МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ТА ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДЛЯ САМОСТІЙНОЇ РОБОТИ СТУДЕНТІВ ЗАОЧНОЇ ФОРМИ НАВЧАННЯ З ДИСЦИПЛІНИ

“ОСНОВИ ПРОГРАМУВАННЯ”

ЧАСТИНА 2

Житомир
2019

УДК 004.424

Марчук Г.В., Левківський В.Л., Чижмотря О.В. Методичні рекомендації та індивідуальні завдання для самостійної роботи студентів заочної форми навчання з дисципліни "Основи програмування". Ч.2 / Г. В. Марчук, В. Л. Левківський, О. В. Чижмотря. – Житомир: ЖДТУ, 2019. – 42 с.

Методичні рекомендації містять теоретичний матеріал, приклади та вказівки для виконання завдань самостійної роботи, пов’язаних з вивченням основ програмування мовою С. Методичні рекомендації призначені для студентів, які навчаються за заочною формою за спеціальностями 121 "Інженерія програмного забезпечення" та 125 "Кібербезпека".

Рецензенти:

Сугоняк І.І. – к.т.н., завідувач кафедри комп’ютерних наук Житомирського державного технологічного університету

Єфремов Ю.М. – к.т.н., доцент кафедри інженерії програмного забезпечення Житомирського державного технологічного університету

Затверджено на засіданні кафедри інженерії програмного забезпечення.

Протокол № 4 від 25 лютого 2019 р.

УДК 004.424

Зміст

Вступ

Методичні рекомендації призначені для виконання завдань самостійної роботи студентів заочної форми навчання з навчальної дисципліни "Основи програмування".

Методичні рекомендації містять опис робіт за 10 темами, кожна з яких складається з 15 варіантів завдань. Перед виконанням кожної роботи необхідно вивчити відповідний теоретичний матеріал. Приклади програм, що наведені у методичних рекомендаціях, слід розглядати лише як один із можливих варіантів розв’язання поставленої задачі.

Під час виконання робіт студент повинен продемонструвати:

- творчий підхід до розробки програмного засобу;
- навики програмування на мові високого рівня С.

Студент повинен вміти провести аналіз поставленої задачі, написати код програми та провести аналіз отриманого результату.

Варіант завдання самостійної роботи обирається відповідно до номера студента за списком групи.

Типовий порядок виконання робіт й методичні рекомендації до її виконання:

1) ознайомитися з методичними рекомендаціями до роботи;
2) провести аналіз поставленої задачі;
3) побудувати блок-схему
4) написати код програми (програма має починатися з коментарів, що пояснює її призначення та авторські права);
5) провести налагодження написаної програми;
6) відповісти на контрольні питання;
7) оформити звіт і здати викладачу.

Звіт з будь-якої роботи повинен містити:

1. Титульний лист відповідно зразка.
2. Опис виконаних завдань:
   - Умова завдання (завдань).
   - Графічне представлення алгоритму.
   - Лістинг програми.
   - План тестування та результати роботи програми на тестових даних.
3. Висновки по роботі з урахуванням усіх виконаних завдань:
   - аналіз отриманих результатів за кожним пунктом завдання;
   - аналіз результатів тестування програм.

Вимоги до оформлення звіту

Поля сторінки: ліве – 2,5 см, праве, верхнє й нижнє –1,5 см; шрифт Times New Roman (14 пт), міжрядковий інтервал – множник 1,5.

Номери сторінок мають знаходитися у правому нижньому куті. Титульний лист не нумерується.

Тема № 16.
Функції.

Мета: отримати практичні навички написання процедур і функцій за допомогою конструкцій мови, а також вибору правильного способу передачі параметрів.

Література:

С/С++: Теорія та практика. Навчально-методичний посібник. / В. В. Войтенко, А. В. Морозов. – Житомир: ЖДТУ, 2004. – 324 с.

Зміст роботи

Завдання:

1	Написати функцію обчислення площі прямокутника	$S = a \cdot b$
2	Написати функцію обчислення периметра прямокутника	$P = 2(a + b)$
3	Написати функцію обчислення площі паралелограма	$S = a \cdot b \cdot \sin \alpha$
4	Написати функцію обчислення периметра трапеції	$P = a + b + c + d$
5	Написати функцію обчислення площі еліпсу	$S = \pi a b$
6	Написати функцію обчислення периметра еліпсу	$P=4((\pi ab + (a - b))/(a+b))$
7	Написати функцію обчислення периметра трикутника	$P = a + b + c$
8	Написати функцію обчислення площі трикутника	$s = \sqrt{p(p-a)(p-b)(h-c)}$
9	Написати функцію обчислення площі трапеції	$s = \frac{a+b}{\lvert a-b \rvert}\sqrt{(p-a)(p-b)(p-a-c)(p-a-d)}$
10	Написати функцію обчислення площі трапеції	$S=1/2(a+b)h$
11	Написати функцію обчислення площі трикутника	$S=1/2ab\sin\gamma$
12	Написати функцію обчислення евклідової відстані між двома точками на площині	$AB=\sqrt{((x_b-x_a)^2+(y_b-y_a)^2)}$
13	Написати функцію обчислення площі чотирикутника	$S = a^2$
14	Написати функцію обчислення довжини кола	$C = 2 \pi r$
15	Написати функцію обчислення площі круга	$S = \pi r^2$

Методичні рекомендації

Функція – іменована послідовність описів і операторів, яка виконує деяку закінчену послідовність дій. Будь-яка функція складається із заголовка (оголошення функції) і тіла (визначення функції).

Оголошення функції описує її *прототип* (іноді кажуть "сигнатура").

Прототип функції оголошується наступним чином:

ТипПовернення Ім'яФункції (СписокОголошенихПараметрів);

ТипПовернення – тип даних, що повертається функцією.

СписокОголошенихПараметрів задає тип і ім'я кожного з параметрів функції, розділених комами. Список параметрів функції може бути порожнім. Приклади прототипів функцій:

double func (double parl, double par2);

int func (int, int) ;

void func() ;

Визначення функції складається з її *заголовка* і власне *тіла*, вкладеного у фігурні дужки і такого, що має смислове навантаження. Якщо функція повертає будь-яке значення, в тілі функції обов'язково повинен бути присутнім оператор повернення з параметром того ж типу.

ТипПоверн Ім'яФункції (СписокОголошенихПараметрів)

{ // тіло функції }

Виклик функції – вказівка ідентифікатора функції (її імені), за яким в круглих дужках слідує список *аргументів*, розділених комами.

double maxValue = func (2.5, 1235.2);

int j = func (10, 2);

func();

Функції, що не повертають значення це функції типу *void* - можуть розглядатися, як деякий різновид команд, реалізований особливими програмними операторами. Оператор *func();* виконує функцію *void func(),* тобто передає керування функції, доки не виконаються усі її оператори. Функція поверне керування в основну програму, коли завершить свою роботу, програма продовжить своє виконання з того місця, де розташовується наступний оператор за оператором *func().*

Якщо в якості параметра функції використовується позначення масиву, то насправді в функцію передається адреса першого елемента масиву.

Приклади:

Визначення максимального елементу масиву:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int maximum(int, int);

int main(){
  srand((unsigned)time(NULL));
  int a[6];
  for(int i=0;i<6;i++){
     a[i]=rand()%10;
     printf("%5i",a[i]);
  }
     int c=a[0];
     for(int i=0;i<6;i++)
           c=maximum(a[i], c);
     printf("\nmax=%i",c);
return 0;
}

int maximum(inta, intb){
     if (a>=b) returna;
     else       returnb;
}
```

Обчислення суми елементів масиву:

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

int sum (intn, inta[] ){
     int i, s=0;
     for( i=0; i<n; i++ )
          s+=a[i];
return s;
}


void main(){
int a[]={ 3, 5, 7, 9, 11, 13, 15 };
int s = sum( 7, a );
     printf("sum=%i\n",s);
}
```

Контрольні питання:

1. Що називають функцією? Що таке опис функції?
2. Що таке прототип функції? Де його розміщують у програмі?
3. Які символи оточують тіло функції? Які символи оточують аргументи функції?
4. Чи обов’язково в прототипах функцій вказувати ідентифікатори змінних?
5. Як відбувається звернення до функції?
6. Яке ключове слово використовується в заголовку функції, щоб показати, що функція не повертає нічого і немає жодного параметра?
7. Чи кожна функція повинна мати оператор повернення?

Тема № 17.
Функції.

Мета:отримати практичні навички написання процедур і функцій за допомогою конструкцій мови, а також вибору правильного способу передачі параметрів.

Література:

С/С++: Теорія та практика. Навчально-методичний посібник. / В. В. Войтенко, А. В. Морозов. – Житомир: ЖДТУ, 2004. – 324 с.

