Temat zajęć: Argumenty wywołania programu, operacje na plikach

Autor: mgr inż. Sławomir Samolej

Zagadnienie 1. (Zmienne statyczne)

W języku C można decydować o sposobie przechowywania zmiennych. Decydują o tym tzw. klasy pamięci. Zdefiniowano 4 klasy pamięci:

1. auto (klasa automatyczna)
   Zmienne automatyczne widoczne są tylko wewnątrz bloku, w którym są utworzone (blok to fragment programu zawarty pomiędzy nawiasami: { }). Tworzone są zawsze w momencie wchodzenia do bloku, a niszczone po wyjściu z bloku. Przykładami zamiennych automatycznych są zmienne deklarowane jako zmienne pomocnicze wewnątrz funkcji. Można jawnie deklarować zmienną jako automatyczną:
   { auto int a;
   }
2. static (klasa statyczna)
   Zmienne statyczne widoczne są również tylko wewnątrz bloku, ale tworzone są tylko raz, a ich wartość po wyjściu z bloku jest zachowywana. Jeśli zadeklarujemy zmienną globalną jako statyczną, to ograniczamy jej zasięg tylko do jednego pliku. Zmienną statyczną powołujemy przy pomocy słowa kluczowego static, np.:
   { static int a=5;
   }
3. extern (klasa zewnętrzna)
   Deklarując zmienną jako zewnętrzną informujemy, że będzie ona widoczna w całym programie we wszystkich plikach.
4. register (klasa rejestrowa)
   Zmienna rejestrowa, jeśli jest to możliwe, umieszczana jest w rejestrach procesora w celu przyspieszenia obliczeń. Nie można się do niej odwołć przez adres (wskaźnik). Przykładowa deklaracja zmiennej może mieć postać:
   { register int a=5;
   }

Przykładowy program ilustruje zastosowanie zmiennych statycznych:

<u>Zadanie 1.</u>

Dany jest program:

```
#include <stdio.h>
void licznik_static(void);
void licznik(void);

void main(void)
{
      int k;
      for(k=0;k<5;k++) licznik_static();
      for(k=0;k<5;k++) licznik();
}
```

```
void licznik_static(void)
{
      static int i=0;
      i++;
      printf(”%d\n”,i);
}

void licznik(void)
{
      int i=0;
      i++;
      printf(”%d\n”,i);
}
```

Porównać i wyjaśnić wyniki działania funkcji “licznik” i “licznik_static”. Należy zwrócić uwagę na zastosowanie i interpretowanie przez program zmiennej statycznej.

Zagadnienie 2. (Tablica wskaźników na teksty)

Do przechowywania zestawu informacji tekstowych w programie można posługiwać się albo dwuwymiarowymi tablicami znakowymi:

```
char dane[13][20]=          // można przechować 20 tekstów do 80 znaków każdy
                  {”bledny miesiac”, ”styczeń”, ”luty”, ”marzec”,
                   ”kwiecen”, ”maj”, ”czerwiec”, ”lipiec”, ”sierpień”,
                   ”wrzesien”, ”pazdziernik”, ”listopad”,
                   ”grudzień”
                  };
```

,albo tablicami wskaźników na teksty:

```
char* name[]=
      {”bledny miesiac”, ”styczeń”, ”luty”, ”marzec”,
       ”kwiecen”, ”maj”, ”czerwiec”, ”lipiec”, ”sierpień”,
       ”wrzesien”, ”pazdziernik”, ”listopad”,
       ”grudzień”
      };
```

Stosowanie tablicy wskaźników oszczędza ilość pamięci przeznaczonej na informacje.

<u>Zadanie 2:</u>

Dany jest program:

```
#include <stdio.h>
char* month_name(int n)
{
static char* name[]=
      {”bledny miesiac”, ”styczeń”, ”luty”, ”marzec”,
       ”kwiecen”, ”maj”, ”czerwiec”, ”lipiec”, ”sierpień”,
       ”wrzesien”, ”pazdziernik”, ”listopad”,
       ”grudzień”
      };
return (n<1 || n>12)? name[0]:mname[n]
}

void main(void)
{
      int nr_miesiaca; char *s;
```

```
        printf("podaj numer miesiaca:");
        scanf("%d",&nr_miesiaca);
        s=month_name(nr_miesiąca);
        printf("Miesiac o numerze %d to %s",nr_miesiaca,s);
}
```

- Uruchomić program
- Wyjaśnić wynik działania programu
- Zwrócić uwagę na:
  - sposób deklarowania i odwoływania się do tablicy wskaźników na tekst
  - zastosowanie operatora „ ? : ”.

