
Zajęcia 6 – wskaźniki i tablice dynamiczne

1. Napisz funkcję `void minMax(int tab[], int w, int* wmin, int* wmax)`, która szuka w podanym ciągu najmniejszego i największego elementu. Znalezione wartości mają zostać zapisane w zmiennych wskazywanych przez parametry (wskaźniki) `wmin` oraz `wmax`.

Parametr `dl` określa liczbę elementów w ciągu (tablicy) `tab`.

Dla powyższej funkcji proszę odpowiednio uzupełnić główną część programu:

```
int main(){
    int t[] = {3, -4, 1, 0, 10, 5};
    int min, max;

    minMax(t, sizeof(t) / sizeof(t[0]), &min, &max);
    cout << "Najmniejszy i największy element ciągu, to " << min
         << ", " << max << endl;

    return 0;
}
```

2. Zdefiniuj funkcję:

```
void strsl(char s1[], char s2[], char* &dluzszy, char* &krotszy),
```

która przyjmuje jako parametry dwa łańcuchy (`s1` oraz `s2`) i dwa wskaźniki (`dluzszy`, `krotszy`). Funkcja ta ma przypisać wskaźnikom `dluzszy` i `krotszy` odpowiednio adres dłuższego oraz krótszego z podanych łańcuchów. W przypadku, gdy są one równej długości to wskaźnik `dluzszy` ma wskazywać na `s1`, natomiast wskaźnik `krotszy` na łańcuch `s2`.

Dla powyższej funkcji proszę odpowiednio uzupełnić główną część programu:

```
int main() {
    char s1[100];
    char s2[100];
    cout << "Podaj pierwszy ciag: ";
    cin >> s1;
    cout << "Podaj drugi ciag: ";
```

```

    cin >> s2;

    char* krotszy;
    char* dluzszy;
    strls(s1, s2, dluzszy, krotszy);

    cout << "Dluzszy z podanych lancuchow to: " << dluzszy << endl
          << "Krotszy z podanych lancuchow to: " << krotszy << endl;

    return 0;
}

```

3. Napisz funkcję `void ptradv(int* wsk, int n)`, która wyświetla adres przechowywany we wskaźniku `wsk` oraz wartość zmiennej przez niego wskazywanej. W kolejnym kroku funkcja przesuwa wskaźnik (`wsk += 1`) i powtarza powyższe czynności tyle razy ile wynosi wartość parametru `n`.

Przykład:

```

Adres: 0x12312320, wartość: 12
Adres: 0x12312324, wartość: -123

```

Dla powyższej funkcji proszę odpowiednio uzupełnić główną część programu:

```

int main() {
    int t[] = { 1, 3, -6, 4};
    int* z;
    ptradv(t, 4);
    z = t;
    ptradv(z, 4);

    return 0;
}

```

4. Napisz funkcję `int * odwroc(int *tab, int rozmiar)`, która tworzy kopię podanej tablicy, ale z elementami w odwrotnym porządku. Wynikiem funkcji jest adres utworzonej tablicy.

Przykład programu korzystającego z tej funkcji:

```

int main() {
    int liczby [] = { 1, 2, 3, 4 };
    int *odwrotnie = odwroc(liczby, 4);
    for (int i = 0; i < 4; ++i) {
        cout << odwrotnie[i] << " ";
    } // powinno wyświetlić: 4 3 2 1
}

```

```

    delete [] odwrotnie; // zwolnienie przydzielonej pamięci
    return 0;
}

```

5. Napisz funkcję `void tabliczka(int r)`, która tworzy tabliczkę mnożenia o wielkości $r \times r$. Następnie w pętli wyświetla wynik mnożenia wskazanych przez użytkownika liczb (dopóty, dopóki użytkownik nie zdecyduje, że już koniec). Następnie wyświetlona zostaje cała tabliczka mnożenia.

Uwaga: Tabliczka mnożenia ma zostać zapisana w dwuwymiarowej tablicy dynamicznej, proszę więc pamiętać o przydziale i zwolnieniu pamięci.

Dla powyższej funkcji proszę odpowiednio uzupełnić główną część programu:

```

int main() {
    int i;
    cout<<"Podaj wielkość tabliczki mnożenia: ";
    cin>>i;
    tabliczka(i);

    return 0;
}

```

6. Napisz funkcję `string * podziel_na_slowa(char tekst[], int &ile_slow)`, której zadaniem jest podzielenie ciągu `tekst` na słowa (słowo to niepusty ciąg znaków nie zawierający białego znaku). Wynikiem funkcji jest dynamiczna tablica słów (każde typu `string`), której długość przekazywana jest przez parametr (referencję) `&ile_slow`.

Przykładowy program korzystający z funkcji:

```

int main() {
    char s[] = "  Ala ma kota. \nKot ma mysz.  ";
    int ile_slow;
    string * slowa = podziel_na_slowa(s, ile_slow);
    for (int i = 0; i < ile_slow; ++i) {
        cout << i << " " << slowa[i] << endl;
    }
    delete [] slowa;
    return 0;
}

```

Wydruk uzyskany w wyniku jego wykonania:

```

0 Ala
1 ma

```