Зміст роботи

Завдання:

1	Написати функцію, яка порівнює два цілих числа і повертає результат порівняння у вигляді одного із знаків: >, < або =
2	Написати функцію Procent, яка повертає відсоток від отриманого в якості аргументу числа.
3	Написати функцію, яка обчислює дохід за вкладом. Вхідними даними для функції є: величина вкладу, процентна ставка (річних) і термін вкладу (кількість днів).
4	Написати функцію, яка повертає 1, якщо символ, отриманий функцією як аргумент, є голосною буквою російського алфавіту, і нуль - в іншому випадку.
5	Написати функцію, яка визначає знак числа і повертає результат порівняння у вигляді: + або -
6	Написати функцію, яка виводить на екран рядок, що складається з зірочок. Довжина рядка (кількість зірочок) є параметром функції.
7	Написати функцію (процедуру), що виводить на екран середнє значення послідовності дійсних чисел довільної довжини. Перший параметр процедури задає кількість елементів послідовності. Підпрограму оформити з використанням прототипу.
8	Написати функцію обчислення факторіала числа x. Підпрограму оформити з використанням прототипу

Методичні рекомендації

Існує три способи передачі параметрів у функцію:

1) за значенням;
2) по посиланню;
3) за вказівником.

Передача параметрів за значенням - найпоширеніший і надійний спосіб передачі, при якому передбачається, що кожен оголошений в заголовку параметр є локальною змінною в тілі функції. В результаті передача за

значенням еквівалентна операції присвоювання. Значення фактичного параметра (в якості якого можуть виступати вираз, змінна, константа) копіюється в змінну-параметр. Якщо фактичний параметр є змінною, то будь-які зміни параметра всередині функції не впливають на значення змінної поза функцією. У всі стандартні математичні функції: sin, cos, exp і ін., - параметри передаються за значенням. У функції приклад 1 параметр передається за значенням.

Приклад 1

```
//Процедура виводу на екран n символів.
void PrintLine(int n)
{
      for(int i = 0; i < n; i++) printf("-");
      printf("\n");
}
// Функція підрахунку залишку від ділення двох чисел.
float GetRest(long double x1, float x2)
{
      float flRest = (float) x1 / x2;
      return (flRest - (int)flRest);
}
```

Спосіб передачі параметрів по посиланню (приклад 2) застосовується коли необхідно змінити значення однієї або декількох змінних, формальних параметрів, в тілі підпрограми. Визначення функції, що виконує обмін значень двох змінних, з передачею параметрів по посиланню.

Приклад 2

```
bool Swap (int &a, int &b)          // параметри по посиланню
{
      if(a == b) return false;            // обмін не виконується
      int t = a;                    // тимчасова змінна
      a = b;
      b = t;
      return true;                  // функція виконала обмін
}
```

Передача параметрів за вказівником використовується у випадках, коли необхідно отримати доступ всередині підпрограми до змінних, розташованим за її межами. Вибір одного із способів передачі параметрів залежить від семантики створюваної функції.

Контрольні питання:

1. Дайте визначення поняттю підпрограма.
2. Що таке функція?
3. Які є відмінності між поняттями процедура і функція?
4. Опишіть синтаксис оголошення параметрів підпрограми.
5. Що таке прототип підпрограми?
6. Скільки відомо способів передачі параметрів в підпрограму? Який з них застосовується найбільш часто?
7. Що слід брати до уваги при виборі способу передачі параметрів в підпрограму?

Тема № 18.
Рекурсія.

Мета: Формування навиків роботи із функціями. Вивчення методів використання алгоритмів і програм з рекурсією в мові Сі

Література:

С/С++: Теорія та практика. Навчально-методичний посібник. / В. В. Войтенко, А. В. Морозов. – Житомир: ЖДТУ, 2004. – 324 с.

Зміст роботи

Завдання 1.

1	За допомогою рекурсивної функції здійснити виведення на екран елементів одновимірного масиву
2	За допомогою рекурсивної функції здійснити пошук максимального елемента одновимірного масиву
3	За допомогою рекурсивної функції здійснити пошук мінімального елемента одновимірного масиву
4	За допомогою рекурсивної функції обчислити суму елементів одновимірного масиву
5	За допомогою рекурсивної функції обчислити середнє арифметичне елементів одновимірного масиву

Завдання 2.

Вирішити задачу двома способами - із застосуванням рекурсії і без неї.

1	Скласти програму для перекладу даного натурального числа в р-ічную систему числення (2 < p < 9).
2	Підрахувати значення поліному степені n за формулою $$P_n = \sum^{n} a_i x^i = a_0 + x\left(a_1 + ...x\left(a_{n-1} + xa_n\right)...\right)$$
3	Знайти максимальний елемент масиву a_1, ..., a_n, використовуя метод ділення навпіл max (a_1, ..., a_n) = max (max (a_1, ..., $a_{n/2}$), max ($a_{n/2+1}$, ..., a_n)).
4	Підрахувати $y(n) = \sqrt{1+\sqrt{2+...+\sqrt{n}}}$
5	Підрахувати значення суми S = 1/1! + 1/2! + ... + 1/k!

Методичні рекомендації

Рекурсія – це спосіб організації обчислювального процесу, при якому функція в ході виконання операторів звертається сама до себе.

Функція називається рекурсивною, якщо під час її роботи можливий повторний її виклик безпосередньо (*прямий виклик*) або шляхом виклику іншої функції, в якій міститься звернення до неї (*непрямий виклик*).

Прямою (безпосередньою) рекурсією називається рекурсія, при якій всередині тіла деякої функції міститься виклик тієї ж функції.

void fn(int i) { ... fn(i); ... }

Непрямою рекурсією називається рекурсія, що здійснює рекурсивний виклик функції шляхом ланцюга викликів інших функцій. При цьому всі функції ланцюга, що здійснюють рекурсію, вважаються рекурсивними.

Для визначення поняття рекурсії розглянемо функцію факторіал, що визначається рівнянням

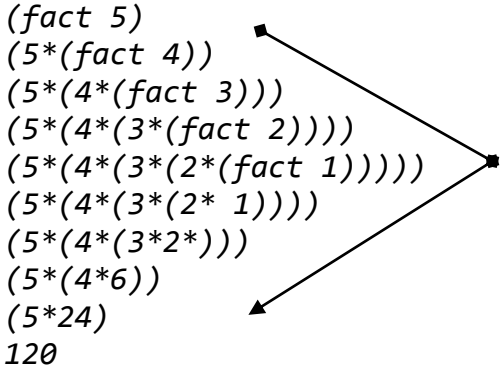
$$n! = n \cdot (n-1) \cdot (n-2) \cdot \cdot \cdot 3 \cdot 2 \cdot 1.$$

Існує безліч способів обчислювати факторіали. Один з них полягає у такому спостереженні – n! для будь - якого позитивного цілого числа n дорівнює n, помноженому на (n −1)!:

$$n! = n \cdot [(n-1) \cdot (n-2) \cdot \cdot \cdot 3 \cdot 2 \cdot 1] = n \cdot (n-1)!.$$

Отже, ми можемо знайти n!, обчисливши спочатку (n -1)!, а потім помноживши його на n. Після того, як ми додамо початкову умову – 1! дорівнює 1, отримане спостереження можна безпосередньо перевести у процедуру , представлену на малюнку:

при n=5;



Отже, такий метод визначення факторіалу має назву рекурсивного тому, що аргументом функції є ця ж сама функція.

```
int fact( int n)
{
return ( (n==1) ? 1 : n*fact(n-1));
}
```

```
int fact(intn)
{
  if(n==0) return 1;
  returnn*fact(n-1);
}
```

Контрольні питання:

1. Дайте поняття рекурсії.
2. В яких завданнях доцільно використовувати рекурсивні функції?

Тема № 19.
Покажчики.

Мета: набути навичок роботи з покажчиками в одновимірному масиві.

Література:

С/С++: Теорія та практика. Навчально-методичний посібник. / В. В. Войтенко, А. В. Морозов. – Житомир: ЖДТУ, 2004. – 324 с.

Зміст роботи

Завдання 1. Дано масив. Скласти програму де необхідно:

1) Визначити розмір масиву в байтах.
2) Визначити кількість елементів масиву.
3) Вивести на екран адреси першого і останнього елементів масиву.
4) Здійснити переписування масиву у зворотному порядку.

Завдання 2. *Індивідуальні завдання.*

Сформувати одновимірний масив цілих чисел, використовуючи датчик випадкових чисел.

