Laboratorium Informatyki
Programowanie w języku C
Ćwiczenie 5

1 Wstęp

W języku C poza dostępnymi typami danych (`int`, `float`, itd) programista może zdefiniować własne złożone typy danych - **struktury** i **unie**. Typy te są budowany za pomocą dostępnych typów danych zwykłych i tablicowych.

Dane przechowywane w zmiennych są ulotne. Tracone są po zakończeniu działania programu. Zapis danych do pliku i ich odczytanie przy ponownym uruchomieniu programu pozwala na zachowanie ciągłości pracy. W języku C dane można zapisywać i odczytywać z pliku tekstowego i binarnego.

2 Złożone typy danych

Złożone typy danych mogą być budowane w języku C w oparciu o podstawowe i złożone typy danych. Mogą zawierać w sobie elementy tablicowe, jak i same być tablicami. Korzystanie z definiowalnych typów złożonych wymaga w pierwszym kroku zdefiniowania danego typu. W kolejnym można deklarować zmienne na podstawie tak utworzonego typu danych. W języku C można zdefiniować dwa rodzaje typów złożonych, struktury i unie.

2.1 Struktura

Struktura jest złożonym deklaratywnym typem danych określanym przez programistę w kodzie programu. Składnia struktury definiowana jest na podstawie dostępnych typów danych. Deklarację struktury rozpoczyna się od słowa kluczowego `struct`, następnie podaje się nazwę danej struktury i określa elementy struktury. Elementami struktury są zmienne zdefiniowane z wykorzystaniem dostępnych typów. Deklaracja struktury tak jak i zmiennej kończy się średnikiem. Równocześnie z deklaracją typu strukturalnego można na jego bazie utworzyć zmienną strukturalną. Strukturę w swojej budowie można porównać do tabeli przechowującej dane różnych typów.

```
struct nazwa{
   typ nazwa1;
   typ nazwa2[45];
   typ *nazwa3;
   struct nazwa_S nazwa_Z;
};
struct nazwa zmiennaStruct;
struct nazwa zmiennaSTab[20];
```

Zmienne strukturalne mogą być zwracane przez funkcję (o ile nie są tablicą). Jeżeli tablicowa zmienna strukturalna ma być zwracana przez funkcję należy problem rozwiązać analogicznie jak w przypadku zwykłej tablicy. Typ strukturalny można deklarować jako globalny, dzięki temu zmienne strukturalne na jego bazie można będzie tworzyć (inicjować) w każdym miejscu kodu programu. Zmienne strukturalne można alokować dynamicznie analogicznie jak zwykłe zmienne.

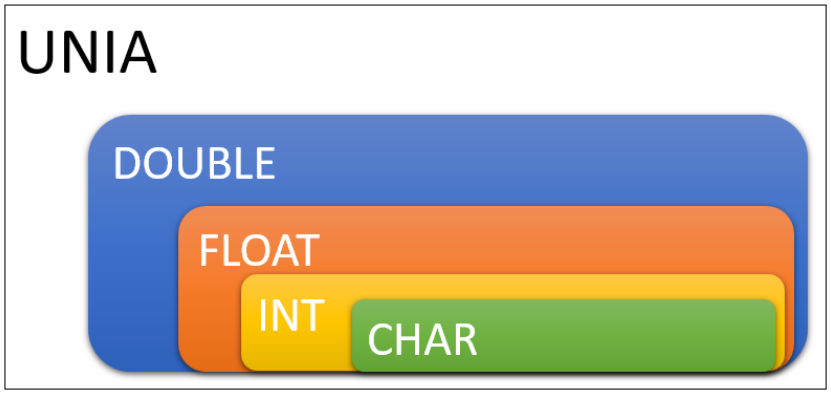
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

struct los{
   int p,k;
   int dokl;
   float liczba;
};

struct los dane();
struct los losowanie(struct los);

int main()
{
   srand(time(NULL));
   printf("Program losuje liczbe rzeczywista w zadanym
               przedziale z okreslona dokladnoscia.\n");
   struct los X;
   X = dane();
   for(int i=0;i<30;i++){
       X = losowanie(X);
       printf("Wylosowano liczbe: %.*f\n",X.dokl,X.liczba);
   }
}
```

