Wstęp do programowania w języku C

Programowanie z użyciem bibliotecznych struktur danych. Przykłady w C i C++ dla uporządkowanych drzew binarnych. Od C do C++.

Marek Piotrów - Wykład 13

20 stycznia 2022

glibc-tree-sort.c I

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <glib.h>

// You must install the following packages: libglib2.0-dev and pkg-config (ubuntu/debian)
// or equivalent ones (other distributions).
// Compile with:
// gcc -std=c11 -Wall -Wextra glibc-tree-sort.c -o gtree-sort `pkg-config glib-2.0 --cflags --libs`

#define MAXLINE 1000

static int str_cmp(const gconstpointer a, gconstpointer b);
static int print_str(gpointer key, gpointer value, gpointer data);

/*****************************************************************************/

// Read strings from stdin and print them in order

int main(void)
{
   GTree *StrTreeRoot = g_tree_new(str_cmp);
   char line[MAXLINE];

   // Read lines into GTree headed by StrTreeRoot
```

glibc-tree-sort.c II

```
    while ((fgets(line, sizeof(line), stdin)) != NULL)
        g_tree_insert(StrTreeRoot, g_string_new(line), NULL);

    if (!(feof(stdin))) {
        fprintf(stderr, "error reading from stdin\n");
        exit(2);
    }

    // Print lines in the lexicographical order

    g_tree_foreach(StrTreeRoot, print_str, NULL);

    return 0;
}


/*****************************************************************************/

static int str_cmp(const gconstpointer a, gconstpointer b)
{
    const GString *aStr = a, *bStr = b;
    int r = strcmp(aStr->str, bStr->str);
    return r == 0 ? 1 : r;  // force duplicate strings to be stored
}
```

glibc-tree-sort.c III

```c
static int print_str(gpointer key, gpointer value, gpointer data)
{
    if (value == NULL && data == NULL)
        printf("%s", ((GString *)key)->str);
    return 0;
}
```

bsd-tree-sort.c I

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <bsd/sys/tree.h>
// You must install the libbsd-dev package (ubuntu/debian) or equivalent (other)
// to compile this example. Compile with:
// gcc -std=c99 -Wall -Wextra bsd-tree-sort.c -o btree-sort

#define MAXLINE 1000

/******************************************************************************/

typedef struct StrTreeNode {
    RB_ENTRY(StrTreeNode) links;
    char str[];
} StrTreeNode;

static int StrTN_cmp(StrTreeNode *a, StrTreeNode *b) {
    int r = strcmp(a->str, b->str);
    return r == 0 ? 1 : r;  // force duplicate strings to be stored
}

RB_HEAD(StrTree, StrTreeNode);
RB_PROTOTYPE(StrTree, StrTreeNode, links, StrTN_cmp);
```

bsd-tree-sort.c II

```
/**************************************************************************/

// Read strings from stdin and print them in order

int main(void)
{
    struct StrTree  StrTreeRoot;
    StrTreeNode *sln;
    char line[MAXLINE];

    RB_INIT(&StrTreeRoot);

    // Read lines into StrTree headed by StrTreeRoot
    while ((fgets(line, sizeof(line), stdin)) != NULL) {
      if ((sln = malloc(sizeof(*sln) + strlen(line) + 1)) == NULL) {
          fprintf(stderr,"memory allocation error\n");
          exit(1);
        }
        strcpy(sln->str, line);
        RB_INSERT(StrTree, &StrTreeRoot, sln);
    }
    if (!(feof(stdin))) {
        fprintf(stderr, "error reading from stdin\n");
        exit(2);
```

bsd-tree-sort.c III

```
    }

    // Print lines in the lexicographical order

    //for (sln = RB_MIN(StrTree, &StrTreeRoot); sln != NULL; sln = RB_NEXT(StrTree, &StrTreeRoot, sln))
    RB_FOREACH(sln, StrTree, &StrTreeRoot)
        printf("%s", sln->str);
    return 0;
}

RB_GENERATE (StrTree, StrTreeNode, links, StrTN_cmp);
```

stl-tree-sort.cpp

```
#include <iostream>
#include <string>
#include <set>

using namespace std;

// Read strings from stdin and print them in order

int main(void)
{
    set<string,less_equal<string>> tree;
    string line;

    while (getline(cin,line))
        tree.insert(line);

    if (cin.bad()) {
        cerr << "error reading from stdin" << endl;
        exit(2);
    }

    // Print lines in the lexicographical order
    for (auto iter = tree.begin(); iter != tree.end(); iter++)
        cout << *iter << endl;

    return 0;
}
```