1	1. Видалити елемент з номером К. 2. Додати після кожного парного елемента масиву елемент із значенням 0.
2	1. Видалити перший елемент що дорівнює 0. 2. Додати після кожного парного елемента масиву елемент із значенням mas[i-1]+2.
3	1. Видалити всі елементи що дорівнюють 0. 2. Додати після першого парного елемента масиву елемент із значенням mas[i-1]+2.
4	1. Видалити елементи, індекси яких кратні 3. 2. Додати після кожного негативного елемента масиву елемент із значенням mas[i-1]+1
5	1. Видалити із масиву всі елементи які кратні 7. 2. Додати після кожного непарного елемента масиву елемент із значенням 0
6	1. Видалити елемент із заданим номером. 2. Додати після першого парного елемента масиву елемент із значенням mas[i-1]+2.
7	1. Видалити останній елемент що дорівнює 0. 2. Додати після елемента масиву із заданим індексом елемент із значенням 100.

8	1. Видалити всі елементи із заданим значенням. 2. Додати перед кожним парним елементом масиву елемент із значенням 0.
9	1. Видалити перший елемент із заданим значенням. 2. Зсунути масив циклічно на К елементів вправо.
10	1. Видалити 5 перших елементи масиву. 2. Додати в кінець масиву 3 нові елементи.
11	1. Видалити 5 останніх елементів масиву. 2. Додати на початок масиву 3 елементи із значенням mas[i+1]+2.
12	1. Поміняти місцями мінімальний і максимальний елементи масиву. 2. Видалити з масиву всі елементи, що перевищують середнє значення більш, ніж на 10%.
13	1. Видалити з масиву всі елементи, які співпадають із його мінімальним значенням. 2. Додати на початок масиву три елементи із значенням середнього арифметичного масиву.
14	1. Перевернути масив і, якщо число елементів масиву непарне, видалити його середній елемент. 2. Додати на початок масиву 3 елементи із значенням mas[i+10]-2.
15	1. Видалити елемент з номером К. 2. Додати після кожного парного елемента масиву елемент із значенням 0.

Методичні рекомендації

Покажчик – це адреса деякого об’єкту, через яку можна звертатися до цього об’єкту. Операція &дає адресу змінної і використовувати її можна тільки до змінних і елементів масиву.

Приклад:

```
int x,*y; y=&x; //присвоєння адреси змінної x змінній-покажчику y
```

Операція * сприймає свій операнд як адресу деякого об’єкту і використовує цю адресу для вибірки вмісту.

Приклад:

```
int x, *y, z; y=&x; z=*y; //отримання значення за адресою покажчика
```

Покажчики можна використовувати як операнди у арифметичних операціях. Арифметичні операції застосовуються тільки до покажчиків одного типу.

- Інкремент збільшує значення покажчика на величину sizeof(тип).

```
char* pc;
int* pi;
```

```
double* pd;
...
pc++; //значення збільшується на 1
pi++; // значення збільшується на 4
pd++; // значення збільшується на 8
```

- Декремент зменшує значення покажчика на величину sizeof(тип).
- Різниця двох вказівників - це різниця їх значень, поділена на розмір типу в байтах. Підсумовування двох покажчиків не допускається.
- Можна підсумувати покажчик і константу

Конструкція y+n (y – покажчик, n – ціле число) задає адресу n-го об'єкту, на який вказує y. Це справедливо для будь-яких об'єктів (int, char, float і т.п.).

Будь-який покажчик можна перевірити на рівність (==) або нерівність (!=) зі спеціальним значенням NULL.

```
int *y; y=NULL; int x=5; if(y==NULL) y=*x;
```

Після створення покажчика в ньому знаходиться довільне значення, тобто він посилається на будь-який фрагмент пам'яті. Якщо по тексту програми покажчик перевіряється на NULL, то після створення покажчика його необхідно приводити до значення NULL.

Зв'язок масивів і покажчиків

Якщо задати: *int mas[100], *p, a;* то:

1) для масиву відводиться пам'ять в адресному просторі під 100 елементів типу int;

2) пам'ять відводиться під покажчик-константу з ім'ям mas, значенням покажчика є адреса масиву;

3) пам'ять відводиться під покажчик-змінну з ім'ям p.

Операція ініціалізації покажчика може здійснюватися тільки операцією присвоєння адреси деякої змінної:

p=&a;

p=&mas[0]; або p=mas; або присвоєнням *p=NULL*.

Допустимо p=0, але не рекомендується.

Помилкою є:

```
a=10;
p=a; // де p – покажчик. Присвоєння неможливе, так як типи int* і int.
p=10; // присвоєння неможливе, так як тип int* и const int.
```

Покажчику неможна присвоювати цілі значення, але можна додавати і віднімати покажчик і цілі числа.

Наприклад:

- *p+=10; збільшує на 10 вміст комірки пам'яті, на яку посилається p, еквівалентно mas[0]=mas[0]+10;
- якщо p=mas; то p+=10; еквівалентно p=p+10 і еквівалентно присвоєнню p=&mas[10];

Якщо 2 покажчика посилаються на елементи одного і того ж масиву, то допускаються операції відношення над ними: ==; !=; <, >, і т. п.

Покажчик можна присвоїти іншому покажчику:

```
int x;
int *p1, *p2;
p1 = &x;
p2 = p1;
printf(" %p", p2); /* Виводить адресу змінної x, не її значення */
```

У прикладі на змінну x посилаються обидва покажчика p1 і p2.

Динамічне виділення пам'яті під масиви

Змінні у програмах повинні розміщатися в одному з трьох місць: в області даних програми, в області стеку, в області вільної пам'яті (купи).

Кожній змінній у програмі може відводитися пам'ять або статично (в момент завантаження), або динамічно (у процесі виконання програми).

До цих пір всі масиви оголошувались статично, а отже, зберігали значення своїх елементів в області даних. Якщо кількість елементів невелика, таке розміщення виправдано. Але досить часто виникають випадки, коли необхідно мати великі масиви даних або розмір масиву заздалегідь не може бути визначений. Тут на допомогу приходить можливість використання динамічної пам'яті.

Для того, щоб у пам'яті можна було розмістити будь-який динамічний об'єкт, для нього необхідно попередньо виділити відповідне місце. По завершенні роботи з об'єктом виділену пам'ять необхідно звільнити.

Виділення пам'яті можна здійснювати за допомогою функцій ***malloc(), calloc()***, ***free()***. Ці функції описані у заголовочному файлі <*stdlib.h*> або <*malloc.h*>

```
void* malloc(size_t size);
void* calloc(size_t n, size_t size);
void* realloc(void* ptr, size_t size);
```

Функція *malloc(size)* виділяє *size* байтів з купи. У випадку успішного виділення пам'яті покажчик встановлюється на виділений блок пам'яті. При невдалому виділенні пам'яті функція повертає *NULL*.

Функція *calloc(num, size)*, окрім виділення області пам'яті під масив об'єктів, ще здійснює ініціалізацію елементів масиву нульовими значеннями. *num* вказує, скільки елементів буде зберігатися у масиві, а *size* – розмір кожного елемента у байтах.

Динамічне виділення пам'яті для одновимірного масиву цілих чисел розміром *SIZE* і заповнення елементів масиву:

```
int* p1 = NULL;
int* p2 = NULL;
p1= (int*) malloc(sizeof(int) * SIZE);
//p1= (int*) calloc(SIZE, sizeof(int));
p2 = p1;
```

```
printf("\n---------------p1------------------\n");
for (int i = 0; i < SIZE; i++)
{
      p1[i] = i;
      printf("%d ", p1[i]);
}
printf("\n---------------p2------------------\n");
int  SIZE_N=SIZE +5;
p2 = (int*) realloc(p1, SIZE_N*sizeof(int));
for (int i = SIZE; i < SIZE_N; i++)
      p2[i] = i;
```

Функція *free*(*p*) звільняє пам’ять, на яку вказує покажчик.

Контрольні питання:

1. В чому різниця між статичними та динамічними масивами?
2. Коли і звідки виділяється пам’ять для статичних і динамічних масивів?
3. Як оголосити одновимірний динамічний масив?
4. Як визначити розмір одновимірного масиву?
5. Наведіть фрагмент коду для заповнення масиву цілими (дійсними) випадковими числами.

Тема № 20.
Покажчики.

Мета: набути навичок роботи з покажчиками.

Література:

С/С++: Теорія та практика. Навчально-методичний посібник. / В. В. Войтенко, А. В. Морозов. – Житомир: ЖДТУ, 2004. – 324 с.

Зміст роботи

Завдання. Написати програму, в якій потрібно створити двовимірний динамічний масив і виконати обробку даних.

