Zajęcia 5 – wskaźniki i tablice dynamiczne

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

   Parametr `size` określa liczbę elementów w tablicy `tab`.

   Przykładowy program korzystający z funkcji `minMax`:

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

       minMax(t, 6, &smallest, &largest);
       cout << "Najmniejszy: " << smallest
            << "\nNajwiekszy: " << largest << endl;
       return 0;
   }
   ```

2. Napisz funkcję `int* copy_rev(const int values[], const int size)`, zwracającą nowo utworzoną tablicę liczb, w której kolejne wartości odpowiadają elementom tablicy `values`, ale w *odwrotnej* kolejności. Rozmiar tablicy określa parametr `size`.

   Przykładowy program korzystający z funkcji `copy_rev`:

   ```
   int main() {
       const int seq_len = 5;
       int seq[seq_len] = { 1, 6, 7, 5, 3 };
       int *seq_rev = copy_rev(seq, seq_len);
       cout << "Ciag odwrotnie: ";
       for (int i = 0; i < seq_len; ++i) {
           cout << seq_rev[i] << " ";
       }
       delete [] seq_rev;
       seq_rev = nullptr;
   ```

```
    return 0;
}
```

Wydruk dla programu:

```
Ciag odwrotnie: 3 5 7 6 1
```

3. Napisz funkcję `std::string* get_words(std::string text, int &num_words)`, która w podanym ciągu `text` znajduje wszystkie *wyrazy*. Przez wyraz rozumiemy dowolny ciąg nie zawierający białych znaków (spacja, tabulator, itd.). Liczba słów zwracana jest przez parametr `num_words`. Wynikiem działania jest wskaźnik do nowo utworzonej tablicy z kopiami znalezionych słów.

   Wskazówki.

   - Do sprawdzenia, czy dany znak jest biały można użyć funkcji `std::isspace`.
   - Długość łańcucha `text` można odczytać za pomocą `text.size()`.
   - W pierwszej kolejności należy znaleźć liczbę słów, co pozwoli na utworzenie tablicy na wynikowe słowa.

   Przykładowy program korzystający z funkcji:

```
int main() {
    int num_words;
    string *words = get_words("  Katowice\tul.\nBankowa   12  ", num_words);
    cout << "Lista slow:\n";
    if (num_words > 0) {
        for (int i = 0; i < num_words; ++i) {
            cout << i << ". " << words[i] << '\n';
        }
        delete [] words;
        words = nullptr;
    }
    return 0;
}
```

   Wynik działania:

```
Lista slow:
0. Katowice
1. ul.
2. Bankowa
3. 12
```

4. Napisz funkcję `void append(int* &seq, int &capacity, int &size, const int el)`, która zapisuje w tablicy `seq` podaną wartość `el` na kolejnej *wolnej* pozycji.

   - Przed zapisaniem elementu należy upewnić się, że jest dla niego miejsce — czyli czy aktualna liczba elementów, `size`, jest mniejsza od pojemności tablicy, `capacity`. Jeżeli tak, to można dodać element inkrementując wartość `size`.
   - W przypadku gdy aktualna liczba elementów jest równa pojemności tablicy, to jest ona *pełna* i należy najpierw utworzyć *nową* tablicę o *dwukrotnie* większej pojemności, i przepisać do niej zawartość `seq`.
   - Następnie należy *zwolnić* pamięć dla „starej" tablicy i przypisać `seq` adres nowo utworzonej tablicy.
   - Należy również pamiętać o zapisaniu nowej pojemności do parametru `capacity`.

   Przykład użycia funkcji demonstruje program:

```
int main() {
    int size = 0;
    int capacity = 2;
    int* results = new int[capacity];
    srand(42);
    int r = rand() % 100 + 1;
    while (r % 17 != 0) {
        append(results, capacity, size, r);
        r = rand() % 100 + 1;
    }

    cout << "Wyniki:";
    for (int i = 0; i < size; ++i) {
        cout << results[i] << " ";
    }
    cout << "\nRozmiar: " << size << "\tpojemnosc: " << capacity << endl;

    delete [] results;
    results = nullptr;

    return 0;
}
```

   Wygenerowany wydruk na konsoli:

```
Wyniki:67 41 82 42 13 59 22 41 36 44 75 44 18 5 97 63 93 49 99 60
Rozmiar: 20 pojemnosc: 32
```