```
struct los losowanie(struct los A)
{
   int e = pow(10,A.dokl);
   int w=1;
   if(A.dokl!=0) w=A.dokl;
   long long l = A.p*e+(long long)(pow(rand(),w))%
                              ((int)(A.k*e-A.p*e+1));
   A.liczba = (float)l*pow(10,-A.dokl);
   return A;
}
struct los dane()
{
   struct los A;
   printf("Podaj poczatek przedzialu: ");
      scanf("%i",&A.p);
   printf("Podaj koniec przedzialu: ");
      scanf("%i",&A.k);
   printf("Okresl dokladnosc generowania (1-3): ");
      scanf("%i",&A.dokl);
   return A;
}
```

Przykładowy program losuje liczbę rzeczywistą z określoną przez użytkownika dokładnością. Program dane i parametry losowania przechowuje w zmiennej strukturalnej `los`.

2.2 Unia

Deklaratywny typ danych typu unia jest tworzony i stosowany w sposób analogiczny do struktury. Rożnica polega na sposobie wykorzystywania elementów unii.



Rysunek 1: Organizacja elementów unii w pamięci

Specyfika unii jest organizacja elementów unii w pamięci i realizacja dostępu do nich. Zmienna typu unii pozwala w danej chwili korzystać tylko z jednego z zdefiniowanych elementów. W unii wszystkie jej elementy współdzielą ten sam obszar w pamięci o rozmiarze największego elementu unii, jak pokazano na rysunku powyżej.

Typ unia nie zapewnia żadnych narzędzi identyfikującej aktywny element unii. Pisząc kod należny o tym pamiętać i kontrolować którego z elementów aktualnie używamy. Zapisanie danych do kolejnego elementu unii powoduje, że tracone sa informacje zapisane do innego elementu unii.