1	Видалити із масиву рядок із заданим номером і стовпець, де заходиться максимальний елемент.
2	Видалити із масиву стовпець із заданим номером і рядок, де заходиться мінімальний елемент.
3	Додати до масиву рядок із заданим номером і видалити стовпець, де заходиться мінімальний елемент.
4	Додати до масиву стовпець із заданим номером і видалити перший рядок, де заходиться 0 елемент.
5	Видалити із масиву рядок із заданим номером і рядок, де заходиться максимальний елемент, якщо рядки співпадають, видати повідомлення
6	Видалити із масиву рядок і стовпець із заданим номером
7	Додати до масиву рядок і стовпець із заданим номером
8	Видалити із масиву всі рядки, в яких зустрічається задане число.
9	Видалити із масиву всі стовпці, в яких зустрічається задане число
10	Видалити із масиву рядок і стовпець, на перетині яких знаходиться мінімальний елемент.
11	Видалити із масиву рядок який знаходиться до рядка з максимальним елементом і додати стовпець після стовпця з максимальним елементом
12	Видалити із масиву рядок до і рядок після рядка з мінімальним елементом
13	Видалити із масиву стовпчик до і стовпчик після стовпчика з мінімальним елементом
14	Видалити із масиву всі рядки, в яких зустрічається максимальне і мінімальне число
15	Видалити із масиву рядок і стовпець, на перетині яких знаходиться максимальний елемент

Методичні рекомендації

Багатомірний масив у мові Сі розглядається як сукупність одновимірних масивів (його рядків). Для звертання до елементів багатомірного масиву можна використовувати нуль і більш індексів (індексних виразів):

ім'я-масиву [вираз1][вираз2] ...

Приклад оголошення двовимірного масиву значень типу *int*: *int a[n][m];*

Цей масив складається з *n* одновимірних масивів (рядків), у кожному з яких утримується *m* елементів (стовпців). При роботі з цим двовимірним масивом можна використовувати одно або 2 індексних вирази. Наприклад:

a[i][j]- містить 2 індекси; використовується для звертання до елемента *i*-рядка, *j*-стовпця масиву; обчислюються індексні вирази, визначається адреса елемента масиву і вилучається його значення;

a[i] - містить 1 індекс; визначає адресу одновимірного масиву: адреса початку і-рядка масиву;

a - не містить індексу і визначає адресу масиву, його нульового елемента.

Таким чином, звертання до двовимірних масивів за допомогою імені і тільки одного індексу визначає покажчик на початок відповідного рядка масиву (адреса його нульового елемента). Наприклад:

*a[0] = &a[0][0] = a+0*n*sizeof(int);*

*a[1] = &a[1][0] = a+1*n*sizeof(int);*

*a[i] = &a[i][0] = a+i*n*sizeof(int);*

В ОП елементи масиву a розташовуються таким чином, що швидше всіх змінюється самий правий індекс, тобто в послідовності: a[0][0] a[0][1] a[0][2]... a[2][2] a[2][3].

При цьому для звертання до масиву a можна використовувати імена:

*&a = a = &a[0][0] = *a* - адреса a[0][0] - елемента 0 рядка 0 стовпця масиву a;

***a = *(&a[0][0]) = a[0][0]* - значення елемента 0 рядка 0 стовпця масиву a;

*a[i] = (a + i) = *(a + i) = &a[i][0]* - адреса елемента *i*-рядка 0-стовпця;

**a[i] = **(a + i) = *(&a[i]) = a[i][0]* - значення 0 елемента *i*-рядка;

*a[i][j] = *(*(a + i) + j) = *(a[i] + j) = a[i][j]* - значення елемента *i*-рядка *j*-стовпця масиву *a*;

де: *(a + i) = *(a + i) = a[i]* - адреса 0 елемента *i*-рядка = &a[i][0];

((a + i) + j)*- адреса *j*-елемента *i*-рядка = &a[i][j];

**(*(a + i) + j)*- значення *j*-елемента і-рядка = a[i][j].

Значення адреси початку *i*-рядка (адреси 0-елемента *i*-рядка) на машинному рівні формується у виді:

*a[i] = a + i == (a+i*n*sizeof(int))*, де *n* - кількість значень в одному рядку.

Таким чином, адреса (i+1)-рядка знаходиться від *i*-рядка на (n*sizeof(int)) байтів, тобто на відстань одного рядка масиву.

Вираз a[i][j] компілятор Сі переводить в еквівалентний вираз: **(*a + i) + j)*. Запис a[i][j] більш традиційний у математиці і більш наочний.

До елементів двовимірного масиву можна звернутися і за допомогою скалярного покажчика на масив. Наприклад, після оголошення:

*int a[n][m], *p = a;*

**(p+i*m+j)* - значення *j* - елемента i-рядка ;

де: m - кількість елементів у рядку;

i*m + j - a[i][j]- елемент відносно початку масиву *a*.

У наведеній нижче програмі використовуються звертання до pi,j - елементів масиву у вигляді: *(p+i*m+j) - адреса pi,j-елемента масиву.

```
#include ...
void main()
{
    srand(time(NULL));
    int *p, n=4, m=5;
            p = (int*) malloc (n*m * sizeof (int));
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j <m; j++ )
        {
             *(p+i*m+j) = rand()%10;
            printf("%3d",  *(p+i*m+j));
        }
        printf ("\n" ) ;
    }
        free( p );
        getch();
}
```

Динамічне виділення пам'яті під масиви

Для створення двовимірного масиву спочатку потрібно розподілити пам'ять для масиву покажчиків на одномірні масиви, а потім розподіляти пам'ять для одновимірних масивів. Наприклад, потрібно створити масив arr[n][m]:

```
#include ...
void main()
{
    srand(time(NULL));
    int **arr=NULL;
    int n,m;
    printf("n= ");scanf("%d",&n);
    printf("m= ");scanf("%d",&m);
    /*arr=(int **)calloc(n,sizeof(int *));      // calloc
    for (int i=0; i<=n; i++)
        arr[i]=(int *)calloc(m,sizeof(int));*/

     arr = (int**) malloc(n * sizeof(int*));    // malloc
    for (int i = 0; i < n; i++)
       arr[i] = (int*) malloc(m * sizeof(int));
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j <m; j++ )
        {
             arr[i][j]= rand()%10;
            printf("%3d", arr[i][j]);
        }
        printf ("\n" ) ;
    }
    for (int i = 0; i < n; i++)
        free(arr[i]);
    free(arr);
    getch();
}
```

Контрольні питання:

1. Що таке покажчик?
2. Як можна використовувати покажчики у багатовимірних масивах?
3. Що таке динамічна пам'ять?
4. Як оголосити одновимірний динамічний масив?
5. Як оголосити динамічний двовимірний масив?
6. В чому різниця між статичними та динамічними масивами?
7. Коли і звідки видялється пам’ять для статичних і динамічних масивів?
8. Наведіть фрагмент коду для заповнення динамічного масиву цілими (дійсними) випадковими числами.
9. Наведіть фрагмент коду для обчислення суми (добутку) елементів одновимірного динамічного масиву?
10. Наведіть фрагмент коду для підрахунку суми (добутку) елементів певного рядка (стовпця) двовимірного масиву?

Тема № 21.
Робота з рядковими та символьними змінними.

Мета: Формування навиків роботи із рядковими змінними та розробки алгоритмів їх обробки. Отримання практичних навиків при передачі рядків у функцію

Література:

С/С++: Теорія та практика. Навчально-методичний посібник. / В. В. Войтенко, А. В. Морозов. – Житомир: ЖДТУ, 2004. – 324 с.

Зміст роботи

Завдання 1. *Написати програму та протестувати. В програмі використовується масив символів.*

1	Дано рядок. Підрахувати кількість цифр в рядку.
2	Дано рядок. Підрахувати в рядку кількість прописних (малих) латинських букв.
3	Дано рядок. Підрахувати загальну кількість великих латинських літер.
4	Дано рядок. Перетворити в ньому всі малі літери у прописні, а прописні - у рядкові.
5	Дано рядок. Підрахувати в ньому кількість «(» та «)» дужок.
6	Дано рядок. Підрахувати в ньому кількість букв та цифр.
7	Дано рядок. Підрахувати в ньому кількість голосних літер.
8	Дано рядок. Вивести рядок, який містить ті ж символи, але у зворотному порядку.
9	Дано рядок, що зображує ціле додатне число. Підрахувати суму цифр цього числа.
10	Дано рядок. Підрахувати в ньому кількість приголосних літер.
11	Дано рядок. Підрахувати в ньому кількість пробілів.
12	Дано рядок, що зображує ціле додатне число. Визначити розрядність цього числа.
13	Дано рядок. Підрахувати в ньому кількість знаків пунктуації.
14	Дано рядок, що складається із цифр, арифметичних операцій та деякої кількості відкриваючих та закриваючих дужок. Перевірити кількість відкриваючих та закриваючих дужок.
15	Дано рядок, що зображує десятковий запис цілого додатного числа. Вивести рядок, що зображує двійковий запис цього ж числа.