Krótka historia języka C++ - przypomnienie

- 1979 - pierwsza wersja języka C++ stworzona przez B. Stroustrup'a jako obiektowe rozszerzenie języka C (pod wpływem języka Simula);
  kompilator **Cfront** tłumaczył kod z C++ na C;
- 1985 - opis języka C++ przez B. Stroustrup'a;
- 1998 - opublikowanie standardu języka C++:
  ISO/IEC 14882:1998 (c++98),
- 2003, 2011, 2014, 2017 - kolejne standardy języka C++:
  ISO/IEC 14882:2003 (c++03), ISO/IEC 14882:2011 (c++11),
  ISO/IEC 14882:2014 (c++14) i ISO/IEC 14882:2017 (c++17).

Podstawowe właściwości języka C++

- nie jest (podobnie jak C) własnością żadnej osoby czy instytucji;
- jest językiem wieloparadygmatowym (można w nim programować przede wszystkim obiektowo, ale też generycznie i proceduralnie);
- próbuje zachować jak największą zgodność na poziomie kodu źródłowego i łączenia modułów skompilowanych z językiem C;
- zakłada statyczną kontrolę typów (podobnie jak C);
- umożliwia programiście bezpośrednie zarządzanie pamięcią (podobnie jak C);
- zakłada, że wydajność programów napisanych w C++ powinna być porównywalna z odpowiednimi napisanymi w C.

Pierwszy przykład I

```
#include <iostream>
#include <fstream>
#include <cstdlib>

using namespace std;

#define losuj(n) (n < 1? 0 : rand()%(n))

inline void zamien (int &p, int &q)
{
   int i=p ;
   p=q, q=i ;
}

void permutacja (int rozm, int *tab)
{
   for (int i=0; i < rozm; i++)
      tab[i]=i;
   for (int j=7; j > 0; j--)
      for (int i=0; i < rozm; i++)
         zamien(tab[i],tab[losuj(i+1)]) ;
}

void drukuj (int rozm, int *tab)
{
```

Pierwszy przykład II

```
    cout<<"Losowa permutacja: ";
    for (int i=0; i < rozm; i++)
        cout<<(i > 0 ? ',' :'[')<<tab[i];
    cout<<']'<<endl;
}

int main(void)
{
    const int rozmiar=10 ;
    int tab[rozmiar] ;

    permutacja(rozmiar,tab);
    drukuj(rozmiar,tab);

    char nazwa[128];
    cout<<"podaj nazwe pliku: ";
    cin>>nazwa;

    ofstream plik;
    plik.open(nazwa,ios::out|ios::binary);
    plik<<rozmiar<<": ";
    for (int i=0; i < rozmiar; i++)
        plik<<tab[i]<<' ';
    plik<<endl;
```

Pierwszy przykład III

```
    plik.close();
    return 0;
}
```

Czego nie wolno w C++ (a można w C)

1. Przypisywać bez rzutowania wskaźnika typu `void *` (na przykład wartości zwracanej przez malloc).
2. Używać nazwy funkcji bez jej wcześniejszej deklaracji lub definicji (standard c99 też tego wymaga, ale kompilator gcc generuje w takiej sytuacji ostrzeżenie, a nie błąd).
3. Używać tablic o zmiennej liczbie elementów, obliczeń na liczbach zespolonych, inicjalizatorów desygnowanych czy stałych strukturalnych (które wprowadził standard C99).

Nowe elementy w C++

1. Klasy i obiekty, dziedziczenie.
2. Mechanizmy hermetyzacji (sekcje prywatne, chronione i publiczne).
3. Obiektowe operacje wejścia/wyjścia.
4. Obsługa wyjątków;
5. Operatory `new` i `delete`.
6. Przestrzenie nazw i operator zakresu `::` .
7. Zmienne i parametry referencyjne.
8. Funkcje i operatory przeciążone.
9. Argumenty domniemane.
10. Domyślne definicje funkcji otwartych w klasach.