elementów tablicy: pętla wykona się i2/2 razy. Jeżeli i2 jest parzyste to należy wykonać i2/2 podmian. Jeżeli nieparzyste to i2/2 – 0.5 (i2 to int więc ((i2/2) – 0.5) == i/2, dla nieparzystego i2), ponieważ element środkowy pozostanie taki sam.

```
for(int i = 0; i < (i2 / 2); i++){
        char pom = magicarray2[i];
        magicarray2[i] = magicarray2[i2 - i];
        magicarray2[i2 - i] = pom;}
```

Zamiana elementów z małych liter na wielkie: wystarczy jedynie zmniejszyć wartości o 32, co wynika z tablicy ASCII, gdzie ‘A’ jest o 32 elementy przed ‘a’.

```
for (int i = 0; i < i1; i++)
        magicarray1[i] -= 32;
```

Histogram – tworzymy dwie tablice, jedna przechowuje liczby losowe, duga jest wypełniona zerami. Za każdym przypisaniem liczby losowej inkremetujemy komórkę o indeksie wylosowanej liczby w histogramie.

```
    srand(time(NULL));
    int randArray[length1], histogram[length1] = {0};
    for (int i = 0; i < length1; i++){ // zapis
        randArray[i] = rand() % 10;
        histogram[randArray[i]]++;}
    for (int i = 0; i < length1; i++){ // wypisywanie
        if (histogram[i] != 0)
            printf("%d :", i);
        for (int j = 0; j < histogram[i]; j++)
            printf(" *");
        if (histogram[i] != 0)
            printf("\n");
    }
```