Завдання 2.

З клавіатури вводиться текстовий рядок. Розробити програму, що використовує покажчики:

1	а) підраховує кількість слів, які мають непарну довжину; б) виводить на екран частоту входження кожної літери; в) видаляє текст, що розміщено в круглих дужках.
2	а) перевіряє, чи співпадає кількість відкритих і закритих дужок у введеному рядку (перевірити для круглих та квадратних дужок); б) виводить на екран найдовше слово; в) видаляє всі слова, що складаються тільки з латинських літер.
3	а) підраховує кількість різних слів, що входять до заданого тексту; б) виводить на екран кількість використаних символів; в) видаляє всі слова, що мають подвоєні літери.
4	а) підраховує кількість слів у тексті; б) виводить на екран слово, що містить найбільшу кількість голосних літер; в) видаляє з тексту всі непотрібні пробіли.
5	а) підраховує кількість розділових знаків у тексті; б) виводить всі слова, що мають парну кількість літер; в) міняє місцями першу і останню літери кожного слова.
6	а) підраховує кількість великих літер у тексті; б) виводить на екран слова, що мають найменшу кількість літер; в) видаляє всі слова, що починаються з малої літери.
7	а) підраховує кількість чисел у тексті (не цифр, а саме чисел); б) виводить на екран всі слова, що складаються тільки з латинських літер; в) видаляє кожне друге слово.
8	а) підраховує кількість цифр у тексті; б) виводить на екран слова, що починаються з приголосних літер; в) знищує всі слова, які починаються і закінчуються за одну й ту ж літеру.
9	а) підраховує кількість слів у тексті, які закінчуються на голосну літеру; б) виводить на екран всі слова, довжина яких менша п'яти символів; в) видаляє всі слова, які містять хоча б одну латинську літеру.
10	а) підраховує кількість слів у тексті, які починаються з голосної літери; б) виводить на екран всі слова, що мають непарну кількість приголосних літер; в) видаляє всі числа з тексту.
11	а) замінює всі великі літери, що входять до тексту на відповідні малі; б) виводить на екран найдовше слово; в) видаляє всі слова, що містять непарну кількість приголосних літер.
12	а) кількість слів, які містять однакову кількість голосних і приголосних літер; б) виводить на екран найдовше слово; в) видаляє з тексту всі слова-паліндроми.

13	а) виводить всі символи, які розташовані після першого символу ":"; б) підраховує кількість речень, що містять непарну кількість слів; в) видаляє з тексту всі слова, які розташовані після ком.
14	а) підраховує кількість слів у кожному реченні; б) виводить на екран найдовше речення; в) видаляє всі слова, передостання літера яких голосна.
15	а) інвертує рядок, подаючи його у зворотному вигляді; б) підраховує кількість чисел у тексті; в) видаляє всі слова, що починаються з голосних літер.

Методичні рекомендації

Рядок – це одномірний масив символів в коді ASCII, що закінчується нульовим символом. Кожний символ рядка можна вибрати по значенню індексу з допомогою оператора циклу, а кінець рядка визначається по значенню ***'\0'***- нульовий символ (нуль-термінатор). Наявність нульового символу означає, що кількість комірок масиву повинна бути принаймні на 1 більше, ніж число символів, які необхідно розмістити в пам'яті.

Необхідно розрізняти одиничний символ (символьна змінна), який записується як ***'a', 'b'*** і т. п., та рядок ***"a", "b"***(рядкова змінна). Рядкова змінна може бути ініціалізована декількома способами:

1. char name[] = "Andriy";
2. char name[10] = "Andriy";

Рядок може бути оголошений за допомогою покажчика:

3. char name[] = "Andriy";
   char *nameStr;
   nameStr = name;

Відзначимо, що при присвоєнні покажчику адреси масиву не потрібно використовувати оператор &.

Для вводу-виводу символьних рядків служать наступні функції (заголовочний файл ***stdio.h***) :

gets() - введеннярядка символів із клавіатури;

scanf() - введеннярядка символів із клавіатури (специфікатор %s);

puts() - виведення символьного рядка на екран;

printf() - виведення символьного рядка на екран (специфікатор %s).

1. char str1[20]; gets(str1); // введення з клавіатури;
2. char str2[20]; scanf("%s",str2); // введення з клавіатури;

Для роботи із рядковими змінними служить клас функцій, що знаходяться в заголовочному файлі ***string.h***(всі вони починаються префіксом ***str***).

Функція	Опис	Пояснення
strlen(ch)	повертає дожину рядку ***ch***	Повертає фактичну довжину рядка, не враховуючи нуль-символ
strcmp(ch1,ch2)	Порівнює посимвольно рядки ***ch1*** та ***ch2***	Якщо ***ch1<ch2***, то результат -1, якщо ***ch1=ch2***, то результат 0, якщо ***ch1>ch2***– результат 1
strncmp(ch1,ch2,n)	Порівнює перші ***n*** символів рядків ***ch1*** та ***ch2***	Для перших ***n*** символів: Якщо ***ch1<ch2***, то результат -1, якщо ***ch1=ch2***, то результат 0, якщо ***ch1>ch2***– результат 1
strcpy(ch1,ch2)	Копіює символи рядка ***ch2*** у рядок ***ch1***	Нуль-символ при цьому теж включається
strncpy(ch1,ch2,n)	Копіює ***n*** символів рядка ***ch2*** у рядок ***ch1***	Кінець рядка відкидається. Якщо нуль-символ у вихідному рядку зустрінеться раніше, копіювання припиняється, а решта символів рядка доповнюються «\0»
strcat(ch1,ch2)	Дописує рядок ***ch2*** до рядка ***ch1***	Перший символ ***ch2*** записується на місце нуль-символу рядка ***ch1***. До результуючого ***ch1***додається «\0»
strcatn(ch1,ch2,n)	Дописує перші ***n*** символів рядка ***ch2*** до рядок ***ch1***	***n*** символів рядка ***ch2*** записується до s1, починаючи з місця нуль-символу ***ch1***
strchr(ch,s)	Шукає символ ***s*** у рядку ***ch***	Повертає вказівник на перше входження символу в рядок справа. Якщо його немає - повертається NULL
strrev(ch)	Змінює порядок символів у рядку ***ch*** на протилежний	Дзеркальне відображення рядка ***ch***
strstr(ch1,ch2)	Шукає підрядок ***ch2*** у рядку ***ch1***	Пошук першого входження ***ch2*** у ***ch1***. В разі вдалого пошуку повертається вказівник на елемент з ***ch1***, з якого починається ***ch2***, інакше - NULL.
strtok(ch1,ch2)	Розбиває рядок на лексеми	Функція повертає вказівник на лексему в ***ch1***, відокремлену символом з набору ***ch2*** (пробілами або розділовими знаками)
atoi(ch)	Перетворює рядкову змінну ***ch*** у цифровий формат цілого типу	Повертає числове значення цілого типу без перевірки коректності перетворення
atof(ch)	Перетворює рядкову змінну ***ch*** до типу ***double***	Повертає числове значення дійсного типу подвійної точності без перевірки коректності перетворення

Для роботи із символьними даними служить клас функцій, що знаходяться в заголовочному файлі ***ctype.h***(всі вони починаються префіксом ***is***).

Функція	Опис	Пояснення
isalpha(ch)	Перевіряє чи є символ ***ch*** буквою або цифрою (A-Z, a-z, 0-9)	Повертається ***true***, якщо ***ch*** є буквою або цифрою, інакше false
isspace(ch)	Перевіряє чи є символ ***ch*** пропуском (пробіл, табуляція, символ нового рядка, нової сторінки)	Повертається ***true***, якщо ***ch*** є узагальненим пробілом, інакше false
isdigit(ch)	Перевіряє чи ***ch*** цифрою (0-9)	Повертається ***true***, якщо ***ch*** є цифрою, інакше false
islower(ch)	Перевіряє чи ***ch*** є буквою нижнього регістру	Повертається ***true***, якщо ***ch*** є буквою нижнього регістру, інакше false
isupper(ch)	Перевіряє чи ***ch*** є буквою верхнього регістру	Повертається ***true***, якщо ***ch*** є буквою верхнього регістру, інакше false
ispunct(ch)	Перевіряє чи ***ch*** є символом пунктуації (. , : ; ? ! тощо)	Повертається ***true***, якщо ***ch*** є символом пунктуації, інакше false

Більшість задач обробки рядкових змінних зводяться до аналізу масивів даних з використанням операторів циклу.