```
#include <stdio.h>
#include <stdlib.h>
union dane{
   int x;
   float y;
   char c[50];
};
void kontrola(union dane);
int main()
{
   union dane nowe;
   printf("Podaj ciag znakowy: ");
   gets(nowe.c);          kontrola(nowe);
   printf("Podaj liczbe calkowita: ");
   scanf("%i",&nowe.x);   kontrola(nowe);
   printf("Podaj liczbe rzeczywista : ");
   scanf("%f",&nowe.y);   kontrola(nowe);
}
void kontrola(union dane U)
{
   printf("Kontrola danych unii:\n");
   printf("Zmienna int: %i\n",U.x);
   printf("Zmienna float: %f\n",U.y);
   printf("Zmienna char[]: %s\n",U.c);
}
```

3 Zapis danych do pliku

W języku C dane do pliku można zapisywać i odczytywać w trybie tekstowym lub binarnym. Obydwa pozwalają w prosty sposób archiwizować dane. Plik jest pewnym zbiorem danych, zapisanym w systemie plików na nośniku danych. Każdy plik ma skończoną długość, a informacja w nim zapisana jest ciągiem zer i jedynek. Dane można zapisywać w:

- **pliku tekstowym** - poszczególne bajty w pliku można zinterpretować jako dane alfanumeryczne (znaki), zapisane przy pomocy określonego kodowania (np. ASCII).
- **pliku binarnym** - poszczególne bajty w pliku mają dowolne wartości, niekoniecznie są interpretowalne jako znaki alfanumeryczne.

3.1 Plik tekstowy

Zapis do pliku tekstowego realizowany jest poprzez konwersję danych do formatu znakowego. Dane zapisywane są w pliku tekstowym otwartym tekstem. zapis i odczyt danych z pliku tekstowego realizowany jest analogicznie jak korzystanie ze standardowego wejścia i wyjścia (ekran i klawiatura). Komunikacja odbywa się w sposób sekwencyjny.

- Otwarcie dostępu do pliku w określonym trybie
- Zapis-odczyt danych z pliku
- Zamkniecie dostępu do pliku

Obsługa pliku w trybie tekstowym jest realizowana jako domyślna. Powiązanie pomiędzy fizycznym plikiem na dysku a portem wejścia/wyjścia realizowane jest przez zmienną typu `FILE` zdefiniowaną w bibliotece `<stdio.h>`. Do zadeklarowanego wskaźnika typu `FILE` można wielokrotnie w kodzie programu podpiąć i odłączyć plik(pliki) tekstowe. Podpięcie do pliku realizowane jest poleceniem `fopen()`, a odłączenie `fclose()`. Dostęp do pliku może zostać otwarty w jednym z trzech podstawowych trybów:

- **w** - zapisy - jeżeli plik istnieje jego zawartość zostanie skasowana, jeżeli pliku nie ma to zostanie utworzony
- **a** - zapis - jeżeli plik istnieje nowe dane zostaną dopisane na końcu pliku, jeżeli pliku nie ma to zostanie utworzony
- **r** - odczytu - dane zostaną odczytane od początku pliku.

Funkcja `fopen()` dodatkowo zwraca także informacje (`true\false` o statusie otwarcia dostępu do pliku. W połączeniu z instrukcją warunkową `if()` można zaprogramować w kodzie reakcję programu na problemy z dostępem do pliku (brak pliku, brak prawa do zapisu itp).

Zapis danych do pliku tekstowego może być zrealizowany jedną z poniższych funkcji:

- `putc('znak',plik);` - do pliku zostanie zapisany pojedynczy znak
- `fprintf(plik,"String",zmienne...);` - do pliku zostanie zapisany sformatowany ciąg tekstowy Śtring"
- `fputs(bufor,plik);` - do pliku zostanie zapisana zawartośc tablicy znakowej "bufor"

Odczyt realizowany jest podobnie jak odczyt z bufora wejściowego, za pomocą dedykowanych funkcji:

- `zmienna=getc(plik);` - do wskazanej zmiennej przypisywany jest pojedynczy znak pobrany z pliku
- `fscanf(plik,"%i",&zmienna);` - odczytanie sformatowanego ciągu znaków z pliku tekstowego i przypisanie danych wyizolowanych z niego do wskazanych zmiennych
- `fgets(bufor,limit,plik);` - pobranie z pliku ciągu znakowego o określonej długości i przypisanie go do tablicy znakowej

Podczas odczytywania danych istotne jest określenie momentu kiedy osiągnięty zostanie koniec pliku. W języku C może to być zrealizowane na dwa sposoby:

- za pośrednictwem zdefiniowanej w bibliotece `<stdio.h>` stałej `EOF`. Odczytując z pliku dane znak po znaku należy sprawdzać czy odczytany znak jest znakiem końca pliku.
- za pomocą funkcji `feof()` można określić, niezależnie od sposobu odczytu danych czy osiągnięto już koniec pliku.

```
#include <stdio.h>
#include <stdlib.h>
FILE *plik;
int main()
{
    printf("Podaj nazwe pliku: ");
    char nazwa[30];
    gets(nazwa);
    if(plik = fopen(nazwa, "w"))
    {
        printf("Podaj dane do zapisania: ");
        char znak;
        do{
            znak = getchar();
            putc(znak,plik);
        }while(znak!='\n');
        fclose(plik);
        printf("Zapisano dane!\n");
    }else{
        printf("Problem z dostepem do pliku!\n");
    }
    printf("Odczytanie danych: ");
    if(plik = fopen(nazwa, "r"))
    {
        char dane[10];
        while(!feof(plik)){
            fgets(dane,10,plik);
            printf("%s",dane);
```