Należy znać różnicę pomiędzy dwuwymiarowymi tablicami tekstowymi
np.:
```
char tekst_tab1[3][20]={"Tekst1","Tekst2","Tekst3"};
```
a tablicami wskaźników na tekst
np:
```
char* tekst[3]={"Tekst1","Tekst2","Tekst3"};
```

Zagadnienie 3. (Argumenty wywołania programu)

Pełna postać wywołania funkcji main w języku C wygląda następująco:

```
int main(int argc, char* argv[]){}
```

Funkcja zwraca do systemu operacyjnego wartość całkowitą. Zwrócenie 0 oznacza pomyślne zakończenie działania programu. Funkcja może przyjmować 2 argumenty. Umożliwiają one przesłanie do programu i przetwarzanie serii łańcuchów tekstowych (tak zwanych argumentów wywołania programu) bezpośrednio z linii poleceń systemu operacyjnego. Parametr argc zawiera ilość parametrów przesłanych do programu. Parametr argv jest tablicą wskaźników na teksty zawierające poszczególne parametry przesłane do programu. Pierwszy element tablicy wskaźników pokazuje na nazwę uruchomionego programu. Poniższy program wypisuje na konsoli wszystkie argumenty jakie przesłano do niego:

<u>Zadanie 3.</u>

Dany jest program:
```
#include <stdio.h>
#include <conio.h>
int main(int argc, char* argv[])
{
        int i;
        for(i=0;i<argc;i++)
                printf("%s %s",argv[i],(i<argc-1)? " ": "");
        printf("\n");
        getch();
        return 0;
}
```

- Należy przeanalizować i uruchomić program
- Wykorzystując parametry wywołania programu napisać prosty kalkulator dwuargumentowy. Program powinien przyjmować 3 parametry: liczbę, liczbę i operację, która ma między nimi być wykonana, np.:
  `calc 2 4 +`
  gdzie calc jest nazwa programu a 2 4 + są argumentami jego wywołania.
- Zadanie nadobowiązkowe: Napisać prosty kalkulator pracujący w odwrotnej notacji polskiej. Odwrotna notacja polska polega na tym, że najpierw podawane są dwie liczby a potem operacja na nich np:
  `calc_pl 2 3 4 + -`
  jest jednoznaczne z wykonaniem operacji 2-(3+4).

Zagadnienie 4. (Operacje na plikach)

Przetwarzanie plików w języku C odbywa się zawsze w trzech etapach:
- otwarcie pliku (funkcja fopen())
- zapis lub odczyt do/z pliku(funkcje fgetc(), fputc(), fscanf(), fprintf())
- zamknięcie pliku (funkcja fclose()).

Komunikacja z plikiem odbywa się przez wskaźnik do "uchwytu do pliku":

```
FILE* plik;
```

Pełna postać deklaracji funkcji fopen ma postać:

```
FILE* fopen(char* name, char *mode);
```

Parametr name powinien zawierać ścieżkę dostępu do pliku, który chcemy otworzyć. Parametr mode powinien zawierać tryb, w jakim plik ma być otwarty. Wybrane tryby otwarcia pliku mają postać:
"r" - plik będzie otwarty do czytania;
"w" - plik będzie otwarty do zapisu, jeśli plik o podanej nazwie istnieje, to zostanie usunięty z dysku, a następnie ponownie utworzony;
"a" - plik jest otwarty do dopisywania na końcu.
Funkcja fopen w przypadku pomyślnego nawiązania komunikacji z plikiem zwraca wskaźnik na obiekt typu FILE. W przypadku braku pliku funkcja zwraca NULL (0).
Funkcje umożliwiające zapis lub odczyt zawartości pliku korzystają z uzyskanego wskaźnika na obiekt typu FILE.
Przykładowo odczyt pojedynczego znaku z pliku może się odbywać następująco:

```
FILE *plik;
plik=fopen("\\autoexec.bat","r");
char c;
c=fgetc(plik);
```

Zapis pojedynczego znaku do pliku może odbywać się następująco:

```
FILE *plik;
plik=fopen("\\tekst1.txt","w");
fputc('a',plik);
```

Możliwy jest również zapis i odczyt danych tekstowych w sposób sformatowany:

```
// odczyt z pliku:
FILE *plik;
plik=fopen("\\dane.txt","r");
float dana;
fscanf(plik,"%f",&dana);
```

```
//zapis do pliku:
FILE *plik;
plik=fopen("\\dane.txt","w");
float dana=3.4;
fprintf(plik,"%f",dana);
```

Przed zakończeniem działania programu, w którym odbywała się komunikacja z plikami dyskowymi należy zamknąć otwarte uprzednio pliki. Służy do tego funkcja:
`fclose(plik);.`

Zadanie 4:

Należy utworzyć plik tekstowy o nazwie "tekst.txt"

Dany jest program wypisujący zawartość zadanego pliku tekstowego:

```
#include <stdio.h>
int main(void)
{
	FILE* plik;
	char c;
	if((plik=fopen("tekst.txt","r"))==NULL)
	{
		printf("Nie moge otworzyc pliku:  tekst.txt");
		return(1);
	}
	while((c=fgetc(plik))!=EOF)
		putchar(c);
	fclose(plik)
	return 0;
}
```

- Sprawdzić działanie programu
- Napisać program, który wykorzystując parametry wywołania programu będzie w stanie wypisywać na ekranie zawartość dowolnie wskazanego pliku tekstowego, np.: **wypisz.exe wypisz.c** (należy zastosować argumenty wywołania programu)
- Napisać program, który oblicza średnia arytmetyczną z danych zawartych w pliku tekstowym postaci:
  3.5
  2.0
  4.5
  5.0
  4.5
  3.0
  4.0
  4.0
  (kolejne oceny studenta zapisane są w pliku tekstowym w postaci pojedynczej kolumny liczb).
- Napisać program, który prosi kolejno użytkownika o podanie danych: imienia, nazwiska i nr telefonu. Podawane dane zapisuje do ustalonego pliku tekstowego w postaci:
  Jan

Kowalski
2345672

Waldemar
Nowak
3345223

- Napisać program, który zapisane w poprzednim programie dane odczytuje i wypisuje w postaci tabeli.