Приклади виконання завдань:

Приклад №1	*Приклад №2*
Програма послідовно, міняє символи рядка на *****, і виводить довжину рядка.	Програма перетворює символи із рядка у цифри.
`` char str[]="0123456789"; for (int i=0; i<strlen(str); i++) { str[i]='*'; printf("%s\n" , str); } printf("\nДовжина рядка=%i символів",strlen(str)); _getch(); ```	``` int a; char str2[20]="qwerty 4728"; printf("%s\n" , str2); for (int i=0; i<strlen(str2); i++) { if (str2[i] >= '0' && str2[i] <= '9') { a= str2[i] - '0'; //в цифру printf("\na=%d\n" , a); } } //str2[i]= a + '0'; //в символ ```

Приклад №3	***Приклад №4***
Для виведення рядків на екран крім функції printf () можна використовувати також функцію puts () бібліотеки stdio.h, яка більш проста у використанні. Наступний приклад демонструє застосування даної функції.	Програма вставляє заданий із клавіатури символ у вказане місце, зсуваючи всі символи, що стоять лівіше нього на одну позицію. Для вставки використовується окрема функція (***insert***).

```
#include ...
#define DEF "Основи програмування"

void main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    char str[] = "Перший рядок";
    puts(str);
    puts(DEF);
    puts(&str[7]);
    _getch();
}
```

```
#include ...
void insert (char *str, int n, char c)
{
    for (int i=strlen(str); i>=n; i--)
        str[i+1]=str[i];
    str[n]=c;
}

void main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    char c,str[50];
    int n;
    printf("Ведіть рядок символів");    gets(str);
    printf("Введіть позицію вставки символа");  scanf("%i",&n);
    printf("\nВведіть символ, що буде вставлено");  c=getch();
    printf("\n%c",c);
    insert(str,n,c);
    puts("\nРезультат вставки\n");  puts(str);
    _getch();
}
```

Контрольні питання:

1. Який заголовочний файл використовується у мові С для роботи з символами?
2. Які функції існують для введення і виведення символів?
3. Наведіть основі функції введення\виведення рядків і їх призначення. Який заголовочний файл містить опис цих функцій?
4. Чим відрізняються різні функції введення\виведення рядків? Наведіть приклади їх використання.
5. Як задаються рядки в програмі на мові Сі?
6. Наведіть приклади використання функцій обробки символів.
7. Для чого призначена функція strcpy () і в якій бібліотеці вона визначена?
8. Запишіть можливі способи початкової ініціалізації рядка.
9. Який керуючий символ відповідає кінцю рядка?
10. Що виконує функцію strcmp ()?
11. Що повертає функція strlen () ?

Тема № 22.
Робота з рядками.

Мета: отримання практичних навиків роботи з рядками

Література:

С/С++: Теорія та практика. Навчально-методичний посібник. / В. В. Войтенко, А. В. Морозов. – Житомир: ЖДТУ, 2004. – 324 с.

Зміст роботи

Для обробки рядків використовувати стандартні функції з бібліотечного файлу <string.h>.

Функція main() повинна містити тільки опис рядків і виклики функцій для формування, виводу на екран і обробки рядків.

Завдання 1. Ввести з клавіатури рядок символів і обробити його відповідно до свого варіанту, використовуючи функції.

1	Видалити всі голосні літери з рядка.
2	Підрахувати кількість слів у рядку, в яких більше двох голосних літер.
3	Віддзеркалити кожне слово в рядку, що складається з непарної кількості літер.
4	Видалити кожне слово з рядка, що складається з непарної кількості літер.
5	Відсортувати слова в рядку в лексикографічному порядку (за алфавітом).
6	Видалити з рядка всі слова, що починаються на голосну літеру.
7	Видалити з рядка всі слова, що закінчуються на голосну літеру.
8	Видалити всі приголосні літери з рядка.
9	У заданому реченні знайти найкоротше слово.
10	Перевернути (представити у зворотному порядку) кожне слово у рядку, що складається з парної кількості літер.
11	Видалити кожне слово у рядку, що складається з парної кількості літер.
12	У вихідному реченні перед кожним словом поставити знак «?».
13	У заданому реченні знайти найдовше слово.
14	Видалити із рядка наступні символи: «.», « ,», « ?», « =», «+».
15	У вихідному реченні видалити всі символи пробілу. Вивести перетворений текст і кількість вилучених пробілів.

Завдання 2.Дано рядок зі стандартним набором символів-роздільників між словами. Необхідно:

1. перевірити, чи є у рядку слова паліндроми;
2. визначити кількість повторень кожного слова;
3. знайти слово, що повторюється в рядку максимальну кількість разів;
4. видалити із рядка повторення слів, сформувати рядок із різних слів;
5. підрахувати у рядку кількість слів, що починаються з однаковій літери;
6. переставити слова за спаданням кількості голосних букв;
7. переставити слова в алфавітному порядку;
8. знайти пари слів, довжини яких кратні.

Методичні рекомендації

Приклад виконання роботи без використання функцій:

```
#include<stdio.h>
#include<string.h>
#include<conio.h>

int main()
{
      char slovo[30],x[120];        // опис рядків
      int i,m=0,n,k=0;
      gets(x);                      // введення рядка x
      for(i=0; i<strlen(x); i++)          //цикл до кінця рядка x
            if(x[i]!=' ') k++;            // лічильник символів до пробілу
            else
            {
                  if (k>m){ m=k;n=i;}     //пошук мах значення лічильника к
                  k=0;
            }
            k=0;
            for(i=n-m;i<n;i++)            //вибір із рідка x слова з мах довжиною
                  slovo[k++]=x[i];
            slovo[k]=0;
            printf("%s \n%s\n",slovo,x); /*виведення знайденого слова і всього
рядка x */
            printf("%d %d\n",strlen(slovo),strlen(x)); //вивід довжини
            getch();
return 0;
}
```

Рядки при передачі у функції можуть передаватися як одномірні масиви типу char або як покажчики типу char *. На відміну від звичайних масивів у функції не вказується довжина рядка, т. к. в кінці рядка є /0.

```
int find(char *s,charc)      //Функція підрахунку кількості заданого символу у
рядку
{
      int count=0;
      for (int i=0; i<strlen(s); i++)
            if(s[i]==c) count++;
return count;
}
```

Контрольні питання:

1. Чи існує якийсь специфічний тип даних для зберігання рядків у мові Сі?
2. Який мінімальний розмір масиву повинен бути для зберігання рядка 'abc'? Чому?
3. Чи припустимо явно не вказувати розмір масиву, в якому буде зберігатися рядок? Якщо допустимо, то в яких випадках?
4. Що зберігається в таблиці ASCII?

Тема № 23.
Програмування з використанням структур.

Мета: отримання практичних навиків роботи зі структурами

Література:

С/С++: Теорія та практика. Навчально-методичний посібник. / В. В. Войтенко, А. В. Морозов. – Житомир: ЖДТУ, 2004. – 324 с.

Зміст роботи

Завдання: Зберігання даних необхідно організувати у вигляді масиву структур. У програмі повинні бути реалізовані наступні функції:

1. Створення нового запису;
2. Виведення масиву структур на екран у вигляді таблиці;
3. Пошук запису за параметром;
4. Видалення запису із масиву.

1	«Людина»	прізвище; ім'я; по батькові; стать; національність; зріст; вага; дата народження (рік, місяць число); номер телефону; домашня адреса (поштовий індекс, країна, область, район, місто, вулиця, будинок, квартира).
2	«Школяр»	прізвище; ім'я; по батькові; стать; зріст; вага; дата народження (рік, місяць число); номер телефону; домашня адреса (вулиця, будинок, квартира); школа; клас.
3	«Студент»	прізвище; ім'я; по батькові; стать; національність; дата народження (рік, місяць число); номер телефону; домашня адреса (поштовий індекс, країна, місто, вулиця, будинок, квартира); ВНЗ; курс; група; середній бал; спеціальність.
4	«Покупець»	прізвище; ім'я; по батькові; стать; дата народження (рік, місяць число); номер телефону; домашня адреса (поштовий індекс, країна, область, район, місто, вулиця, будинок, квартира); номер кредитної картки; номер банківського рахунку.
5	«Пацієнт»:	прізвище; ім'я; по батькові; стать; національність; зріст; вага; дата народження (рік, місяць число); номер телефону; домашня адреса (вулиця, будинок, квартира); номер лікарні; відділення; номер медичної карти; діагноз; група крові.
6	«Власник автомобіля»	прізвище; ім'я; по батькові; номер телефону; домашня адреса (місто, вулиця, будинок, квартира) марка автомобіля; номер автомобіля; номер техпаспорта.