```
            dane[0]='\0';
        }
        fclose(plik);
        printf("Koniec odczytu danych.\n");
    }
    else{
        printf("Nie mozna otworzyc pliku do odczytu!\n");
    }
}
```

3.2 Plik binarny

W odróżnieniu od pliku tekstowego dane w pliku binarnym zapisywane są jako surowe dane (bajty) w sposób analogiczny do sposobu w jaki dane zapisywane są w pamięci. Dostęp do pliku jest realizowany znacznie szybciej jak do pliku tekstowego. Z plików binarnych warto korzysta przy zapisywaniu dużych partii danych. Aby odczytać dane należy znać sposób zapisu tych danych. Po otwarciu pliku w edytorze tekstowym widać tak zwane 'krzaczki".

Otwarcie dostępu do pliku w trybie binarnym wymaga ustawienia flagi `b` i flagi właściwej dla wybranego trybu dostępu (zapisanie, dopisywanie i odczytywanie). W trybie binarnym zapis realizowany jest funkcją `fwrite()`, a odczyt za pomocą funkcji `fread()`.

```
fwrite(void *tab, sizeof(void), int item, FILE plik);
fread(void *tab, sizeof(void), int item, FILE plik);
```

Za ich pomocą można zapisywać pojedyncze dane jak i tablice wybranego typu. Poniżej przedstawiono przykład zapisu i odczytu danych tablicowych.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int **tab;
void zapis(int,int);
void odczyt();
void print(int,int);


int main()
{
    srand(time(NULL));
    int a,b;
    printf("Podaj rozmiar tablicy:\n");
```

```
    printf("Liczba wierszy: ");
    scanf("%i",&a);
    printf("Liczba kolumn: ");
    scanf("%i",&b);
    tab = (int**)malloc(a*sizeof(int));
    for(int i=0;i<a;i++){
        tab[i] = (int*)malloc(b*sizeof(int));
        for (int j=0;j<b;j++)
            tab[i][j] = 0+rand()%256;
    }
    printf("Wylosowano:\n");
    print(a,b);
    zapis(a,b);

    for(int i=0;i<a;i++)
        free(tab[i]);
    free(tab);

    printf("Odczytano dane z pliku binarnego:\n");
    odczyt();
}
void print(int a, int b)
{
    for(int i=0;i<a;i++){
        for (int j=0;j<b;j++)
            printf("%c ",tab[i][j]);
        printf("\n");
    }
}
void zapis(int a, int b)
{
    FILE *plik = fopen("tabela.bin","wb");
    fwrite(&a,sizeof(int),1,plik);
    fwrite(&b,sizeof(int),1,plik);
    for(int i=0;i<a;i++)
        fwrite(tab[i],sizeof(int),b,plik);
    fclose(plik);
}
void odczyt()
{
    FILE *plik = fopen("tabela.bin","rb");
    int a,b;
    fread(&a,sizeof(int),1,plik);
```

```
    fread(&b,sizeof(int),1,plik);
    tab = (int**)malloc(a*sizeof(int));
    for(int i=0;i<a;i++)
        tab[i] = (int*)malloc(b*sizeof(int));
    for(int i=0;i<a;i++)
        fread(tab[i],sizeof(int),b,plik);
    fclose(plik);
    print(a,b);
    for(int i=0;i<a;i++)
        free(tab[i]);
    free(tab);
}
```

Zadania

Na podstawie materiału omówionego w instrukcji napisać programy wykorzystując strukturę kolekcjonujący dane o trasie podróży (współrzędne x i y, wysokości oraz nazwy punktu). Można wprowadzić dowolną liczbę punktów trasy. Na podstawie zdefiniowanych punktów program oblicza odległości pomiędzy poszczególnymi punktami oraz długość pokonanej trasy od punktu startowego. Program wyświetla tabele z zestawieniem danych.

a) Program zapisuje tabelę z danymi do pliku tekstowego (tą samą co jest wyświetlana na ekranie).

b) Dane poszczególnych punktów zapisuje do pliku binarnego. Program może pobierać dane od użytkownika lub wczytywać dane o punktach z pliku binarnego.