7	«Військово-службовець»	прізвище; ім'я; по батькові; домашня адреса (поштовий індекс, місто, вулиця, будинок, квартира); національність; дата народження (рік, місяць число); військова частина; посада; звання.
8	«Робочий»	прізвище; ім'я; по батькові; домашня адреса (місто, вулиця, будинок, квартира); дата народження (рік, місяць число); № цеху; табельний номер; освіта; стаж.
9	«Держава»	назва країни; столиця; державна мова; населення; площа територія; грошова одиниця; державний стрій; глава держави.
10	«Товар»	найменування; вартість; термін зберігання; сорт; дата випуску; термін придатності.
11	«Кінострічка»	назва; режисер (прізвище, ім'я); рік виходу; країна; вартість; дохід; прибуток.
12	«Рейс»	марка автомобіля; номер автомобіля; пункт призначення; вантажопідйомність (у тонах); вартість одиниці вантажу; загальна вартість вантажу.
13	«Книга»	назва; автор (прізвище, ім'я); рік виходу; видавництво; ціна; кількість сторінок; тираж; жанр.
14	«Будівля»	адреса; тип будівлі; кількість поверхів; кількість квартир; термін експлуатації; термін до капітального ремонту (25 років - термін експлуатації).
15	«Авто»	марка; колір; серійний номер; реєстраційний номер; рік випуску; рік техогляду; ціна.

Методичні рекомендації

Структура - це сукупність різнотипних елементів, яким присвоюється одне ім'я. Елементи, що складають структуру, називаються *полями*.

Синтаксис опису структури має вигляд:

```
struct [<ім'я структури>]
{
        <тип 1> ім'я поля 1;
        <тип 2> ім'я поля 2 . . .;
};
```

де struct - службове слово;

<ім'я структури> - ім'я типу структура;
<тип 1>, <тип 2> - тип даних;
ім'я поля 1, ім'я поля 2,... - імена полів структури;

Приклад 1:

```
#include<stdio.h>
#include<stdlib.h>
struct ZrPlat{
char family[20];
int mis;
int summa;
};
int main()
{    //Масив структур
struct ZrPlat z[6] =
     {     {"Ivanov", 1, 20000},
           {"Petrov", 1, 15000},
           {"Fedorov", 1, 4500},
           {"Bubnov", 1, 15500},
           {"Sidorov", 2, 12350},
           {"Vasichkin", 2, 8000}
     };
//Виводиммо массив структур на екран
   printf("\tFamily:\t Mis:    Summa: \n");
for (int i=0; i<6; i++)
                printf("%15s  %3d  %8d\n",z[i].family, z[i].mis,
z[i].summa);
return 0;
      }
```

Приклад 2:

```
#include<string.h>
#include<stdlib.h>
#include<stdio.h>
#define n 4
#define k 3
int main()
{
  struct ZACHETKA
  {
        char last_name[30];
        int no;
        int ocenka[k];
  };
  ZACHETKA* box = (ZACHETKA*)calloc(n, sizeof(ZACHETKA));
  ZACHETKA tmp;
  for(int i=0; i<n; i++)
  {
        printf("Vvedite Familiy\n");
        scanf("%s", &box[i].last_name);
        printf("Vvedite nomer\n");
        scanf("%d",&box[i].no);
        printf("Vvedite ocenki (%d)\n", k);
        for(int j=0; j<k; j++)
              scanf("%d",&box[i].ocenka[j]);
  }
  for(int i=n-1; i>0 ; i--)// Сортування за призвіщем
  for(int j=0; j<i ; j++)
  if(strcmp( box[ j ].last_name, box[ j+1 ].last_name ) > 0)
          {
```

```
                tmp = box[ j ];
                box[ j ]= box[ j+1 ];
                box[ j+1 ]= tmp;
            }
    for(int i=0; i<n; i++) //Виводиммо массив структур на екран
    {
          printf("\n%s ",box[i].last_name);
          printf("%d  ",box[i].no);
          for(int j=0; j<k; j++)
                printf("%d ",box[i].ocenka[j]);
    }
    return 0;
}
```

Контрольні питання:

1. Що таке структура? Коли вона використовується?
2. Як створювати тип структура?
3. Як описувати дані типу структура?
4. Як здійснюється доступ до полів структури?
5. Як можна ініціювати поля структури?

Тема № 24.
Робота з файлами.

Мета: Вивчити програмні засоби для роботи з файлами та потоками

Література:

С/С++: Теорія та практика. Навчально-методичний посібник. / В. В. Войтенко, А. В. Морозов. – Житомир: ЖДТУ, 2004. – 324 с.

Зміст роботи

Завдання: Написати програму, в якій:

1. Зчитується символьний рядок даних з файлу text.txt.
2. Проводиться обробка символьних даних згідно варіанту.
3. Отримані дані записуються в новий файл new_text.txt.

ПРИМІТКА. Словом вважати послідовність символів, обмежену пробілами, комами, крапками, лапками, або їх комбінацією.

1	У символьному рядку замінити всі символи лапок (") на символи дужок що відкриваються та закриваються (лапки з непарними номерами замінити на дужки що відкриваються, а з парними номерами – на дужки що закриваються).
2	В символьному рядку видалити найдовшеслово.
3	Для кожного символу, який не є буквою і пробілом, вказати скільки разів він зустрічається в тексті. Результат записати в новий файл. ПРИМІТКА. Результат записати у вигляді: «символ – скільки разів зустрічається кінець - рядка».
4	Зчитати з клавіатури два числа M і N. Видалити з символьного рядка N символів починаючи з M -тій позиції. ПРИМІТКА. Числа M і N повинні бути менше 100.
5	Підрахувати скільки разів в рядку зустрічається слово «structure», а також знайти починаючи з якого символу це слово зустрічається в тексті. Результат записати в новий файл. ПРИМІТКА. Результат записати у вигляді: «поточний номер слова - structure: номер символа – початку слова».
6	Визначити відсоток голосних і приголосних букв від загальної кількості букв в символьному рядку. ПРИМІТКА. Результат записати у вигляді: «голосні / приголосні: відсоток кінець - рядка».
7	Визначити процентне співвідношення буквених і небукв ених символів в рядку.

8	Підрахувати кількість однобуквених, двобуквених і т.д. слів в рядку. Результат записати в новий файл. ПРИМІТКА. Результат записати у вигляді: «кількість букв в слові - скільки разів зустрічається кінець рядка».
9	Виписати в новий файл всі слова, що мають парну кількість букв
10	Знайти найдовше слово в тексті символьного рядку і замінити в ньому всі символи на #. ПРИМІТКА. Якщо в тексті зустрічається декілька слів з найбільшою довжиною, то зробити заміну в кожному з них.
11	Знайти найкоротший слово в тексті символьного рядка і замінити в ньому всі символи на символ $. ПРИМІТКА. Якщо в тексті зустрічається декілька слів з найменшою довжиною, то зробити заміну в кожному з них.
12	Підрахувати скільки разів в рядку зустрічаються символи «a», «b» та «c». Результат записати в новий файл. ПРИМІТКА. Результат записати у вигляді: «символ – скільки раз зустрічається кінець рядка».
13	В символьному рядку замінити всі символи дужок, що відкриваються на символ *, а символи закриваються дужок на #.
14	Зчитати з клавіатури два числа M та N (M, N <20). Видалити з рядка N слів починаючи з M того слова.
15	Видалити з символьного рядка все однобуквені слова.

Методичні рекомендації

Базові функції для роботи з файлами описані в бібліотеці stdio.h. Вся робота з файлом виконується через файлову змінну - покажчик на структуру типу FILE:

FILE * fp;

Відкрити файл можна функцією fopen, що має 2 параметри:

FILE * fopen (char * ім’я_файлу, char * режим_доступу)

Параметр *ім’я_файлу* може містити відносний або абсолютний шлях до файлу, що відкривається:

1) “test.txt” - файл відкривається з поточної папки.

2) “d: \\ test.dat” - відкривається файл з головної папки диска d:

3) ім'я файлу запитується у користувача:

char buf [80];

printf (“\ nвведіте ім'я файлу:”);

fflush (stdin);

gets (buf);

Параметр *режім_доступа* визначає, які дії будуть дозволені з файлом:

- “r” - відкриваємо для читання текстовий файл;
- “r + b” - відкриваємо для довільного доступу (читання і запис) бінарний файл;
- “a” - відкриваємо текстовий файл для додавання даних в кінець файлу;
- “w” - відкриваємо файл для запису.

Після *відкриття файлу* слід обов'язково перевірити, чи вдалася ця операція. Для цього є 2 основні методи:

- стандартний обробник ferror;
- порівняти покажчик, який повернула fopen, з константою NULL (nullptr) зі стандартної бібліотеки:

fp = fopen (“text.txt”, “r + b”);

if (fp == NULL) {// Обробка ситуації “Не вдалося відкрити файл”}

Приклад. Додаток перевіряє, чи вдалося відкрити файл з поточної папки, ім'я файлу запитується у користувача

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

int main ()
{

	SetConsoleCP (1251);
	SetConsoleOutputCP (1251);
	FILE * fp;
	char buf [80];
	printf ("\nвведіте ім'я файлу:");
	fflush (stdin);
	gets (buf);
	fp = fopen (buf, "r + b");
	if (fp == NULL)
	{
		printf ("\nне вдалося відкрити файл");
		getchar ();
		exit (1);   // Вийти з кодом завершення 1
	}
	else
		printf ("\nфайл відкрито");
	getchar ();
	return 0;
	}
```

Важливо! Функції, які повертають покажчик, в тому числі, fopen, вважаються небезпечними в ряді нових компіляторів, наприклад, Visual Studio

2015. Якщо їх використання призводить не просто до попередження, а до генерації помилок, є 2 основних способи вирішення проблеми:

1. відповідно до рекомендацій компілятора, замінити старі назви функцій на їх безпечні версії: наприклад, strcpy на strcpy_s і fopen на fopen_s. При цьому може змінитися і спосіб виклику функцій, наприклад,

FILE * f; fopen_s (&f, "data.txt", "w");Замість

FILE * f = fopen_s ("data.txt", "w");

2) в початок файлу включити директиву

#define _CRT_SECURE_NO_WARNINGS

Якщо використовується предкомпіляція, то можна визначити цей макрос в заголовки stdafx.h.

Вибір способу читання або запису даних залежить від того, якою має бути структура файлу.

Якщо файл форматований, тобто, є текстовим і складається з лексем, розділених стандартними роздільниками (пробіл, табуляція, переклад рядка), обмін даними з ним можна виконувати функціями:

• fscanf - для читання

• fprintf - для запису

Першим параметром цих функцій вказується файлова змінна, в іншому робота збігається зі стандартними scanf і printf.

Приклад. Файл text.txt має наступний вигляд:

1.5 -3.5

3.5

Прочитаємо його як послідовність дійсних чисел.

```
FILE * fp = fopen ( "text.txt", "r");
    if (fp == NULL){
          printf ( "\nне вдалося відкрити файл");
          getchar (); exit (1);
    }
    float a;
    while (1) {
          fscanf (fp, "%f", &a);
          if (feof (fp)) break;
          printf ( "% .2f", a);
    }
    fclose (fp);
```

Приклад. У файл text.txt запишемо масив a:

```
FILE * fp=fopen("text.txt", "w");
    if (fp == NULL) {
    printf ( "\nне вдалося відкрити файл");
          getchar (); exit (1);
    }
          constint n=5;
          int a[n],i;
    for (i=0; i<n; i++) a[i]=i+1;
    for (i=0; i<n; i++) {
          fprintf (fp,"%5d ",a[i]);
          if ((i+1)%5==0) fprintf (fp,"\n");
    }
    fclose (fp);
```

Функції для роботи з файловою системою

Назва	Що робить
fopen()	Відкриває файл
fclose()	Закриває файл
putc()	Записує символ у файл
fputc()	Записує символ у файл
getc()	Читає символ з файлу
fgetc()	Читає символ з файлу
fgets()	Читає рядок з файлу
fputs()	Записує рядок у файл
fseek()	Встановлює покажчик поточної позиції на певний байт файлу
ftell()	Повертає поточне значення покажчина у файлі
fprintf()	Для файлу те саме, що printf() для консолі
fscanf()	Для файлу те саме, що scanf() для консолі
feof()	Повертає значення true (істина), якщо досягнуто кінець файлу
error()	Повертає значення true, якщо виникла помилка
rewind()	Встановлює покажчик поточної позиції на початок файлу
remove()	Знищує файл
fflush()	Дозапис потоку у файл

Тема № 25.
Робота з файлами.

Мета: Дослідити основні функції роботи з файлами та реалізовувати найпростіші операції з ними. Навчитись застосовувати у своїх програмах текстові і бінарні файли.

Література:

С/С++: Теорія та практика. Навчально-методичний посібник. / В. В. Войтенко, А. В. Морозов. – Житомир: ЖДТУ, 2004. – 324 с.

Зміст роботи

Дослідити процес реалізації завдання на прикладі.

Розробити власну програму, яка реалізує індивідуальне завдання (Лабораторна робота 27).

Підготувати звіт:

- № варіанту і текст завдання;
- лістинг програми;
- схему взаємозв'язку функцій;
- роздруківку вхідних і вихідних файлів і результати виконання;
- висновки.

Завдання 1.

Взявши за основу лабораторнеу роботу №27, змінити код програми таким чином, щоб:

- вхідні дані вводилися не з клавіатури, а з файлу,
- результати виконання виводились і на екран, і у файл.

Завдання 2.

Взявши за основу лабораторну роботу №27, змінити код програми таким чином, щоб:

- вхідні дані (поля структури) вводилися з клавіатури і після введення записувалися у файл (окрема функція);
- програма мала можливість дописувати дані у файл (окрема функція);
- дані з файлу виводились на екран (окрема функція);
- результати виконання другого підпункту виводились на екран і у файл.

Методичні рекомендації

Приклад виконання завдання:

```
#include<stdio.h>
#include<fstream>
#include<conio.h>
int N;

structNNN{
   int n;
```

```
    char name[10];
    char sity[10];
    float den;
}*buf=NULL;

void Buff(FILE *f)
{
    fseek(f,0 ,SEEK_END);
N=ftell(f)/sizeof(NNN);
    rewind(f);
    buf=(NNN*)malloc(N*sizeof(NNN));
    //buf = new NNN[N];
    fread(buf,sizeof(NNN),N, f);
}

void InPut(){
    NNN *buf1=(NNN*)malloc((N+1)*sizeof(NNN));
    for(int i=0;i<N;i++)
          buf1[i]=buf[i];
    free(buf);
    buf=buf1;
    printf("\n%d\n",N);
    printf("\nВведіть n:");
    scanf("%d",&buf[N].n);
    printf("\nВведіть name: ");
    scanf("%s",&buf[N].name);
    printf("\nВведіть sity: ");
    scanf("%s",&buf[N].sity);
    printf("\nВведітьden:  ");
    scanf("%f",&buf[N].den);
          N++;
}

void OutPut(){
    for(int i=0; i<N; i++)
printf("\n%d\t%s\t%s\t%4.2f",buf[i].n,buf[i].name,buf[i].sity,buf[
i].den);
}
void Menu(){
    int menuM=1;
    do{
    cout<<"\nВыберите:\n0-выход\n1-ввод данных\n2-вывод данных\n";
          cin>>menuM;
          switch (menuM){
                case 1: InPut();  break;
                case 2: OutPut(); break;
          default:                break;
          }
    }while(menuM!=0);
}
int main(){
    setlocale(LC_ALL, "Russian");
    FILE *f;
    if((f=fopen("text","rb"))==NULL){
          printf("ERROR\n");
          return 1;
    }
```

```
        Buff(f);
        fclose(f);
        Menu();
                    f=fopen("text","wb");
                    for(int i=0;i<N;i++)
                          fwrite(&buf[i],sizeof(NNN),1,f);
                    fclose(f);
        return 0;
    }
```

Контрольні питання:

1. Яким чином можна оголосити і відкрити файл?
2. Шо таке режими відкриття файлів і які режими ви знаєте?
3. Яким чином можна встановлювати покажчик файлу у задану позицію?
4. Чи можна одночасно використовувати файл для запису і читання?
5. Які режими доступу до файлових потоків ви знаєте?
6. Яким чином можна виводити у файлові потоки і вводити великі обсяги даних (структури, масиви)?
7. Як можна перевірити, чи закінчився файл, і наявність помилок?

Н а в ч а л ь н о - м е т о д и ч н е в и д а н н я

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ
ДЛЯ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ
З ДИСЦИПЛІНИ

“ОСНОВИ ПРОГРАМУВАННЯ”

ЧАСТИНА 2

Підготували
Марчук Галина Вікторівна
Левківський Віталій Леонідович
Чижмотря Олексій Володимирович

Комп’ютерний набір та верстка: Марчук Г.В., Левківський В.Л., Чижмотря О.В.

Підписано до друку __.__.19. Формат 60×84/16.
Ум. друк. арк. ____. Зам. __ офс.

Безкоштовно

Друкарня ЖДТУ